МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Уральский федеральный университет имени первого Президента России Б.Н.Ельцина»

Институт Физико-технологический

Кафедра Технической физики

УТВЕРЖДАЮ

Проректор по учебной работе

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.Т. Князев

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2015 г.

**Математическая логика и теория алгоритмов**

Учебный план № 3771 (в ЕИСУ)

Рекомендовано Учебно-методическим советом Физико-технологического института

для направлений подготовки и специальностей:

|  |  |  |  |
| --- | --- | --- | --- |
| **Код ООП** | **Направление/**  **Специальность** | **Профиль/Программа магистратуры/**  **Специализация** | **Код дисциплины по учебному плану** |
| 09.03.02-02-2011  (230400.62-02-2011) | Информационные системы  и технологии | Информационные системы в технической физике | Б2.2.12.1 |

ФТИ.590;62.20 15

**Екатеринбург, 2015**

**Содержание**

[Введение 4](#_Toc475372157)

[1. Алгебра логики (высказывания и предикаты). 8](#_Toc475372158)

[1.1. Понятие алгебраической системы. 8](#_Toc475372159)

[1.2. Примеры алгебраических систем. 8](#_Toc475372160)

[1.3. Булева алгебра. 9](#_Toc475372161)

[1.4. Алгебра высказываний. 10](#_Toc475372162)

[1.5. Алгебра предикатов. 10](#_Toc475372163)

[1.6. Истинностное значение формулы. 13](#_Toc475372164)

[1.7. Отношение следования. 17](#_Toc475372165)

[1.8. Равносильность. 18](#_Toc475372166)

[2. Булевы функции и нормальные формы. 19](#_Toc475372167)

[2.1. Способы задания. 19](#_Toc475372168)

[2.2. Унарные и бинарные операции. 21](#_Toc475372169)

[2.3. Сведение к нескольким операциям. 22](#_Toc475372170)

[2.4. Теорема Поста. 22](#_Toc475372171)

[2.5. Нормальные формы. 23](#_Toc475372172)

[2.6. Полиномы Жегалкина. 25](#_Toc475372173)

[3. Исчисление высказываний. 25](#_Toc475372174)

[3.1. Общее понятие о логическом исчислении. 25](#_Toc475372175)

[3.2. Правила вывода исчисления высказываний. 26](#_Toc475372176)

[3.3 Аксиомы. 27](#_Toc475372177)

[3. 4. Доказуемость формул. 27](#_Toc475372178)

[27](#_Toc475372179)



[3.5. Выводимость формул. 27](#_Toc475372180)

[3.6. Транзитивность секвенций. 28](#_Toc475372181)

[3.7. Теорема дедукции. 28](#_Toc475372182)

[3.8. Непротиворечивость исчисления высказываний. 29](#_Toc475372183)

[3.9. Полнота исчисления высказываний. 29](#_Toc475372184)

[4. Исчисление предикатов. 30](#_Toc475372185)

[4.1. Теорема дедукции. 31](#_Toc475372186)

[4.2. Правило силлогизма. 32](#_Toc475372187)

[4.2. Вспомогательные правила вывода. 32](#_Toc475372188)

[Структурные: 32](#_Toc475372189)

[Логические: 32](#_Toc475372190)

[4.3. Непротиворечивость исчисления предикатов. 34](#_Toc475372191)

[4.4. Полнота исчисления предикатов. 35](#_Toc475372192)

[4.5. Другие теоремы. 37](#_Toc475372193)

[5. Алгоритмические модели. 38](#_Toc475372194)

[5.1. Понятие алгоритма. 38](#_Toc475372195)

[6. Машины Тьюринга. 38](#_Toc475372196)

[6.1. Описание одноленточной машины Тьюринга. 38](#_Toc475372197)

[6.2 Функции, вычислимые по Тьюрингу. 40](#_Toc475372198)

[6.3. Свойства машин Тьюринга. 41](#_Toc475372199)

[6.4. Тезис Тьюринга. 43](#_Toc475372200)

[7. Частично-рекурсивные функции. 44](#_Toc475372201)

[7.1. Определение частично-рекурсивных функций. 44](#_Toc475372202)

[7.2. Примеры ЧРФ. 45](#_Toc475372203)

[7.3. Тезис Чёрча. 46](#_Toc475372204)

[8. Соотношение между классами «Т» и «Ч». 47](#_Toc475372205)

[9. Нумерации наборов чисел и слов 49](#_Toc475372206)

[9.1. Нумерация пар чисел. 49](#_Toc475372207)

[9.2. Наборов из n чисел. 49](#_Toc475372208)

[9.3. Нумерация множества M=N&N2&… 49](#_Toc475372209)

[9. 4. Нумерация слов. 49](#_Toc475372210)

[10. Нумерация алгоритмов. Невычислимые функции. 50](#_Toc475372211)

[10.1 Нумерация машин Тьюринга 50](#_Toc475372212)

[10.2. Невычислимые функции. 52](#_Toc475372213)

[10.3. Универсальная функция. 52](#_Toc475372214)

[11. Алгоритмически неразрешимые проблемы. 53](#_Toc475372215)

[11.1. Понятие массовой и индивидуальной алгоритмической проблемы. Определение алгоритмически неразрешимой проблемы. 53](#_Toc475372216)

[11.2. Способы доказательства алгоритмической неразрешимости. 53](#_Toc475372217)

[11.3. Полувычислимые и невычислимые алгоритмически проблемы. 55](#_Toc475372218)

[11.5. Другие неразрешимые проблемы из теории рекурсивных функций. 55](#_Toc475372219)

[11.6. Различные примечательные неразрешимые проблемы математики. 56](#_Toc475372220)

[12. Сложность вычислений. 56](#_Toc475372221)

[12.1. Характеристики сложности. 56](#_Toc475372222)

[12.2. Теорема об отсутствии верхней границы сложности вычислений. 58](#_Toc475372223)

[12.3. Теорема Блюма о линейном ускорении. 58](#_Toc475372224)

[13. Классы сложности. 59](#_Toc475372225)

[13.1. Сложность массовой проблемы. 59](#_Toc475372226)

[13.2. Примеры лёгкорешаемых и труднорешаемых задач. 61](#_Toc475372227)

[13.3. Класс «NP». 62](#_Toc475372228)

[13.4. Как соотносятся классы P и NP? 64](#_Toc475372229)

[14. Труднорешаемые задачи. 65](#_Toc475372230)

[14.1. Полиноминальная сводимость задач. 65](#_Toc475372231)

[14.2 Класс «NPC» 65](#_Toc475372232)

[14.3. Основные NP-полные задачи. 66](#_Toc475372233)

[14.3.1. Теорема Кука (NP-полнота задачи «ВЫП»). 66](#_Toc475372234)

[14.3.2. Примеры NP-полных задач: 67](#_Toc475372235)

[14.3.3. Псевдополиноминальные алгоритмы. Сильная NP-полнота. 69](#_Toc475372236)

# Введение

Математическая логика – это наука, изучающая методы доказательства рассуждений и логического вывода с помощью средств математики.

Мат. логика отличается тем, что использует специализированные символьные языки, причём предполагая, что символьный язык может полностью заменить обычный.

Различают язык субъекта и объекта, т. е. язык, который изучают и с помощью которого изучают.

Логика изучает возможности специализированных языков в 2-х аспектах:

* Семантический
* Синтаксический

Логика делится на:

* Теория моделей (семантический подход)
* Теория доказательств (синтаксический подход)

Оба подхода эквивалентны.

Первое логическое учение - силлогистика Аристотеля (модель дедуктивных рассуждений).

Аристотель разделял все суждения на 4 типа:

* А – общеутвердительные: . Пример: все натуральные числа являются целыми.
* I – частноутвердительные: . Пример: некоторые люди — студенты.
* E – общеотрицательные: . Пример: ни один из китов не рыба.
* O – частноотрицательные:. Пример: некоторые люди не являются студентами.

Всё познание состоит в установлении соответствия между разными группами объектов.

В силлогистике рассматриваются заключения, которые следуют из двух посылок.

1 – на этом месте одно из 4-х высказываний (A, I, E, O)

2 – на этом месте одно из 4-х высказываний (A, I, E, O)

3 – заключение

Можно получить 64 комбинации утверждений в рамках доказательства, но возможны не все сочетания, а только 11 являются корректными:

AAA, AAI, AEE, AEO, AII, AOO, EAE, EAO, EIO, IAI, OAO.

В остальных случаях при истинных посылках заключение не всегда является истинным. Пример невыполнения: AEA.

Не любое разрешенное сочетание является корректным. Чтобы выяснить в каких посылках заключение всегда истинно при истинных посылках, нужно рассмотреть более глубокую структуру. Выделяют 4 фигуры силлогизма.

Одинаковый член в 2-х посылках можно исключить

Фигура силлогизма

M

P

M

S

S

P

2)

P

M

S

M

S

P

3)

M

P

S

M

S

P

1)

P

M

M

S

S

P

4)

P – первый (prima)

S – второй (secundo)

M – средний (middle)

Пример: 

Для каждой фигуры есть свой набор сочетаний:

1. фигура: AAA, EAE, AII, EIO.
2. фигура: EAE, AEE, EIO, AOO.
3. фигура: AAI, IAI, AII, EAO, OAO, EIO.
4. фигура: AAI, AEE, IAI, EAO, EIO.

Если у фигуры одно из данных сочетаний и предположение истинно, то вывод всегда истинен.

Канторовская теория множеств противоречива. Вся математика построена на теории множеств, но присутствуют парадоксы.

**1) Парадокс Рассела**

Доказательство: Рассмотрим множество S – множество всех множеств, которые не содержат себя в качестве элемента. Предположим, что множество S не содержит себя в качестве элемента но т.к. S мн. всех множеств, не содержащих себя в качестве элемента =>.

Пусть , но т.к. множество всех множеств, которые не содержат себя в качестве элемента => . Получаем противоречие 

**2) Парадокс Ришара**

Назовем фразой любую последовательность из букв русского алфавита или знаков препинания и пробел. Примеры: «абк», «а в квадрате», «ХА-ХА». Все фразы могут быть упорядочены в лексикографическом порядке «а,б,в,…, ххххх, ….». Некоторые фразы - скажем, приведенная выше фраза «а в квадрате»—являются описаниями одноместных арифметических функций на русском языке. Вычеркнем теперь из нашего пересчета все фразы, не являющиеся такого рода описаниями функций; в результате получим пересчет  всех таких описаний. Обозначим функции, описываемые этими фразами, соответственно через  Рассмотрим теперь следующую фразу: «Функция, значение которой для любого данного натурального числа а в качестве аргумента равно увеличенному на единицу значению для этого же аргумента той функции, которая определяется фразой, соответствующей в только что упомянутом пересчете этому натуральному числу». Мы могли бы заменить в этой фразе ее кусок «в только что упомянутом пересчете» исчерпывающим описанием точной конструкции этого самого пересчета, так что в результате из всей нашей фразы получилась бы некоторая другая фраза Р, полностью описывающая ту же самую функцию. Эта фраза Р описывает некоторую арифметическую функцию, а именно . Следовательно, Р входит в пересчет , но это невозможно, так как функция, описываемая фразой , отличается от функции, описываемой фразой , своим значением для  от функции, описываемой  значением для ; от функции, описываемой , значением для  и т. д. Оформим это рассуждение иначе. Поскольку фраза  входит в пересчет , то она имеет в нем некоторый номер . Значит, подставляя сюда  вместо  и сравнивая с предыдущим равенством, приводим к противоречию.

**3) Парадокс Бери**

Рассмотрим выражение «Наименьшее натуральное число, которое нельзя назвать посредством меньше чем тридцати трех слогов». Это выражение называет некоторое определенное натуральное число - обозначим его через N. Т.к. каждое непустое множество натуральных чисел (в данном случае речь идет о множестве натуральных чисел, которые нельзя назвать посредством меньше чем тридцати трех слогов) имеет наименьший элемент. Согласно своему определению, N нельзя назвать посредством меньше чем тридцати трех слогов. Но ведь наше выражение определяет N, причем с помощью ровно тридцати двух слогов!

**4) Парадокс лжеца**

«Высказывание, которое я сейчас произношу, ложно». Очевидно, что заключенное в кавычки высказывание не может быть ни истинным, ни ложным.

Открытие парадоксов привело к кризису в математике, т.к. математика была сформулирована на теории множеств. Выход был найден с помощью аксиоматического подхода. Задача состояла в том, чтобы найти универсального математического робота, который с помощью формальных процедур мог дать результат, который когда-либо могла дать математика.

Создаем сигнатуры и какие-то базовые формулы, из которых получается все остальные формулы.

Если использую правила вывода можно вывести формулу либо её отрицание, то теория считается **полной.**

Если для формулы нельзя вывести ни формулу, ни отрицание, нужно использовать какую-то более полную теорию. В 1931 Гёдель сформулировал теорему о неполноте: «Всякая достаточно полная теория не полна».

Понятия, характеризующие теорию:

* **Непротиворечивость теории** – для любой формулы теории нельзя одновременно доказать формулу и её отрицание;
* **Полнота** – для любой формулы можно вывести либо формулу, либо отрицание;
* **Разрешимость** – теория называется разрешимой, если для каждой формулы после определенного количества шагов, можно сказать является она истинной или ложной.

Примеры полных теорий: геометрия, теория векторных пространств.

# 1. Алгебра логики (высказывания и предикаты).

1.1. Понятие алгебраической системы.

Пусть есть множество , тогда декартова n-ая степень множество А есть множество упорядоченных наборов по n элементов:

*,где (isN).*

- упорядоченный набор. Причём порядок в наборе имеет значение, т.е. .

Введём n-местное отношение R на определённом множестве А есть подмножество n-степени множества A: (n>0). Любое подмножество множества А есть отношение.

Введём n-местную операцию на множестве А – называется любое отображение декартовой степени множества А на множество А: (n≥0).

- нольместая операция, есть const равная одному из элементов множества.

Каждому упорядоченному набору соответствует элемент из множества А.

**Алгебраическая система** на множестве A – есть множество из совокупности трёх множеств:

* множество А – является носителем алгебраической системы;
* - Множество отношений, определённых на множестве А;
* - Множество операций, определённых на множестве А.

, где , .

Если алгебраическая система не имеет множества отношений, то её называют алгеброй , а если отсутствует множество операций, то это система называется модель .

1.2. Примеры алгебраических систем.

* Множество действительных чисел с операциями сложения и умножения {R, {·, +}};
* Множество целых чисел с операциями сложения и умножения {Z,{ ·, +}};
* Множество рациональных чисел с операциями сложения и умножения {Q, {·, +}};
* Множество натуральных чисел с операциями сложения и умножения {N,{ ·, +}};
* Множество векторов c операцией сложения , (a, b) , a, b R.

С операцией сложения определённой на плоскости

* Алгебра подмножеств:

Пусть A – множество, рассмотрим множество всех подмножеств множества A:

Можно ввести алгебру операций:

- пересечение;

- объединение;

¬­ - дополнение.

* Алгебра подграфов:

- F – полный граф

Есть полный граф – это совокупность 2-х множеств: вершин и ребер, соединяющих эти вершины.

Если множество вершин графа является подмножеством множества F, а множество ребер является подмножеством множества F – это подграф.

Рассмотрим множество всех его подграфов P(F) введя операции сложения, вычитания и дополнения.

{P(F), + , -, ¬}, где “+” – Объединение множества вершин и множества рёбер.

Сложение и вычитание любых подграфов является подграфом.

1.3. Булева алгебра.

Булевой алгеброй называют алгебру, определенной на булевом множестве В с тремя операциями +, · , ¬ и константами 0, 1.

{В, +, · , ¬, 0, 1} – булева алгебра.

Данные операции должны обладать следующими свойствами:

1. Свойство идемпотентности:

а+а=а , а·а=а

1. Свойство коммутативности:

a+b=b+a , a·b=b·a

1. Свойство ассоциативности:

(a+b)+c=a+(b+c) , a·(b·c)=(a·b) ·c

1. Свойство дистрибутивности:

a+(b·c)=a+b·a+c , a·(b+c)=a·b+a·c

1. ¬¬a=a
2. a+0=a , a+1=1 , a·0=0, a·1-a

Простейшее булево множество: В={0,1}.

Более сложное: B={00,01,10,11}.

Определим алгебру на множестве {0,1} с операциями следующим образом:

“+”:(a+b+a·b)mod2

“·”: (a·b)mod2

“”: (a+1)mod2

1.4. Алгебра высказываний.

Высказывание – это повествовательное предложение, которое может быть истинным или ложным.

Простые высказывания (A, B, C) могут объединяться в сложные с помощью пропозициональных связок. Таких связок 5:

1. Конъюнкция

|  |  |  |
| --- | --- | --- |
| **А** | **В** | **&** |
| Л | Л | Л |
| И | Л | Л |
| Л | И | Л |
| И | И | И |

1. Дизъюнкция

|  |  |  |
| --- | --- | --- |
| **А** | **В** |  |
| Л | Л | Л |
| И | Л | И |
| Л | И | И |
| И | И | И |

1. Импликация

|  |  |  |
| --- | --- | --- |
| **А** | **В** |  |
| Л | Л | И |
| И | Л | Л |
| Л | И | И |
| И | И | И |

1. Эквивалентность

|  |  |  |
| --- | --- | --- |
| **А** | **В** |  |
| Л | Л | И |
| И | Л | Л |
| Л | И | Л |
| И | И | И |

1. Отрицание

|  |  |
| --- | --- |
| **А** | **¬А** |
| И | Л |
| Л | И |

Импликация и эквивалентность могут быть выражены следующим образом:

Пример: Если завтра будет солнце, мы пойдём купаться = мы не пойдём купаться или будет солнце.

Пример: Завтра мы пойдём купаться только если будет солнце = если завтра будет солнце, то мы пойдём купаться и если солнце будет мы пойдём купаться.

Пример: Завтра мы пойдём купаться только если будет солнце = если завтра будет солнце, то мы пойдём купаться или если не пойдём купаться то не будет солнца.

**Свойства операций:**

1. Коммутативность
2. Ассоциативность

(

1. Дистрибутивность

(

Носителем алгебры является множество {истина, ложь} и свойства операций алгебры высказывания совпадают со свойствами операций булевой алгебры. Алгебра высказываний – это разновидность булевой алгебры.

Математическое определение высказывания – это пропозициональная функция, отображающая множество т.е. .

Логика высказываний – это булева алгебра, определенная на множестве семантических значений множеств высказываний, включающая операции “НЕ”, ”И”, ”ИЛИ”, импликацию, эквивалентность в соответствии с их таблицами истинности.

1.5. Алгебра предикатов.

Расширением алгебры высказываний является алгебра предикатов.

Маша Любит Некоторое множество (Х)

Предикат Р(Х) - высказывательная форма, содержащая переменную часть .

При подстановки в переменную часть переменной из некоторого множества такая высказывательная форма превращается в высказывание, которое может быть истинным или ложным.

Двуместный предикат:

Y Любит X

Множество, из которого выбираются значения для предиката называться объектным множеством.

Y Любит X Объектное множество

Примеры предикатов:

X > 0, X ∈ Z;

X > Y , X, Y ∈ N.

n-местный предикат может быть ассоциирован с некоторым подмножеством декартовой степени n, для которого значение предиката истинно

1. К предикатам могут быть применены все те же операции, как и к высказываниям

, значения определяются в соответствии с таблицей истинности.

1. Существует дополнительная операция для предикатов – *навешивание кванторов:*

– квантор всеобщности (результат истинный, если значение предиката истинно по всем переменным);

– квантор существования (результат истинный, если среди элементов объектного множества существует такой элемент, при котором предикат принимает истинное значение).

Примеры:

A=N

В результате действия квантора арность предиката уменьшается.

Результирующий предикат от конкретного значения х1 не зависит.

Переменная, по которой действует квантор, называется *связанной*, остальные – *свободными*. Связанная переменная может быть заменена на другую.

Математически? n-местный предикат является пропозициональной функцией, отображающей n-ю декартову степень множества А на булево множество:

Высказывание с этих позиций может быть интерпретировано как отображение

**Определения и формулы алгебры предикатов.**

Алфавит состоит из:

1. x,y,z – индивидные переменные (применяются для обозначения элементов из основного множества). В общем виде – xi, где i=1,2…

Набор пропозициональных букв – A,B,C.

Набор предикатных букв – , где n – арность предикатной буквы, n>0, а j – индекс, j>0.

Скобки, запятые.

1. Определение формулы

Определение построено по принципу математической индукции:

1. Любая пропозициональная или предикатная буква – формула: (А), (Pn(x1…xn));
2. Если А и В – формулы алгебры предикатов, то (АВ), (А&В), (АВ), (АВ), (А) – формулы;
3. Если А – произвольная формула алгебры предикатов, х – индивидная переменная, входящая свободно в форму АЮ то  *,* - формулы алгебры предикатов;
4. Других формул нет!

**Соглашение о скобках.**

Любая предикатная и пропозициональная буква должна заключаться в скобки. Следовательно, необходимо писать:

.

Соответственно, правильными будут являться записи:

А записи вида или является не корректной

При использовании скобок в соответствии с определением порядок действий всегда однозначен.

Для упрощения записи применяется **соглашение о старшинстве:**

таршая операция находится левее.

Будем опускать скобки в случае, если порядок действий будет совпадать с приоритетом операций.

Также можно опускать внешние скобки над всей формулой.

A и (A), также как и , и – это, в общем случае, разные вещи.

Следует заранее обговаривать в каких случаях скобки применяются в одном и том же смысле.

Применение скобок и запятых не должно вызывать противоречие.

**Свободные и связанные переменные**

1) Вхождение переменной х в формулу свободно;

2) Свободное вхождение х в формулу с А и В остается свободным в формулах вида:

(АВ), (А&В), (АВ), (АВ), (А), , ;

3) Свободное вхождение становится связанным в формулах: , ;

4) Связанное вхождение остается связанным в формулах: (А), (А&В), , ;

Формулы, не имеющие свободных вхождений переменных называются ***замкнутыми.***

1.6. Истинностное значение формулы.

Пример:

Имеем формулы

1)

2)

Для того чтобы определить истинностное значение формулы алгебры предикатов необходимо задать её интерпретацию.

Интерпретация = модель + значение свободной переменной, а модель = объектное множество + отношения, соответсвующие предикатным символам.

Например, для формулы (1)зададим следующую интерпретацию:

(I): А: 5>3

B: 3>0

C: 4<3

По соглашению о скобках имеем :

следовательно ()(I)=Л

Чтобы задать интерпретацию (I) Для формулы (2) необходимо

а) Конкретизировать отношение, соответствующее предикату P(x,y):

, R

б) Задать объектное множество A, из которого должны выбираться значения индивидных переменных:

x,y

в) Задать значение свободной переменной y:

y=2

Ф(2)(I)=И Ф(10)(I)=Л

– модель, на которой интерпретируются формулы , т.е. в нашем случае

Запись – означает, что Ф(2) истинно на модели А

А запись  *–* соответствует тому, что Ф(2) ложно на модели А

**Таблица истинности.**

В случае, если конкретная интерпретация не задана, можно пытаться говорить о том, какие истинностные значения в том или ином случае может, в принципе, иметь данная формула. Для формулы алгебры высказываний это можно сделать с помощью Таблицы истинности, в которой указывается какие значения будет иметь формула при конкретных значениях входящих в нее атомов.

В частности, таблица истинности для формулы будет иметь вид:

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** |  |
| И | И | И | И |
| Л | Л |
| Л | И | И |
| Л | Л |
| Л | И | И | И |
| Л | Л |
| Л | И | Л |
| Л | Л |

Для формулы алгебры высказываний **всегда** можем построить конечную таблицу. Для формулы алгебры предикатов этого сделать **нельзя**, т.к.:

1) Не все множества конечны

2) Нас интересуют все возможные множества

Однако возможно построить таблицу истинности формулы в конкретной интерпретации с конечным, объектным множеством.

Пример:

Имеем одноместный предикат Ф(у) и объектное множество A={0,1}. Тогда существует всего 4-е различных варианта пропозициональных функций, соответствующих одноместному предикату

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Y** | **L1** | **L2** | **L3** | **L4** |
| 0 | Л | И | Л | И |
| 1 | Л | Л | И | И |

Множество упорядоченных пар отображается на булево множество И, Л

{<00>, <01>, <10>, <11>} {И, Л}

В случае объектного множества A={0,1, 2} имеем 8 вариантов определения одноместного предиката.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Y** | **L1** | **L2** | **L3** | **L4** | **L5** | **L6** | **L7** | **L8** |
| 0 | Л | И | Л | И | Л | И | Л | И |
| 1 | Л | Л | И | И | Л | Л | И | И |
| 2 | Л | Л | Л | Л | И | И | И | И |

В общем случае: , где ||A||-количество элементов в А.

**Выполнимость формул.**

Определение 1: В алгебре высказываний формула называется ***выполнимой***, если имеется набор истинности значений атомов, при которых значение формулы истинно.

В алгебре предикатов формула ***выполнима***, если существует интерпретация формулы, при которой она истинна.

Определение 2: Множество формул алгебры предикатов называется ***выполнимым***, если существует интерпретация, в которой значения этих формул *одновременно* истинны.

F = (A,B,C)

Чтобы доказать, что формула выполнима, необходимо предъявить интерпретацию, в которой она истинна. В алгебре высказываний нужно предъявить строку таблицы истинности, в которой значение формулы истинно.

Определение 3: Если ни в одной из интерпретаций формула не принимает истинного значения, то она является ***невыполнимой***. Такая формула называется ***противоречие***.

Пример

Ни при каком значении атома А формула не принимает истинного значения

Определение 4: Формулы, истинные в любой интерпретации называются ***тождественно истинными,*** ***общезначимыми*** или***тавтологиями,*** обозначается .

В случаем алгебры высказываний общезначимой будет формула, значение которой истинно во всех строках своей таблицы истинности.

Примеры:

Логические законы:

1) – закон отрицания отрицания

2) – закон отрицания противоречия

3) – закон исключенного третьего

4)

В случае алгебры предикатов доказать общезначимость с помощью таблицы истинности нельзя, однако имеются примеры, для доказательства которых необходимо воспользоваться общими рассуждениями

1)

Доказательство:

Пусть

и - одноместный предикат. Тогда возможны 2-а различных случая:

а) для , т.е.

также истинно, т.к. истинно при любых значениях объектных переменных

имеем импликацию вида ИИ=И

Если предикат является истинным для всех значений индивидной переменной, то формула истинна.

б) Существуют значения индивидных переменных, при которых P(x) будет ложным

т.к. ложная посылка импликацию дает истину.

2)

Доказательство:

а) Существуют значения индивидных переменных, для которых предикат P(y) истинен, тогда

, т.к. заключение истинно

б) , для

Тогда , т.к. *P(y)I = Л*

Для опровержения общезначимости необходимо привести хотя бы 1 интерпретацию или 1 строку из таблицы истинности, в которой формула ложна.

Нахождение всех общезначимых формул – основная задача алгебры логики.

1.7. Отношение следования.

Определение 5: Следование формулы В из формулы А будем обозначать:

1) Формула В **следует** из формулы А если в любой интерпретации, в которой формула А истинна, формула В также истинна.

Пример

Чтобы в этом убедиться, необходимо построить таблицу истинности.

2) Формула А следует из множества формул Г= если во всех интерпретациях, в которых формулы истинны, формула А также истинна.

, … И

(Здесь и далее символ будет соответствовать словесной формулировке «если…, то…»)

Пример

Убедиться можно аналогично, построив таблицу истинности.

**Теорема 1: Теорема о следовании и импликации.**

(Здесь и далее символ будет соответствовать словесной формулировке «тогда и только тогда»)

Необходимость:

т.к. ,,

или ,

Достаточность:

Значит ,

**Обобщение теоремы 1:**

Г, Г, где Г=

**Следствие теоремы 1:**

- формула B следует из множества формул A тогда и только тогда, когда формула A – общезначима. Таким образом, следование равнозначно, когда некая другая формула является общезначимой.

1.8. Равносильность.

Определение 6: Равносильность формулы В формуле А будем обозначать АВ

Формула А равносильна формуле В, если в любой интерпретации I. В случае логики высказываний формулы равносильны, если их таблицы истинности одинаковы, что используется для доказательства равносильности.

Основные формулы равносильности:

1)А А – идемпотентность;

2) ,  – коммутативность;

3)  – ассоциативность;

4)  – дистрибутивность;

5)Законы де Моргана





Для алгебры предикатов необходимо добавить закон коммутативности для кванторов:





и законы де Моргана для кванторов





**Теорема 2. Отношение равносильности связано с отношением следования:**

А  B

**Необходимость:** Пусть AB, то

или

**Достаточность:** (в обратную сторону) и , и когда , то и AB.

**Теорема 3. Связывает отношение равносильности с операцией эквивалентности:**

AB

**Следствие:** AB, BC (, )

**Теорема 4. Теорема об эквивалентной замене:**

Если AB и есть C, в которое A входит в качестве подформулы,



CB – формула, в которой все вхождения формулы А заменены на формулу В.

Эта теорема позволяет изменять цепочки формул без изменения их истинностного значения.

Пример:





# 2. Булевы функции и нормальные формы.

Логическими функциями называются функции от булевых переменных, значения которых есть булева величина. Логическое выражение: BnB

2.1. Способы задания.

**Задание логической функции с помощью таблицы.**

Каждая логическая функция может быть задана с помощью таблицы.

Пример: Задана некая булева функция.

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **F(x,y,z)** |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Всего 2n логических наборов переменных (комбинаций).

Сколько может быть различных булевых функций от n переменных?

2n-количество различных упорядоченных наборов из n переменных. Логическая функция от n переменных может быть задана как 2n различных наборов, при этом каждому набору должно соответствовать одно из двух значений.

Таким образом, при n=1, получаем 22=4 логических функций f(x).

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | ***x*** | **0** | **1** |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |

При n=2 ,получаем 16 функций.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **x** | **y** | **(x)** | **(y)** | ***x*** | ***y*** | **0** | **1** |  |  |  |  |  |  |  |  | **|** |  |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

**(x)**– константа х **(y)**– константа y

***x*** – отрицание х ***y*** – отрицание y

**0** – константа 0 **1** – константа 1

**-** конъюнкция – Дизъюнкция

- Импликация - эквивалентность

- сравнение x>y - сравнение x<y

- исключающее или - обратная импликация

**|** - штрих Шеффера («И НЕ») - стрелка Пирса («ИЛИ НЕ»)

**Задание логической функции с помощью формулы.**

Логическую функцию можно задать с помощью формулы через унарные и бинарные операции.

Пример:

У нас есть некая бинарная функция и.

Если вместо одной из переменных поставить любую бинарную функция, то получим функцию от 3-х переменных.

В целом, любая функция может быть задана как таблицей, так и формулой.

2.2. Унарные и бинарные операции.

Рассмотрим свойства унарных и бинарных операций:

1. **Коммутативность**

Коммутативными называют логические функции или логические операции, если они обладают следующим свойством:

x # y = y # x, где # - это &, v, ~, |, ↓, \oplus

1. **Ассоциативность**

(x # y) # z = x # ( y # z), где # - это &, v, ~, \oplus

1. **Дистрибутивность**

x # (y Ъ z) = (x # y) Ъ (x # z), где (#,Ъ) - (&, v), (v,&),( &, \oplus)

1. **Обратные операции**

¬(x#y)=xЪy, где (#,Ъ) - (&,|), (v,↓), (>,→), (<,←), (~,\oplus)

2.3. Сведение к нескольким операциям.

В частности все бинарные операции и логические функции могут быть сведены к трем операциям: .

**Преобразования:**

**а)** все логические функции могут быть выражены через

**б)** через

**в)** через

**г)** можно выразить одной операцией

**д)** через :

Определение 1: Множество функций является **полным**, если можно выразить все другие функции через функции этого множества.

Как определить какое множество функций является полным? Ответ на вопрос дает теорема Поста.

2.4. Теорема Поста.

**Введем 5 различных классов функций:**

1. F0 – функции сохраняющие «0».

: , если .

При нулевых значениях аргумента, имеют значение - ноль.

1. F1 – функции сохраняющие «1».

, если .

В случае, когда все значения аргумента=1,значение фнкции=1.

1. F 2– монотонные функции.

есть два набора

,

n-местная логическая функция принадлежит классу монотонных функций, если у нас есть 2 набора , причем , а это значит, что тогда значение

.

Т.е. если набор увеличивается, значение функций также должно увеличиваться, поэтому функция монотонная.

1. F 3 – линейные функции.

где C0 ….Cn –являются булевыми константами (0 или 1).

1. F 4– самодвойственные функции.

Двойственной функцией называется функция вида:

Самодвойственной функцией называется функция двойственная самой себе.

**Теорема Поста.**

**Определение:** Множество функций является полным, если можно выразить все логические функции через функции этого множества.

**Теорема Поста:** Для того, чтобы множество было полным, необходимо и достаточно, чтобы среди этого множества нашлась хотя бы одна функция, не сохраняющая 0, одна не сохраняющая 1, одна не монотонная, одна нелинейная и одна не самодвойственная функция.

2.5. Нормальные формы.

**СДНФ. Совершенная дизъюктивная нормальная форма.**

СДНФ – называется представление логической функции в виде дизъюнкций элементарных конъюнкций

F= C1 V C2 V…V Cs **,** где Ci –элементарная конъюнкция.

Элементарная конъюнкция имеет следующий вид:

C= L1 & L2 & … & Ln**,**  где L1=x или L1= ¬ x.

**СКНФ. Совершенная конъюктивная нормальная форма.**

СКНФ – называется представление логической функции в виде конъюнкций элементарных дизъюнкций.

F= D1 & D2 & … & Dn**,** где D1 - элементарная дизъюнкция.

Элементарная дизъюнкция имеет следующий вид:

D= L1 V L2 V … V Ln, где L1=x или L1= ¬ x.

**Теорема 1:** Любая логическая функция может быть представлена в виде СДНФ.

Доказательством теоремы является алгоритм, который позволяет из любой логической функции получить СДНФ.

Пусть логическая функция задана таблицей истинности.

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | **y** | **x** | **f** |
| 0 | 0 | 0 | 0 |
| **0** | **0** | **1** | **1** |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **1** |
| 1 | 1 | 1 | 0 |

Берем строчки, где значение функции =1 и составляем из них элементарные конъюнкции, причем если значение переменной в таблице 1, то она входит в элементарную конъюнкцию в прямой форме, иначе входит отрицание этой переменной.

СДНФ: (¬x & ¬y & z) V (x & ¬y & z ) V (x & y & ¬z)

Полученная формула является ДНФ, кроме того она имеет значение совпадающее со значением функции f. Поскольку если мы берем набор, который соответствует значению 1, то какая-то элементарная конъюнкция имеет значение 1.

Таким образом мы доказали теорему о том, что любая логическая функция может быть представлена в виде ДНФ.

А также теорема доказывает, что логическая функция может быть представлена выражением элементарных унарных и бинарных логических значений.

**Теорема 2:** Любая логическая функция может быть представлена в виде СКНФ.

Доказательством теоремы является алгоритм, который позволяет из любой логической функции получить СКНФ.

Пусть логическая функция задана таблицей истинности.

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | **y** | **x** | **f** |
| **0** | **0** | **0** | **0** |
| 0 | 0 | 1 | 1 |
| **0** | **1** | **0** | **0** |
| **0** | **1** | **1** | **0** |
| **1** | **0** | **0** | **0** |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| **1** | **1** | **1** | **0** |

Берем строчки, где значение функции =0 и составляем из них элементарные дизъюнкции, причем если значение переменной в таблице 0, то она входит в элементарную дизъюнкцию в прямой форме, иначе входит отрицание этой переменной.

СКНФ: (x V y V z ) & (x V ¬y V ) & (x V ¬y V ¬z) & (¬x V y V z) & (¬x V ¬y V ¬z).

Полученная формула является КНФ, кроме того она имеет значение совпадающее со значением функции f. Поскольку если мы берем набор, который соответствует значению 0, то какая-то элементарная конъюнкция имеет значение 0.

2.6. Полиномы Жегалкина.

F= C 1 C2 …. Cn ,где - элементарная конъюнкция

Для того, чтобы получить полином, нужно в СДНФ выразить «V и ¬» через «& и » в соответсвии с формулами и затем, с помощью преобразований, привести к соответствующему виду.

# 3. Исчисление высказываний.

3.1. Общее понятие о логическом исчислении.

Для определения логического исчисления необходимо задать:

1. Язык
2. Правила вывода
3. Аксиомы

Рассмотрим пример определения логического исчисления:

1. Язык
   1. Алфавит А
   2. Формула (будем считать, что в нашем исчислении формулой является произвольная последовательность букв А)
2. Правила вывода 
3. Аксиомы: (посылочные правила)

Таким образом, мы получили исчисление, в котором может появиться произвольная последовательность букв . Полученное логическое исчисление не является содержательным.

Чтобы исчисление было содержательным оно должно удовлетворять двум требованиям:

1. Правила вывода должны быть корректными (при истинных посылках, истинные следствия)
2. Аксиомы должны быть общезначимы.

**Исторически сложившиеся содержательное исчисление высказываний:**

Язык совпадает с языком алгебры высказываний, содержит пропозициональные буквы, пропозициональные связки. Формулы совпадают с формулами алгебры высказываний.

3.2. Правила вывода исчисления высказываний.

Будем использовать 2 правила вывода:

* 1. Правило отделения (modus ponens): 
  2. Правило подстановки (S):  Каждую букву заменяем на формулу. Конечная формула будет истинна, если в каждой общезначимой формуле заменили каждую букву на формулу.

Корректность правил:

1. Если , правило отделения корректно.
2. Пусть есть формула, которая истинна, если в ней буквы заменить на формулы, то она может быть И или Л, то и новая формула И исходная формула является общезначимой.

3.3 Аксиомы.

1. 
2. 
3.  - аксиома введения 
4.  - аксиома удаления 
5.  - аксиома введения 
6.  - аксиома удаления &
7.  - аксиома введения 
8. - правило удаления 
9.  - аксиома введения 
10.  - аксиома удаления 

Если вместо пропозициональных букв будем использовать любую формулу исчисления высказываний, то можем получить бесконечное множество аксиом заданной структуры (схем аксиом).

3. 4. Доказуемость формул.

Определение 1: Доказательством формулы  является конечная последовательность формул , каждая из которых является либо аксиомой, либо получена применением одного из правил вывода, а в конце последовательности стоит формула .

Пример доказательства формулы: 

 - по первой аксиоме

 - используем подстановку

- по второй аксиоме

 - подстановка 



 - подстановка в формулу 1 



## 



3.5. Выводимость формул.

Пример: (вывод  из посылок )

 - аксиома удаления &

- первая посылка

 - modus ponens 1.2

- аксиома удаления &

- первая посылка

- modus ponens 4,5

- вторая посылка

- modus ponens 3,7

- modus ponens 6,8



Возникает вопрос: Можем ли мы вместо С подставить любую формулу?

Ответ: Нельзя, правило подстановки имеют ограничение, его нельзя применять к пропозициональным буквам, которые содержатся в формулах, введенных в качестве посылок.

Определение 2 : Секвенцией (выводом из посылок) называется последовательность формул , где - либо аксиома, либо получена по правилу вывода, либо , где - множество посылок вывода.

3.6. Транзитивность секвенций.

**Теорема 1: Правило силлогизма.**

Если существуют выводы , то существует вывод

Доказательство:

Вывод - означает, что есть последовательность формул . Вывод - означает, что есть последовательность формул . Запишем два вывода последовательно , полученная запись и есть вывод.

3.7. Теорема дедукции.

**Теорема 2: Теорема дедукции.**

Импликация доказуема тогда и только тогда, когда формула ****выводима из .

Доказательство:

Необходимость:

Пусть формула  является доказуемой. Это означает что существует последовательность . Добавим в качестве посылки:  получили.

Достаточность: не будем рассматривать в силу громоздкости вывода.

**Следствие Теоремы 2:** ,

Если существует вывод формулы из множества , то формула является доказуемой.

3.8. Непротиворечивость исчисления высказываний.

**Теорема 3: Теорема об общезначимости доказуемых формул.**

Если формула доказуема, то она общезначима.

Доказательство:

Пусть есть некоторая последовательность формул , где является либо аксиомой, либо получена из правил вывода. Каждая формула в последовательности является общезначимой все формулы в последовательности являются общезначимыми формула общезначима.

**Следствие 1 Теоремы 3: О непротиворечивости исчисления высказывания.**

Ни для какой формулы из исчисления высказываний не может быть доказана и сама формула и её отрицание.

Доказательство:

Предположим, что формула доказуема это означает, что она общезначима . Общезначимая формула в любой интерпретации истинна . Теперь рассмотрим формулу , она будет ложна в любой интерпретации это означает, что  не общезначима. По теореме об общезначимости .

3.9. Полнота исчисления высказываний.

**Теорема 4: Теорема о полноте исчисления высказываний.**

Если формула общезначима, то она доказуема

**Следствие 1 Теоремы 4:**

Если к системе аксиом ИВ добавить произвольную формулу , которая не является общезначимой, то найдется такая формула, для которой будет существовать .

( E-не общезначима)

**Нельзя усилить ИВ добавлением недоказуемой формулы к системе аксиом.**

Из теорем 3 и 4 следует: , значит, что любая общезначимая формула доказуема, а также добавив не общезначимую формулу, мы получим противоречивое исчисление.

Получили эквивалентность синтаксического и семантического подхода в рамках исчисления высказываний.

# 4. Исчисление предикатов.

Для того чтобы задать исчисление предикатов необходимо задать:

1. **Язык (алфавит)**. Он совпадает с языком логики и включает в себя: пропозициональные буквы, предикатные буквы, индивидные переменные и формулы.
2. **Аксиомы**. Все аксиомы исчисления высказываний актуальны для исчисления предикатов. При этом чаще говорят о схемах аксиом. Кроме того, в ИП добавлены две дополнительные схемы аксиом для кванторов:
   1. 
   2. 

Где x,y – произвольные термы, которые могут совпадать, а , т.е. формула получается из путем замены y на x. При этом, если x и y разные термы, x не может входить свободно в формулу

Терм – это запись, которая может принимать значения из объектного множества. В узком смысле, термы – индивидные переменные.

1. **Правила вывода** в исчислении предикатов включают правило отделения (*modus ponens*) и два дополнительных правила для кванторов. Правило подстановки, ввиду использования *схем аксиом* становиться в ИП не актуальным.
   1. Правило всеобщности. ()

, где С – формула исчисления предикатов, х – не входит свободно в С. Тогда можно записать .

* 1. Правило существования ()

, если х не входит свободно в С, то мы можем записать 

Дополнительные правила вывода должны быть корректны (т.е. при истинных посылках давать истинные следствия):

Правило всеобщности: имеем ,

Возможны два случая

1. Пусть тогда должно быть Истинно при всех ;
2. Пусть ,тогда при любых .

Правило существования: имеем

1. Пусть существует значение x0, такое что
2. Пусть для всех значений x .

Определения доказательства и вывода ИП - расширение ИВ: аксиомы совпадают + 2 аксиомы для кванторов, правила вывода аналогичны+2 новых для кванторов.

Доказательство формулы A – это некоторая последовательность формул которая заканчивается формулой , причем формулы являются аксиомами или формулами, полученными с помощью правил вывода.

Вывод формулы A из множества формул Г – это некоторая последовательность формул которая заканчивается формулой , причем формулы являются аксиомами, формулами, полученными с помощью правил вывода или формулы являющиеся посылками т.е. .

При этом должно выполнятся следующее

Замечание: Дополнительные правила вывода нельзя применять к переменным, введенным в качестве посылки. Такие переменные называются **фиксированными** переменными.



 - фиксированные переменные.

**Пример вывода:**

- допущение

- аксиома всеобщности (, )

- modus ponens 1,2

- аксиома всеобщности

 - modus ponens 3,4

4.1. Теорема дедукции.

- формулы исчисления предикатов, - множество формул исчисления предикатов.

Необходимость:

Пусть тогда имеем вывод  где  . Далее запишем  в качестве посылки и применим правило отделения

Достаточность;

Это доказательство мы опустим в связи с его громоздкостью.

4.2. Правило силлогизма.

Если , то существует вывод где - формулы ИП.

Для доказательства воспользуемся теоремой дедукции:

Тогда имеем вывод используя правило отделения получим

Теорему о дедукции и правило силлогизма можно записать в виде вспомогательных правил вывода. Вспомогательные правила вывода используются для доказательства значения факта, что вывод или доказательство существует.

- правило введения импликации (из теоремы о дедукции);

– правило силлогизма.

Если существует вывод, записанный сверху, то существует вывод записанный снизу.

4.2. Вспомогательные правила вывода.

## Структурные:

1. – закон тождества
2. – правило добавления
3. - правило перестановки
4. – правило сокращения посылки
5. – правило сечения (обобщенное правило силлогизма)

## Логические:

1. Правило введения и удаления импликации
2. Правило введения и удаления «И»
3. Правило введения и удаления «ИЛИ»
4. Правило введения и удаления отрицания
5. Правило введения и удаления эквивалентности
6. Правило введения и удаления квантора всеобщности

Примечание:

y не входит свободно во множество формул Г;

x не входит свободно в A(y);

A(x) получается заменой всех вхождений y на x в A(y).

1. Правило введения и удаления квантора существования

Примечание:

Всё так же как и в пункте 6, но y не входит свободно и в формулу C.

Для того чтобы можно было использовать вспомогательные правила вывода они должны быть предварительно доказаны. Для примера докажем правило введения и удаления эквивалентности:

Введение эквивалентности:

1. Г, А ⊢ В (по теореме о дедукции) Г ⊢ А → В
2. Г, В ⊢ А (по теореме о дедукции) Г ⊢ В → А
3. (А → В) → ((В → А) → (А ~ В)) - аксиома введения эквивалентности
4. Используем modus ponens два раза: (3, 1) и (3, 2)
5. Получаем А ~ В
6. Т.е. получили Г ⊢ А ~ В

Удаление эквивалентности:

1. Г ⊢ А
2. Г ⊢ А ~ В
3. (А ~ В) → (А → В) - Аксиома удаления эквивалентности
4. А → В - modus ponens 3, 2
5. В - modus ponens 4, 1
6. Получили Г ⊢ В

С помощью вспомогательных правил вывода покажем существование следующего вывода:

⊢ (А ∨ В) ~ ¬(¬А & В)

1. А, (¬А & ¬В) ⊢ А - Закон тождества и добавление посылки
2. А, ¬А, ¬В ⊢ ¬А - Закон тождества и добавление посылки
3. А, (¬А & ¬В) ⊢ ¬А - Правило удаления «И»
4. А ⊢ ¬(¬А & ¬В) - Введение отрицания
5. В ⊢ ¬(¬А & ¬В) - Введение отрицания
6. (А ∨ В) ⊢ ¬(¬А & ¬В) - Введение «ИЛИ»
7. А, ¬(А ∨ В) ⊢ (А ∨ В) - По правилу введения «ИЛИ»
8. А, ¬(А ∨ В) ⊢ ¬(А ∨ В) - Закон тождества и добавление посылки
9. ¬(А ∨ В) ⊢ ¬А - Введение отрицания
10. ¬(А ∨ В) ⊢ ¬В - Введение отрицания
11. ¬(А ∨ В) ⊢ (¬А & ¬В) - Правило введения «И»
12. ¬(¬А & ¬В), ¬(А ∨ В) ⊢ (¬А & ¬В) - Добавление посылки
13. ¬(¬А & ¬В), ¬(А ∨ В) ⊢ ¬(¬А & ¬В) - Добавление посылки
14. ¬(¬А & ¬В) ⊢ ¬¬(А ∨ В) - Введение отрицания
15. ¬(¬А & ¬В) ⊢ (А ∨ В) - Удаление отрицания
16. ⊢ (А ∨ В) ~ ¬(¬А & В) - Введение эквивалентности (используя 15 , 6)

Эта последовательность доказывает, что существует вывод (А ∨ В) ~ ¬(¬А & В)

4.3. Непротиворечивость исчисления предикатов.

Ранее было доказано, что исчисление высказываний является полным и непротиворечивым. Является ли исчисление предикатов непротиворечивым и полным?

Определение 1: Множество Г называется **выполнимым**, если существует интерпретация, в которой все формулы, входящие в это множество истины, т.е. Г – выполнимо, если ∃I: для А ∈ Г.

Определение 2: Множество Г называется **непротиворечивым** множеством, если не существует какой либо формулы, для которой не существует Г ⊢ Е и Г ⊢ ¬Е одновременно, т.е. из Г невозможно вывести противоречие.

**Теорема 1:**

Если формула А выводима из множества Г (Г ⊢ А), то во всех интерпретациях, где множество Г истина, формула А является истинной (Г ⊨ А).

Доказательство:

Существует вывод А: где либо ∈ Г , либо – аксиома, либо получена с помощью правил вывода. Аксиома является общезначимой формулой т.е. истина в любой интерпретации. Правила вывода являются корректными т.е. при истинной посылки дают истину.

Получается что все формулы Г истины, в том числе и А ,т.е. при И.

**Следствие 1 Теоремы 1:**

Любая доказуемая формула является общезначимой т.е. ⊢ Е => ⊨ Е

Доказательство:

Тот же вывод, как и в теореме только без посылок.

**Следствие 2 Теоремы 1:**

Не для какой формулы ИП не существует доказательство формулы и её отрицания одновременно

⊢ Е и ⊢ ¬Е

Это отражает непротиворечивость исчисления предикатов.

4.4. Полнота исчисления предикатов.

Рассмотрим полноту исчисления предикатов.

Определение 3:

Множество Т называется полным если для любой формулы исчисления предикатов существует вывод этой формулы из Т, либо существует вывод отрицания этой формулы,т.е. для ∀Е ∈ ИП существует вывод Т ⊢ Е или Т ⊢ ¬Е

**Теорема 2.**

Всякое непротиворечивое множество выполнимо. Г – непротиворечиво => Г – выполнимо

Рассмотри вспомогательную Лемму для доказательства теоремы.

Для любого непротиворечивого множества S существует полное непротиворечивое множество включающее в себя множество S, т.е. S c T ,где Т – полное, непротиворечивое множество.

Доказательство:

Пусть существует множество S и формулу φ S при этом S φ, причем если φ добавить в S, то – непротиворечиво.

Рассмотрим иерархию множеств

S c c …

Процесс включения будет продолжаться пока не получится некоторое максимальное непротиворечивое множество включающее S. Его получим т.к. множество всех формул больше чем любое непротиворечивое множество формул.

- даст уже противоречивое множество, где ∀f и f,

т.е. существует формула, Ψ для которой существует , f ⊢ Ψ и , f ⊢ ¬Ψ одновременно.

По правилу введения отрицания можем записать что ⊢ ¬f , а это означает, что максимальное непротиворечивое множество является полным т.е. .

**Теорема 3. О выполнимости.**

Всякое непротиворечивое множество выполнимо, следовательно, существует такая интерпретация, при которой все формулы множества истинны.

I: <M, >, где М – объектное множество, *σ* – множество отношений, которые определяют предикатные символы, <M, > - модель.

М = {x1,…,xn,y1,…,yn} – множество всех термов (символов объектных переменных)

Pn(x1,…,xn)(I)=

Т – полное, непротиворечивое множество, включающее S

Доказать, что на модели <M, >: А(I)= И T A

Доказательство: (по методу математической индукции)

1. База индукции

Для А = Pn(x1,…,xn) – данное утверждение выполняется по определению отношений ;

1. Шаг индукции
2. А = ¬А1
3. А = А1 А2
4. А = А1 & А2
5. А = А1 А2
6. А = А1 А2
7. A = x A1



1. A = x A1



Докажем, что если для формул A1, А2 выполняется:

А1(I)= И , А2(I)= И Т ⊢ А1 , Т ⊢ А2

То для формулы А выполняется:

А(I)= И T ⊢ A

1. А = ¬А1

Необходимость

Докажем, что для А = ¬А1 теорема выполняется.

А(I)= И А1(I)= Л Т ⊢ ¬А1  T ⊢ А

Достаточность

T ⊢ А т.к. Т – полное, непротиворечивое множество, то : Т ⊢ ¬А1  Т ⊬ А1 А1=Л А=¬А1=И

1. А = А1 & А2

Необходимость

А(I)= И А1=И и А2=И Т⊢ А1 и Т ⊢ А2 //по правилу введения И// Т ⊢ А1 & А2  T ⊢ А

Достаточность

(от противного)

Т ⊢ А1 & А2 . Пусть А1 & А2 = А = Л

Т.к. Т ⊢ А1 & А2  // аксиома удаления И// А1 & А2 →А1 //modus ponens// А1  Т ⊢ А1 ; Т ⊢ А2

А1 = Л или А2 = Л Т ⊬ А1 или Т ⊢ А2

Противоречие! А(I) = А1(I) & А2(I) = И

…

Таким образом, в рассматриваемой интерпретации значение формулы будет истинно тогда и только тогда, когда данная формула выводима из T

А(I)=И Т ⊢ А.

Поскольку S T для любого f S T ⊢ f. Соответственно, f S истинны ( f(I) = И) множество S – выполнимо.



**Теорема 4. О полноте ИП.**

Любая общезначимая формула доказуема.

⊨E ⊢E

Доказательство:

ЕI=И ¬ ЕI = Л {¬E} – невыполнимо. По теореме о выполнимости {¬E} – противоречиво существует формула ψ для которой ¬E ⊢ ψ; ¬E ⊢ ¬ ψ //по правилу введения НЕ// ⊢E (формула Е – доказуема) ч.т.д.

4.5. Другие теоремы.

**Теорема 5. О непротиворечивости.**

Любое выполнимое множество формул является непротиворечивым.

Ф – выполнимо Ф – непротиворечиво

Доказательство:

( от противного)

Пусть Ф – выполнимо, но противоречиво найдется формула ψ, для которой можно вывести противоречие.

Ф ⊢ (ψ & ¬ψ) существует некая конечная последовательность формул , , …, ψ & ¬ψ Ф1=Ф {,…,} ⊢ ψ & ¬ψ //по теореме о дедукции// ⊢ (…(  (ψ & ¬ψ)…)



Но т.к. любая доказуемая формула является общезначимой //по теореме о полноте//

(…(  (ψ & ¬ψ)…) //по теореме импликации и следовании// … ψ & ¬ψ

(связаны отношением следования)

Т.к. (ψ & ¬ψ)I=Л => (Ф1) =Л – невыполнимо Ф – невыполнимо. Противоречие! Ф – выполнимо и непротиворечиво.

**Теорема 6. Теорема Мальцева (теорема компактности).**

Если множество выполнимо, то выполнимо любое его конечное подмножество.

Ф – выполнимо S – выполнимо, S Ф и S – конечно.



Необходимость: Доказательство тривиально.

Достаточность:

Предположим, что все S- выполнимы, а Ф – невыполнимо Ф – противоречиво ( по теореме о выполнимости) найдется такая формула ψ, для которой Ф ⊢ ψ & ¬ψ (вывод всегда конечен) S Ф , S – конечно, S ⊢ ψ & ¬ψ S является противоречивым, S не выполнимо. Противоречие! Ф – выполнимо.



**Теорема 7. Об адекватности.**

Если формула *φ* следует из множества формул Ф, то формула *φ* выводится из Ф.

Ф Ф ⊢

Необходимость:

Ф {¬ } – невыполнимо противоречиво найдется такая формула ψ, что можно вывести и ψ и ¬ψ т.е. Ф, ¬ψ ⊢ ψ и Ф, ¬ψ ⊢ ¬ψ Ф ⊢ .



Достаточность:

Уже доказывали (теорема 1) Ф ⊢ Ф

**Семантический и синтаксический подходы эквивалентны как в исчислении предикатов, так и в исчислении высказываний.**

# 5. Алгоритмические модели.

5.1. Понятие алгоритма.

Алгоритм – конечная последовательность шагов, приводящая к получению результата из исходных данных.

Такое определение алгоритма является интуитивно ясным, но не определяемым понятием.

Проблемы, которые подвели к проблематике алгоритмов:

1. Алгоритмическая разрешимость. Имеет ли данная задача алгоритм решения?
2. Наибыстрейший алгоритм решения. Определение нижней границы сложности решения.

В рамках интуитивного определения алгоритма нельзя разрешить данные алгоритмические проблемы, поэтому рассмотрим, что именно в интуитивном понятии алгоритма нуждается в уточнении.

1. Что такое данные (алгоритмический объект)
2. Память. Способ хранения данных
3. Элементарное действие, команда
4. Последовательность порождения операций, шагов
5. Что есть результат
6. Способ реализации

Многочисленные исследования показали, что данные моменты невозможно определить строго. Поэтому на настоящий момент общепризнано, что Алгоритм это – фундаментальное, интуитивно ясное, но неопределяемое понятие.

Поэтому в теории алгоритмов принят подход, основанный на использовании алгоритмических моделей. **Алгоритмическая модель** – строго очерченный класс алгоритмов, в котором все указанные выше моменты интуитивного определения алгоритма, нуждающиеся в уточнении, уже строго определены. Для того чтобы результаты, полученные с помощью конкретной алгоритмической модели, могли быть распространены на любой алгоритм, используемые алгоритмические модели должны быть Универсальными (т.е. алгоритм, имеющий реализацию в любой разумной алгоритмической модели, имеет аналог в рамках данной). При этом, универсальность каждой конкретной алгоритмической модели является недоказуемым утверждением и оформляется в виде соответствующего тезиса.

Различают 3 класса алгоритмических моделей:

1. Класс как описание некоторого технического устройства: машина Тьюринга, машина прямого доступа, машина Поста.
2. Описание алгоритма как некоторого класса функций с определенными способами их образования: частично рекурсивные функции (исторически первая формализация понятия алгоритма).
3. Набор правил по преобразованию слов: алгорифм Маркова.

# 6. Машины Тьюринга.

6.1. Описание одноленточной машины Тьюринга.

Техническое устройство, состоящее из:

1. Ленты разделенной на ячейки и бесконечной в обе стороны. В каждой ячейке может быть записан один из символов некоторого множества называемого внешним алфавитом. При этом один из символов будем считать пустым символом
2. Управляющее устройство, которое может находиться в одном из внутренних состояний – внутренний алфавит.

Из набора внутренних состояний выделяют которые объявляются и считаются исходным и заключительным состояниями.

1. Считывающая головка, которая всякий раз может обозревать только одну ячейку, может перемещаться вдоль лент и может считывать, стирать напечатанные символы и печатать новые
2. Команда. Работа машины Тьюринга проходит потактно в зависимости от внутреннего состояния машины и считываемого символа на ленте машина Тьюринга:
3. Записывает в эту ячейку символ внешнего алфавита;
4. Сдвигает считывающую головку на один шаг влево или один шаг вправо или оставляет её на месте;
5. Переходит в новое внутреннее состояние.

Таким образом, работа машины определяется системой команд вида

,

где - внутреннее состояние машины, - считываемый символ, - новое внутреннее состояние, - новый записываемый символ, - перемещение головки вдоль ленты: L – влево, R – вправо, E – на месте.

Элементарная операция строго определена и однозначна. Для каждой пары, где i=1…n, j=0…m должна быть одна и только одна команда такого вида. Множество этих команд называется программой машины. В программе имеется n(m+1) команд.

Работа начинается с внутреннего состояния , как только машина попадает в состояние машина Тьюринга останавливается, а слово, которое окажется на ленте является результатом.

Состояние машины Тьюринга в каждый момент времени (конфигурация) представляет собой совокупность внутреннего состояния, состояния ленты (т.е. размещения букв или слова внешнего алфавита по ячейкам), положения головки на ленте.

α

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | … |  | … |  |  |

β

Конфигурация

Работа машины заключается в изменении конфигураций.

Пусть некая конфигурация имеет следующий вид:

α

Тогда после выполнения команды вида новая конфигурация будет иметь вид:

Стандартная начальная конфигурация имеет вид.

Тогда работу машины Тьюринга можно представить в следующем виде

в зависимости от слова на ленте

…. и набора команд

Если приходит в состояние то машина применима к начальной конфигурации. Если же нет, то неприменима.

Если машина перешла в состояние и при этом считывающая головка находится на первой ячейке, то говорят что машина находится в канонической(стандартной) форме. Если машина не в стандартной форме т.е. находится на произвольной ячейке, то машину всегда можно привести к канонической форме, путём добавления новых команд.

*,* где - новое заключительное состояние.

При этом новая машина Тьюринга является эквивалентной старой машине, в том смысле, что она будет применима только к тем конфигурациям к которым применима исходная машина и результаты вычислений обеих машин будут одинаковыми.

6.2 Функции, вычислимые по Тьюрингу.

Пусть имеется некий алфавит .

*–* множество слов этого алфавита.

Рассмотрим словарную функцию отображающую множество слов алфавита на себя.

f : .

Пусть существует Машина Тьюринга такая, что:

1. Если определено и , то машина Т применима к начальной конфигурации и заключитальной конфигурацией является
2. Если не определена, то машина Т неприменима к начальной конфигурации

Тогда машина вычисляет словарную функцию ; является функцией вычислимой по Тьюрингу.

Чтобы построить машину, вычисляющую числовую функцию необходимо договориться о схеме кодирования чисел, т.е. способе погружения элементов числового ряда во внешний алфавит МТ.

Простейшей схемой кодирования является унарная кодировка.

n ∈ N

n кодируется n+1 палочкой на ленте Ι…Ι

при этом Ι соответствует n=0

**Пример:** Рассмотрим машину Т вычисляющую функцию

Набор команд будет следующим

Для того чтобы на машине Тьюринга можно было вычислять функции нескольких переменных договориться и добавить во внешний алфавит МТ символ разделитель, например, «\*».

**Пример:** Рассмотрим машину Т вычисляющую сумму двух переменных в унарной кодировке.

, Q={}

Пример вида ленты

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Ι | Ι | \* | Ι | Ι | Ι |  |

Набор команд машины Т:

6.3. Свойства машин Тьюринга.

**Теорема 1. Существование суперпозиции.**

Существование суперпозиции МТ. Пусть есть МТ вычисляющая функцию и МТ вычисляющая функцию , тогда существует МТ вычисляющая суперпозицию

Доказательство:

Tf1: А1, Q1=

Tf2: А2, Q2=

Тогда машину Т можно сконструировать следующим образом:

Tf: А= А1А2, Q=

Конечное состояние первой машины и начальное состояние второй машины объединим в одно состояние.

Переобозначим символы: Q2=.

Программы машины Tf будут составлены из программ машин и .

Рассмотрим, как будет работать машина Tf:

p

**Теорема 2. Машина «ИЛИ».**

Пусть f(ε,p) = , где ε – постоянная, которая может принимать значения 0 или 1.

Принцип работы:

, при

, ε\*p

, при

Доказательство:

Построим такую МТ:

А= А1А2, Q=, где , .

МТ необходимо дополнить следующими командами:

0 λR и 1λR

\* λR \* λR

Нетрудно убедится, что полученная машина будет вычислять функцию f(ε,p). Данный прием является универсальным и может быть применен для любых машин и .

**Теорема 3. Машина «И».**

Пусть f(p)=

Принцип работы машины является следующим:

Она должна после работы оставлять на ленте оба результата через разделитель

= p\*

Доказательство:

А = , где , А1А2, = .

Тогда машина «И» должна будет проходить через следующие фазы:

Машина Тьюринга «И» - это суперпозиция машин , , и .

**Теорема 4. Машина Тьюринга, реализующая цикл.**

Пусть существуют машины:

Тогда существует машина TC, реализующая цикл:

Ф(р)=1

: Р Ф()=0

Ф(р)=0

Ф( )=1 и т.д.

Блок-схема алгоритма, реализующего данный цикл выглядит следующим образом:

Да

P

Ф(р)\*р

Ф(р)=1

нет

Таким образом, любую процедуру можн реализовать с помощью суперпозиции машин T1 и T2.При этом нужно, дополнительно, объединить заключительное состояние машины с начальным состоянием машины T1, а заключительное состояние машины  сделать общим заключительным состоянием машины TC. Нельзя придумать такую, чтобы невозможно было реализовать.

6.4. Тезис Тьюринга.

Мы видим, в этом нас убеждает общая практика вычислений:

* Существуют машины, реализующие основные алгоритмические конструкции. Во всех алгоритмах другие конструкции можно представить через эти.
* Любой алгоритм всегда может быть реализован с помощью МТ (на данный момент не нашлось алгоритмической процедуры, которая не может быть реализована с помощью МТ).

Из этого следует тезис Тьюринга: любая вычислимая функция является вычислимой по Тьюрингу (В=Т)

Фактически, этот тезис утверждает универсальность машин Тьюринга.

# 7. Частично-рекурсивные функции.

7.1. Определение частично-рекурсивных функций.

Будем рассматривать функции натурального аргумента:

f(x), где xN

всюду определенная функция частично определенная функция

Определим три базовые функции:

1. Ноль-функция

0(х)=0

Всюду определенная функция, значение которой всегда равна нулю.

1. Функция следования

s(x)=x+1

1. Функция взятия аргумента

, где

m –m-местная функция,

n – порядковый номер аргумента, значению которого равно значение функции.

Определим три типа операций, с помощью которых можно получать новые функции из уже имеющихся:

1. S – суперпозиция:

Пусть нам даны функции: g(и ( … (.

Тогда можно определить новую функцию:

f(= S(g, )= g(( … ())

1. R - примитивная рекурсия:

Пусть нам даны функции: g(=g() – n-местная функция и h( – (n+2)-местная функция.

Тогда может быть определена новая (n+1)-местная функция:

f(=R(g,h)= ,

т.е. функция определяется через себя же.

Предположим, что некоторое значение при , тогда для y>значение функции будет не определено (f() – не определено f(y) при y> будет не определено).

Как S, так и R, будучи применимыми ко всюду определенным функциям, дают всюду определенную функцию.

1. M – операция минимизации:

Пусть дана n-местная функция g(. Будем рассматривать уравнение относительно y (y):

g(=

Решать будем следующим образом: будем перебирать значения y, начиная с 0. Как только получим первое значение, которое удовлетворяет уравнению, мы получим минимальное решение данного уравнения.

=M(g(=)

Определение: **Частично рекурсивные функции (ЧРФ)** – это функции, полученные с помощью конечного числа использований операций *R, M, S* к функциям *0*(x), *s*(x), (Под x везде понимаем множество ).

Данное определение можно представить в следующем виде:

1. , где - класс ЧРФ;
2. ;
3. ;
4. ;
5. Других ЧРФ нет!

7.2. Примеры ЧРФ.

1. Функция является рекурсией от функций

, где

1. Sg(x) =
3. функция предыдущего значения
4. – целая часть от деления
5. - остаток от деления
6. - отлична от нуля в конечном количестве точек
7. - количество простых чисел < x
8. – простое число

Дополнительно, в классе ЧРФ можно выделить 2 подкласса:

- общерекурсивные функции;

**-** примитивно рекурсивные функции - функции, полученные без операции минимизации;

Возникает вопрос о соотношении подклассов ***Ч0*** и ***Чпр***: Существует ли функция принадлежащая классу которую нельзя было бы получить с помощью операций ***R*** и **S**? Поскольку очевидно, что все примитивно рекурсивные функции являются общерекурсивными. Ответ на этот вопрос даёт функция *Аккермана* которая определятся следующими реккуретными соотношениями:

Данная функция является рекурсивной (При этом, она возрастает быстрее чем любая функция, принадлежащая к классу т.к. нашлась функция которая является , но не является ., т.е.:

7.3. Тезис Чёрча.

Фактически, любая функция, которую можно вычислить? может быть получена путем использования операций *S, M, R* к функциям *0(x), s(x),* . Таким образом, можно утверждать, что ЧРФ также являются универсальной алгоритмической моделью. Это утверждение называется Тезис Черча:

Всякая вычисляемая(рекурсивная) функция является ЧРФ

, т.е. классы вычислимых и частично-рекурсивных функций совпадают (В = Ч)

Как тезис Тьюринга, данное утверждение является принципиально недоказуемым т.к. в нём утверждается совпадения класса ЧРФ с принципиально неопределяемым классом вычислимых функций. Подтверждением тезиса Чёрча является вся существующая вычислительная практика и

# 8. Соотношение между классами «Т» и «Ч».

Из тезисов Тьюринга и Черча следует что класс функций вычислимых по Тьюрингу совпадает с классом ЧРФ (в том случае если эти тезисы верны).

Т.е. если ,то тезис Тьюринга или Черча не верен, либо они оба не верны.

Это можно проверить, т.к. оба этих класса являются строго определенными.

**Вычислимость по Тьюрингу ЧРФ**

Для того чтобы доказать, что необходимо доказать, что:

1) ***Ч ⊂ Т***

2) ***Т ⊂ Ч***

Для того чтобы доказать первое утверждение необходимо доказать что:

1. являются функциями вычислимыми по Тьюрингу.
2. Операций *S, M, R* являются вычислимыми по Тьюрингу.

Докажем второй пункт.

Вычислимость суперпозиции

Построим машину вычисляющую суперпозицию.

т.е. существуют

Вычислимость рекурсии:

Существуют .

В общем случае после каждого цикла

Да

Нет



Вычислимость минимизации:

В общем случае после каждого цикла

Да



0

Нет

Существует доказательство, что T с Ч Ч = T (данный вывод мы опустим).

Совпадение классов ЧРФ и функций вычисляемых по Тьюрингу показывает важное косвенное свидетельство истинности тезисов Тьюринга и Черча.

# 9. Нумерации наборов чисел и слов

9.1. Нумерация пар чисел.

где - упорядоченная пара.

Упорядочиваем пары x+y=n по n, а внутри скобок < > упорядочивание происходит следующим образом:

Например для x+y=n :

Т.е. начало будет выглядеть так:

При таком способе упорядочивания можем ввести функцию - характеризующая номер упорядоченной пары.

– канторовский номер.

Такая нумерация называется Канторовской.

Также возможно по известному номеру определить x и y

Для

9.2. Наборов из n чисел.

Тогда в общем случае

9.3. Нумерация множества M=N&N2&…

Упорядочим элементы множества M которое имеет вид:

- все возможные наборы натуральных чисел, т.е. произвольный упорядоченный набор чисел.

где n – количество чисел в наборе.

…

9. 4. Нумерация слов.

Нумеровать можно не только наборы чисел, но и наборы слов.

1. Пусть есть алфавит множество всевозможных слов и символов этого алфавита и хотим упорядочить слова, т.е. однозначно поставить каждому слову в соответствие номер. - слово из при этом .

Можно интерпретировать каждый символ, как r-ичную цифру, тогда слово будет представлено s-значным числом, равным .

1. Пусть количество символов в алфавите бесконечно и пусть такое [бесконечное множество](http://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%81%D0%BA%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE) является счётным. Тогда алфавит называется счётным алфавитом. Можно поставить каждому символу в соответствие номер. Тогда слово может быть интерпретировано как набор чисел .

А как пронумеровать набор чисел было показано выше. Обозначим канторовский номер слова в счетном алфавите следующим образом где - канторовский номер набора чисел.

Для пустого символа канторовский номер будет равен нулю

# 10. Нумерация алгоритмов. Невычислимые функции.

10.1 Нумерация машин Тьюринга

В качестве внешнего и внутреннего алфавита возьмём объединение алфавитов всех возможных машин Тьюринга.

- множество всех возможных внутренних состояний МТ

Внутренний и внешний алфавит являются бесконечными и счётными. Тогда можно ввести объединенный алфавит: .

Упорядочим символы в этом алфавите следующим образом:

С учётом заданного перечисления символов объединенного алфавита, каждому символу можно сопоставить номер.

В рамках объединённого алфавита каждая конкретная машина Тьюринга будет отличаться только набором команд:

Упорядочим команды каждой по левой части в лексикографическом порядке и запишем их в одно слово, в соответствии с порядком следования:

*.*

Очевидно, что полученное слово будет иметь вполне определённую структуру. В частности, оно состоит из пятёрок символов, при этом первый и третий символы соответствуют символам состояний, второй и четвёртый, символам на ленте, пятый является командой движения головки и т.д.

При этом существует достаточный набор признаков, по которым можно однозначно отличить слово, являющееся программой МТ от слова, которое программой не является.

Если заменить каждый символ соответствующим номером, получится упорядоченный набор чисел, являющийся числовым кодом данной МТ:

При этом, коды всех машин Тьюринга можно упорядочить в порядке возрастания их канторовских номеров.

Следует учитывать, что **канторовский номер кода машины, не равен номеру машины Тьюринга**, т.к. машины при упорядочивании расположены по мере возрастания их канторовских номеров, но не подряд, например, в таком виде:

Упорядочивание является эффективным, т.е. можно по коду МТ определить её номер и по номеру восстановить код и программу МТ.

Для определения номера какой-либо конкретной МТ по её коду необходимо подсчитать количество МТ, имеющих канторовский номер, который меньше номера данной машины. Соответствующую процедуру можно представить в виде следующего алгоритма:

Нет

Да

Р

Нет

Да

Для получения кода МТ по номеру следует последовательно необходимое количество канторовских номеров, являющихся номерами МТ, а затем по найденному таким образом номеру восстановить код машины. Сделать это можно с помощью следующего алгоритма:

Нет

Да

Нет

Да

Перечисление МТ позволяет построить эффективное перечисление ЧРФ, в котором каждая ЧРФ будет находиться под номером первой МТ, которая ее вычисляет (т.к. ЧРФ может быть вычислена бесконечным количеством МТ). Так как по коду МТ можно восстановить ЧРФ, перечисление МТ является также эффективным перечислением ЧРФ.

Существование эффективного перечисления алгоритмов имеет два замечательных следствия:

1) Существование невычислимых функций.

2) Существование универсальной функции.

10.2. Невычислимые функции.

Рассмотрим следующую функцию

Функция всюду определена и не является ни одной из функций в перечислении ЧРФ. Т.к. в перечисление ЧРФ находятся все возможные ЧРФ, данная функция не является ЧРФ, и? следовательно, по тезису Черча, функция не является вычислимой функцией, т.е. она не может быть вычислена с помощью какой-либо алгоритмической процедуры. Очевидно, что подобным можно предложить бесконечное множество невычислимых функций. Например

Вывод: Существует бесконечное множество невычислимых функций.

10.3. Универсальная функция.

Существование универсальной функции.

Определение: Пусть существует класс ,то

*-* называется **универсальной** функцией класса F если:

Т.е. функция, содержащая в себе все функции одного класса и только их.

**Теорема 1. О существовании универсальной функции класса одноместных ЧРФ.**

Рассмотрим класс одноместных ЧРФ

И рассмотрим функцию

Докажем, что функция является универсальной функцией для класса одноместных ЧРФ.

1. , т.к. существует алгоритм ее вычисления

тогда ,

Функция *f(x)* принадлежит ЧРФ т.к. является суперпозицией ЧРФ

1. Выберем некоторую функцию из класса ЧРФ, для ;

*-* по определению функции U, то U –универсальная функция класса одноместных ЧРФ (класса ).

**Следствие Теоремы 1.**

Для класса n-местных ЧРФ существует - местная универсальная функция

Доказательство:

1. т.к. является суперпозицией ЧРФ
2. Берем произвольную n-местную ЧРФ

И определим новую функцию от канторовского номера

– все переменные выражены только через канторовский номер

т.к. является суперпозицией

– универсальная одноместная функция, тогда

- универсальная ЧРФ

Для классов справедливы следующие теоремы:

**Теорема 2.** Не существует универсальной функции для класса общерекурсивных функций .

**Теорема 3.** Для класса n-местных примитивно рекурсивных функций существует универсальная -местная общерекурсивная функция т.е. для .

# 11. Алгоритмически неразрешимые проблемы.

11.1. Понятие массовой и индивидуальной алгоритмической проблемы. Определение алгоритмически неразрешимой проблемы.

Под массовой алгоритмической проблемой будем понимать бесконечные серии подобных задач, которые отличаются друг от друга только набором значений параметров.

1. Индивидуальная проблема : найти решение квадратного уравнения

Массовая проблема: найти решение произвольного квадратного уравнения *,* где

1. Для функции :

–индивидуальная проблема;

для произвольного *xi* – массовая проблема;

Будем рассматривать алгоритмические проблемы, которые дают ответ «Да» или «Нет». Такие задачи называют задачами распознавания.

С каждой задачей распознавания может быть связана характеристическая функция.

, где - набор параметров из П – массовая проблема;

Если характеристическая функция является вычислимой, то соответствующая массовая проблема является алгоритмически разрешимой, в противном случае является не разрешимой.

11.2. Способы доказательства алгоритмической неразрешимости.

**1. Проблема самоприменимости МТ**

Пусть есть некоторая машина Тьюринга Т, применима ли она к конфигурации Код(Т):

Пример самоприменимой МТ:

Пример не самоприменимой МТ:

Проблема заключается в определении для любой МТ по её коду, является ли она самоприменимой или нет.

Докажем что не существует МТ способной решить данную проблему.

Доказательство: будем доказывать от противного. Пусть существует машина Тьюринга М которая решает проблему самоприменимости.

Сконструируем машину которая будет включать в себя все команды машины М + ещё две новых команды:

, где - новое заключительное состояние

очевидно, что машина будет применима к не самоприменимым и никогда не остановится если применить к кодусамоприменимой машины T.

Является ли машина самоприменимой?

Пусть самоприменима, тогда:

получили противоречие.

Тогда предполагая, что машина не самоприменима:

*.*

Из этого следует, что машины существовать не может начальное предположение о существовании машины М, решаемой проблему самоприменимости ложно не существует машин, решаемых проблему самоприменимости.

**2.Метод сведения**

Пусть существуют две алгоритмические проблемы . И существует некоторая функция которая преобразует набор , причём так, что:

()

Причём выполняется следующее условие:

Тогда говорят, что задача (проблема)

**Теорема 1.** Если задача сводится к задаче и задача алгоритмически неразрешима, то и задача также алгоритмически неразрешима.

Доказательство: Пусть задача не разрешима, а задача - разрешима, тогда для справедливо:

– вычислима

И по определению сводимости имеем для задачи :

- снова вычислима

Таким образом получили противоречие. Следовательно, алгоритмически не разрешима.

**3.Проблема останова**

Есть произвольная машина Т и произвольное слово Р, остановится ли машина с начальной конфигурацией ?

Проблема самоприменимости сводится к проблеме останова, т.к. выбрав в качестве слова Р=Код(Т) получим проблему самоприменимости. Т.к. проблема самоприменимости является не разрешимой, то и проблема остатова также является алгоритмичести не разрешимой.

11.3. Полувычислимые и невычислимые алгоритмически проблемы.

**1. Проблема «*x ∈ Wx*»**

Определить для произвольного числа X принадлежит ли оно области определения ЧРФ стоящей под номером X.

Доказательство:

Рассмотрим характеристическую функцию данной проблемы

и функцию следующего вида

Можно утверждать, что если вычислима , то тоже вычислима. Пусть g(x) вычислима, тогда она стоит в ряде ЧРФ под некоторым номером n т.е.

Определена ли функция g(x) в точке n ()?

Пусть g(n) определена, это означает что , т.е. получили противоречие.

Если предположить, что g(n) не определена тогда , т.е. g(n) определена, тогда Снова получили противоречие.

Следовательно исходная посылка, что g(n) вычислима не верна и функция g(n) не вычислима, а это значит, что функция f(x) также не вычислима.

**2.Полувычислимые проблема**

Рассмотрим две функции отвечающие двум проблемам:

- отвечает проблеме «»

- отвечает проблеме «»

Обе эти функции не вычислимы. Однако функции вида:

- также является не вычислимой

- данная функция уже является вычислимой

Видно, что функция g(x) является «более невычислимой», чем f(x), а проблема «» называется полувычислимой проблемой.

11.5. Другие неразрешимые проблемы из теории рекурсивных функций.

* Принадлежит ли произвольное число Y области определения ЧРФ под номером X. (проблема «»)
* Проблема определения принадлечит ли функция в ряду ЧРФ под номером X классу общерекурсивных функций (проблема «»)
* Проблема определения, является ли функция под номером X является нуль-функцией. (проблема «»)
* . Являютия ли две произвольные функции, взятые по д номером x и y из ряда ЧРФ, одинаковы (проблема «»)
* Проблема определения обладает ли ЧРФ под номером Х произвольным нетривиальным свойством (монотонность, неотрицательность и др.)

11.6. Различные примечательные неразрешимые проблемы математики.

1. Формула в элементарной арифметике состоит из:

Т.е. записи вида:

Проблема заключается в том чтобы для любой произвольной формулы сказать является ли она истиной или нет.

1. Разрешимость логики предикатов первого порядка.

Является ли формула логики предикатов истинной?

1. Проблема сочитаемости Поста

Даны алфавит и множество пар слов в алфавите А. Из - множество конечных слов в алфавите А.

P, Q PQ называются сочетаемыми, если существует такой набор так, что (k-сколь угодно большое).

Проблема состоит в том является ли произвольное множество пар слов сочетаемым.

1. Проблема представимости матриц.

Представима ли матрица U в базисе

Cущес твует ли такой набор где так чтобы

В этом случае рассматриваются – матрицы и неразрешимость данной проблемы начинается с n=4.

1. Проблема тождества элементарных функций вещественного переменного. Т.е. функций:

, где *U* и *V* это вещественные переменные и константы или функции полученные суперпозицией этих функций. Проблема состоит в определении являются ли две произвольне функции тождественными ()

1. Десятая проблема Гильберта. Проблема разрешимости в целых числах диофантового уравнения.

Задано диофантово уравнение (т.е. уравнение вида:) с произвольными неизвестными и целыми коэффициентами. Необходимо после конечного числа операций установить, разрешимо ли данное уравнение в целых числах.

# 12. Сложность вычислений.

12.1. Характеристики сложности.

1. Пусть есть машина Тьюринга – Т, которая вычисляет функцию f(x)

Мерой сложности является каличество шагов вычисления.

Определим функцию , равную числу шагов до остановки - n машины Т, выполненному при вычислении f(x), если f(x) определена. Если f(x) не определена, то значение считается неопределенным. Функция  **называется временной сложностью.**

( n-количество шагов до остановки);

Активной зоной при работе машины Т на слове X называется множество всех ячеек ленты, которые содержат непустые символы, либо посещались в процессе вычисления f(x) головкой машины Т.

Определим функцию , равную длине активной зоны при работе машины Т на слове X, если f(x) определена. Если f(x) не определена, то значение считается неопределенным. Функция  **называется емкостной(ленточной) сложностью.**

Пусть машина Тьюринга Т имеет внешний и внутренний алфавит мощности k и r соответсвенно.

2. Тогда для функций сложности и справедливы оценки:

Покажем это:

В начальный момент на ленте записано слово x длины . На каждом шаге в активной зоне становится не более одной новой ячейки, поэтому

Найдём теперь число различных конфигураций с активной зоной . Имеется вариантов записи на ленте, S вариантов выбора положения головки и S вариантов для значения длины конфигурации К. Таким образом, число рассматриваемых конфигураций не превосходит (r-мощность внутреннего алфавита, k-мощность внешнего алфавиита).

Далее, если в процессе работы машины встретятся две одинаковые конфигурации, то машина зациклиться, поскольку конфигурация однозначно определяет следующую за ней. Значит, если машина в процессе вычисления использует зону , то число её шагов не превосходит числа различных конфигураций с зоной не превышающей В итоге получаем неравенство

3. Вводятся функции временной и емкостной сложности по худшему случаю:

Отметим свойства мер сложностей:

Т.е. области определения функций сложности и функция сложности, которая эти функции отражает должны совпадать.

2) Предикаты вида P(x,y) определенные соотшениями является разрешимым.

Т.е. существует машина Т разрешающая предикаты

В начале на ленте написано слово x, считаем количество шагов y до того как на ленет будет написано f(x): если машина остановится значит y, если не осановится то не y. Таким образом всегда можно сказать, что функция f(x) имеет значение y или не имеет.

Определение абстрактной мерой сложности. Под абстрактной мерой сложности будем понимать любое семейство функций которое обладает свойствами:

1. Преликат – разрешим

Примеры: 1. Значение максимального символа при перечислении в качестве меры сложности.

2. Количество внутренних состояний в качестве меры сложности.

12.2. Теорема об отсутствии верхней границы сложности вычислений.

**Теорема 1. О не существовании верхней границы сложности вычисления.**

Для любой общерекурсивной функции (с помощью которой пытаемся ограничить сложность) существует общерекурсивная 0,1-функция такая, что существует значение , для которого справедливо .

Т.е. ограничения на сложность вычисления не существует. Не существует общерекурсивной функции которая ограничила бы сверху сложность вычисления любой функции. Всегда найдётся такая функция сложность вычисления которой больше чем любое предположенное значение.

**Теорема 2. О существовании сложновычислимой функции.**

Для любой общерекурсивной функции существует общерекурсивная функция с коротким ответом (0, 1) такая, что существует значение , такое что выполняется условие при всех т.е.:

1. Для всех x, кроме конечного числа точек;
2. Сложность вычисления которой будет почти во всех точках x т.е. почти для всех x.

12.3. Теорема Блюма о линейном ускорении.

**Теорема 3. Теорема Блюма (о несуществовании нижнего предела сложности ).**

Для любой общерекурсивной функции существует общерекурсивная функция с коротким ответом (0, 1) такая, что для любой машины вычисляющей f(x) найдется машина вычисляющая f(x) такая, что .

Рассмотрим следствия этой теоремы:

1. Пусть функция , тогда существует функция f(x), которая допускает вычисление со сложность на машине *Ti*, и для которой возможно ускоренное вычисление на машине *Tj*, причём . В свою очередь для справедливо ускорение на машине *T’j* : Т.е. существует бесконечная последовательность машин, допускающих всё большее ускорение вычисления функции f(x)

Таким образом, не существует нижней границы сложности вычисления, т.к. существуют задачи которые всегда можно ускорить.

1. Пусть функция и есть процессор который совершает 1 шаг за 1 нс, тогда общее время выполнения программы *Ti* будет (нс)

Возьмём более быстрый процессор, который будет совершать 2 шага за 1 нс, тогда общее время выполнения программы на новом процессоре будет (нс)

Используем новый алгоритм решения и запустим его на первом процессоре, тогда общее время выполнения программы будет , т.е.

Таким образом, общее время на выполнение программы на старом процессоре но с новым алгоритмом меньше затраченного времени на выполнение такой же задачи на новом процессоре, но по старому алгоритму. Из этого следует вывод, что поиск новых алгоритмов эффективнее, чем построение более быстродейственной техники. Необходимо искать новые алгоритмы решения задач.

# 13. Классы сложности.

13.1. Сложность массовой проблемы.

Каждая массовая проблема может быть решена на определенной машине Тьюринга. Пусть имеется массовая задача , характеризуемая множеством параметров ,  - индивидуальная задача, в которой эти параметры фиксированы. Пусть с массовой задачей  связана и зафиксирована схема кодирования , которая ставит каждой индивидуальной задаче в соответствие слово в некотором алфавите *А*. При этом размером задачи  понимается длинна машинного слова 

- отображение, погружающее набор параметров индивидуальной массовой проблемы во внешний алфавит МТ.

Если индивидуальная задача имеет решение, работа МТ будет выглядеть следующим образом:



Сложность задачи можно охарактеризовать функциями сложности:

1. , где  - асимптотическая временная сложность (т.е. сложность по худшему случаю), -длина машинного слова.
2. , где  - асимптотическая емкостная сложность, -длина машинного слова.

Возможно ли оценить нижнюю границу сложности?

Существуют методы прямого получения нижних оценок, например, **метод следов**.

Методом следов можно определить сложность следующих задач:

1. Является ли произвольное слово p - симметричным. 
2. Является ли P декартовым квадратом 

Установление прямых нижних оценок сложности задач затруднительно. В связи с этим получил подход, связанный с получением косвенных нижних оценок, т.е. установление таких утверждений, в которых существование эффективного разрешающего алгоритма для конкретной задачи влечет за собой существование эффективного алгоритма для многих общепризнанно трудных задач.

Выделяют следующие классы сложности:

1. Класс «Р» – полиноминальная сложность. Говорят, что машина Т решает задачу  за полиноминальное время, если существует степенной многочлен (полином), который аппроксимирует сложность массовой проблемы . Если задача решается за полиноминальное время, то она принадлежит классу р.
2. Класс «Экспоненциальной сложности». В противном случае говорят, что машина Т решает задачу  за экспоненциальное время - экспоненциальная сложность задачи.

Из «экспоненциальных» алгоритмов иногда дополнительно выделяют два подкласса

* 1. Если сложность задачи апроксимируется функцией, которая возрастает быстрее любого полинома, но медленнее любой показательной функции, то задача имеет **субэкспоненциальную** сложность.
  2. Если сложность задачи апроксимируется функцией, которая возрастает быстрее любой показательной функции , то задача имеет **гиперэкспоненциальную**  сложность.

Практическая значимость класса P.

Для того чтобы решать задачи больших размерностей мы либо увеличиваем время, либо увеличиваем производительность.

Какую задачу можно решить при заданной скорости быстродействия.

|  |  |  |  |
| --- | --- | --- | --- |
| Функция временной  сложности | Размер решаемой  Задачи на ЭВМ | Размер решаемой  Задачи на ЭВМ | Размер решаемой  Задачи на ЭВМ |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Из таблицы видно, что полиноминальные алгоритмы лучше реагируют на увеличение производительности, а любой экспоненциальный алгоритм, начиная с некоторой размерности, оказывается практически не решаемым. Это также означает, что при  не разрешается сейчас, а также не решается в обозримом будущем. В то время как, полиноминальная задача, не разрешимая в настоящий момент для значащих размерностей, в обозримом будущем может быть решена.

Поэтому, полиноминальная задача называются легкорешаемыми, а экспоненциальная задача – труднорешаемыми.

Свойства класса «P»:

1. Класс P одинаков для всех разумных алгоритмических моделей (ЧРФ, машины поста и т. д.)
2. Схема кодирования является существенной при выделении класса P. При изменении схем кодировки задача может стать экспоненциальной. Поэтому нужно обязательно указывать схему кодирования.
3. Класс P является замкнутым – можно комбинировать полиноминальные алгоритмы, используя один в качестве «подпрограммы» другого и при этом результирующий алгоритм будет полиноминальным. .

13.2. Примеры лёгкорешаемых и труднорешаемых задач.

Пусть имеется множество 

1. Поиск наибольшего/наименьшего элемента 
2. Упорядочивание элементов в порядке возрастания  . Имеется алгоритм порядка - каждый элемент нужно сравнить с каждым. 
3. Установление является ли отношение на множестве отношением эквивалентности. Задачу зададим матрицей ,где  Для того, чтобы отношение M являлось отношением эквивалентности необходимо выполнение условий:  Для решения задачи необходимо проверить  условий. Сложность задачи 
4. Проверка является ли отношение на множестве гамильтоновым. Бинарное отношение является гамильтоновым, если элементы можно так упорядочить , что выполнено соотношение . В настоящее время неизвестно полиноминального алгоритма проверки гамильтоновости произвольного отношения . Тривиальный алгоритм требует  упорядочивований .
5. Задача выполнимости КНФ. Пусть имеется формула от булевых переменных . Пусть формула  имеет конъюнктивную нормальную форму, т. е. имеет вид :



, где , если переменна присутствует в КНФ в составе элементарной дизъюнкции и , если переменной нет. Входными данными в эту задачу является матрица  с размерностью [*m×n*]. Задача состоит в чтобы определить существует ли такой набор значений для , при котором . На настоящий момент не известно полиноминального алгоритма решения этой задачи. Тривиальный алгоритм предполагает проверку всех возможных значений , т. е. требует перебора наборов значений переменных  и вычисления для каждого такого набора значения формулы . 

13.3. Класс «NP».

Определим класс **задач распознавания,** как класс задач, имеющих ответ «ДА» или «НЕТ» (т.е. ).

Для каждой задачи распознавания ***П*** можно определить соответствующую задачу удостоверения ***П’***, для которой в случае если индивидуальная задача  проблемы имеет ответ «Да», существует такое слово , что соответствующая индивидуальная задача проблемы ***П’*** с начальными данными (,) имеет ответ «Да». Слово  называется удостоверением или догадкой, т.е.

.

Для того, чтобы массовая задача ***П*** содержалась в классе требуется только, чтобы соответствующая ей задача удостоверения ***П’*** принадлежала классу ***P***.

Примеры задач удостоверения:

1. Имеет ли уравнение  для набора коэффициентов решение в целых числах.  Доказательство существования решение – предъявление этого решения: 
2. Пусть дана задача проверки гамильтоновости бинарного отношения. Если R – гамильтоново отношение, то удостоверением этого будет последовательность элементов  - конкретный набор номеров элементов для которого выполняется свойство гамильтоновсти.
3. Пусть дана задача проверки выполнимости КНФ. Если  - выполнимая формула, то удостоверением будет соответствующий выполняющий набор .

Хотя задачи 2 и 3 решается за экспоненциальное время, соответствующие им задачи удостоверения решается за полиноминальное время. Таким образом, все три задачи принадлежат классу ***NP***

Математически, класс  определяется через понятие недетерминированного алгоритма. Введем понятие недетерминированной машины Тьюринга (машина с оракулом). 

x

x

\*













пп

Угадывающий модуль (Оракул) начинает работать в момент времени , двигается влево от МТ и пишет слово догадку (по одному символу за такт) через разделитель. Дальше работает как обычная . 

 существует слово-догадка такое, что через конечное число шагов остановится с ответом «Да».

Сложность НТ: , где  - сложность работы УМ (длина слова -догадки),  - сложность решения детерминированного алгоритма.

Сложность вводится только в том случае, когда соответствующая задача распознавания имеет ответ «Да», иначе сложность не определена.

Чтобы задача принадлежала классу необходимо, чтобы сложность НТ была полиноминальной: 

Свойства класса NP:

1. существует алгоритм, который позволяет решить эту задачу со сложностью , где p – некоторый полином. Существует НТ, . Всего может быть записано , где k – мощность внешнего алфавита задачи, . Для каждой из догадок существует такой , что через шагов НТ остановится в позиции , либо вообще не остановится  проделав  шагов если мы пришли в положение  - ответ «Да», иначе переходим к новому слову. Если она не остановилась ни для одного слова, то ответ будет «Нет». . Мы должны перебрать все полиномы, чтобы создать перечисление многочленов, в рамках которого должны достигнуть каждый полином за полиноминальное время.

пп

\*

x

x

x

К -слов





KK - слов

Для одного слова К просматриваем один такт работы. Для 2 –х слов KK просматриваем два такта работы. Если слово занимает одну ячейку смотрим остановится ли машина Тьюринга за один такт Если слово занимает две ячейки смотрим остановится ли машина Тьюринга за 2 такта.

1. Класс  однозначно определяется относительно всех разумных алгоритмических моделей.
2. Класс инвариантен относительно обратной задачи:  А вот для класса  это не так:  Пример: Если проверка выполнимости КНФ решается за полиноминальное время. Для того, чтобы убедится что КНФ не выполнима полиноминальной догадки не существует.

13.4. Как соотносятся классы P и NP?

Очевидно, что  подкласс . Существует много фактов указывающие на то, что , однако строгого доказательства этого факта пока нет. Поэтому принято говорить, что  - является маловероятно.

Если , то  (дополнение класса  до ) - трудно решаемые задачи.

# 14. Труднорешаемые задачи.

14.1. Полиноминальная сводимость задач.

Проблема  **сводится** к , если существует такое отображение , при котором 

Если , где  (полиноминально вычислима), то  **полиноминально сводится ()** к 

Для полиноминальной сводимости задач справедливы следующие утверждения:

Утверждение 1: Если задача полиноминально сводится к задаче из класса , то она сама принадлежит классу . 

Доказательство:

Утверждение 2: Если

Доказательство:

Каждая из функции полиноминально вычислима => их суперпозиция также вычислима.

14.2 Класс «NPC»

Определение:

, если:

1. 
2. 

Если выполняется только второе условие, то получаем класс 

1. Для классов **** и  справедливо следующее утверждение: Если хотя бы одна задача лежит в классе , то это означает, что все  - полные задачи лежат в классе , а классы  и  совпадают. Другими словами: Если для какой-то - полной задачи существует полиноминальный алгоритм, то и для любой задачи из класса  существует полиноминально разрешающий алгоритм.





Доказательство:

2. Если хотя бы одна -полная задача не принадлежит классу , то все задачи не принадлежат классу : 

Для чего нужен класс  полных задач?

Поскольку мы считаем, что все **** задачи не принадлежат классу , то эти задачи не имеют полиноминальное решение. Поэтому возникает вопрос: является ли задача практически решаемой или нет?

Все  труднорешаемые задачи, не имеют полиноминального решения.

Рассмотрим способ доказательства того, что задача принадлежит классу .

**Теорема 1. «Метод сводимости».**

Пусть задача

Доказательство: Доказательство строится на транзитивности полиноминальной сводимости

Таким образом, чтобы доказать, что задачи сводятся нужно иметь хотя бы одну  задачу.

14.3. Основные NP-полные задачи.

14.3.1. Теорема Кука (NP-полнота задачи «ВЫП»).

**Теорема 2 . Теорема Кука**.

Задачи проверки выполнимости произвольной логической функции в конъюнктивной нормальной форме является  .

Доказательство:

Пусть «ВЫП» - идентификатор данной задачи. . Пусть  - произвольная задача из . Необходимо показать, что . Для этого множеству индивидуальных задач  с ответом «Да» поставим в соответствие недетерминированную машину Тьюринга (НТ), работающая за полиноминальное время. Другими словами, существует слово догадка (с(х)) и недетерминированная машина Тьюринга (НТ), которая за полиноминальное число шагов приходит в состояние  для всех задач имеющих ответ «ДА». 

Этот факт соответствует набору следующих выражений:

G1: В любой момент времени соответсвующая машина Тьюринга находится в одном и только одном состоянии 

G2: В любой момент времени считывающая головка машины может обозревать одну и только одну ячейку

G3: В любой момент времени ячейка в активной зоне может содержать только один символ внешнего алфавита .

G4: В начальный момент времени «0» машина находится в конфигурации 

G5:Не более чем через полиноминальное число шагов машина приходит в состояние 

G6:В каждый момент времени конфигурация машины в следующий момент времени  определена одной из команд машины НТ.

Выполнение утверждений G1-G6 означает, что задача принадлежащая классу  имеет ответ «ДА»

Теперь эти утверждение нужно представит в виде КНФ

G1:  - мощность внутреннего алфавита 

 - одновременно в двух состояниях  находится не может.

Получаем порядка  дизъюнкций для каждой из пар  Общее количество переменных будет порядка  - полиноминальная оценка задачи, где  - длина входа.

Таким образом имея задачу удостоверения мы конструируем КНФ и задача удостоверения сводится к проверке выполнимости КНФ.

14.3.2. Примеры NP-полных задач:

1. Задача «3ВЫП». Пусть  формула от булевых переменных в конъюнктивной нормальной форме, где каждая дизъюнкция имеет не более, чем три вхождения переменных. Задача проверки выполнимости таких формул называется задачей 3 – выполнимости («3ВЫП»). Задача «3ВЫП» является NP – полной  . Причем  формулу с 3-я дизъюнкциями преобразовать в задачу с 2-я дизъюнкциями за полиноминальное число шагов нельзя.
2. Задача «КЛИКА». Рассмотрим графическую задачу: по произвольному графу  и числу узнать, имеется ли в графе полный подграф с  вершинами (клика). (Граф называется полным, если любые вершины соединены ребром .

- клика мощности 3

G

Задача – имеется ли в графе G полный подграф с k вершинами (существует ли клика мощности k)?

1. Задача «вершинное покрытие» (ВП). Говорят, что некоторое множество вершни  графа образует вершинное покрытие графа, если для любого ребра  найдется инцидентная ему вершина  этого множества. 

Задача - существует ли вершинное покрытие мощности k :

1. Задача «независимые вершины» (НВ). Говорят, что множество вершин  графа  независимо, если никакие две вершины из  не связаны ребром. Задача о независимом множестве вершин заключается в том, чтобы для произвольного графа  и целого числа  выяснить, существует ли в  независимое множество из  вершин.

Задача состоит в том, что существует ли в этом графе подмножество НВ мощности k.

1. Задача «Гамильтонов цикл» (ГЦ). Цикл на графе – это замкнутый путь (замкнутая траектория) состоящая из ребер, при котором каждая из вершин не повторяется дважды. Гамильтонов цикл – это цикл, который проходит все вершины. Гамильтонов цикл – это некое упорядочивование вершин. Для произвольного графа  требуется узнать, существует ли перестановка вершин , такая, что выполнено: . Задача состоит в том, чтобы выяснить существует ли в графе гамильтонов цикл.
2. Задача о множестве ребер разрезающих цикл РЦ. Множество ребер разрезающих циклы. Для произвольного графа  и целого числа  выяснить существует ли множество , такое, что и каждый цикл графа  содержит ребро из .
3. Задача о множестве вершин разрезающих циклы (ВЦ). Для каждого цикла имеется вершина из этого множества. 
4. Задача о изоморфизме подграфу (ИП). Два графа называются изоморфным, если существует отображение одного графа на другой, при котором вершины одного графа переходят в вершины другого графа:   .

Задача: для заданных двух графов  и  выяснить, содержит ли граф  подграф, изоморфный .

1. Задача целочисленный рюкзак (ЦР). Для произвольных чисел  и K (K-большое) требуется узнать существует ли набор целых чисел , что выполнено .

Вариантом задачи является  - рюкзак, в которой требуется установить существование  - чисел  с условием .

1. Проблема разрешимости диофантовых уравнений 2-ой степени.  Вопрос: разрешимо ли в целых числах?

14.3.3. Псевдополиноминальные алгоритмы. Сильная NP-полнота.

Задача целочисленный рюкзак в виде графа.

k

…

2

3

1

0

Имеем  вершину. Соединяем их ребрами следующим образом: если , то соединяем.

Рассмотрим следующий случай если : 







2

3

1

0

5

4

Построить граф можем за  действий, где -количество входных чисел  - размерность задачи.

Число  может быть представлено в виде суммы , если в графе существует путь по ребрам из нулевой вершины в -ую. 

Если пути нет, то  не представимо в виде суммы .

Построим пут по шагам:

Шаг 0: находимся в вершине 0

Шаг 1:метим все вершины меткой «1» , которые можем достичь за один шаг ( соединены с вершиной 0 ребром)

Шаг 2: для каждой из вершин с меткой «1» находим вершины, которые могут быть достигнуты из вершин с меткой «1» за один шаг и отмечаем их меткой «2» и т. д.

Эта процедура требует порядка  действий. Если в рамках данной процедуры пометим число , то такой путь существует, иначе пути нет.

Если  не достижимо, то действий будет  - вершин и ребер. Если набор ребер упорядочен, то будет  действий, чтобы пометить все вершины.

Сложность это алгоритма  - трудоемкость построения самого графа.

Нашли ли полиноминальный алгоритм для решения  - полной задачи? Другими словами, является ли оценка  полиноминальной?















…







…

…

, где  - набор входных параметров ( и ,  - максимальный из них),  - средний размер для кодировки одного параметра.

Значения, которые мы кодируем определяются пропорционально  и зависят от основного машинного алфавита. В  ячейках для двоичной системы кодирования . . В  ячейках для десятичной системы кодирования . Оценка  не является полиноминальной. т. к. при любой кодировке кроме унарной зависимость размера задачи от входа экспоненциальная.



Такие алгоритмы выделяют в отдельный класс – псевдополиноминальные. 

**Сильная NP-полнота.**

Рассмотрим проблему . В ней выделим подпроблему , где  - произвольная функция.

Под  будем понимать такие задачи, в которых максимальный параметр ограничен 



Задача  называется сильно  полной, если существует некоторый полином для которого подзадача  является 

**Теорема 3. О сильной  полноте.**

Если , то никакая сильно  задача не имеет псевдополиноминального алгоритма решения.



Доказательство:

Пусть  - сильная  задача и существует алгоритм решения проблемы : . По определению сильной  задачи: , для которого существует подзадача  для проблемы  алгоритм разрешения имеет полиноминальную сложность , что противоречит исходному предположению «такого алгоритма не существует» (

Сильная NPC задача делает маловероятным существование псевдополиноминального алгоритма.