

Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
Амурский государственный университет
(ГОУВПО «АмГУ»)

УТВЕРЖДАЮ

Зав. кафедрой ИиУС

_____ А.В. Бушманов

«__» _____ 2007 г.

ДИСКРЕТНАЯ МАТЕМАТИКА

Учебно-методический комплекс дисциплины

для специальности

230201 – Информационные системы и технологии;

Составитель:

Семичевская Н.П.

2007 г.

*Печатается по решению
редакционно-издательского совета
факультета математики
и информатики
Амурского государственного
университета*

Дискретная математика для специальности 230201 «Информационные системы и технологии»: учебно-методический комплекс дисциплины. / Семичевская Н.П. – Благовещенск. Изд-во Амурского гос. ун-та, 2007. 158с.

©Амурский государственный университет, 2007

©Кафедра информационных и управляющих систем, 2007

ОГЛАВЛЕНИЕ

1. Выписка из государственного общеобразовательного стандарта высшего профессионального образования
2. Рабочая программа
3. График самостоятельной работы студентов
4. Методические рекомендации по проведению самостоятельной работы студентов
5. Перечень учебников, учебных пособий
6. Конспект лекций
7. Методические указания по выполнению практических работ
8. Методические указания по выполнению контрольных работ
9. Методические указания по организации межсессионного контроля знаний студентов
10. Задания к расчетно-графическим работам
11. Комплекты экзаменационных билетов
12. Тестовые задания
13. Карта кадровой обеспеченности дисциплины

**1. ВЫПИСКА ИЗ ГОСУДАРСТВЕННОГО
ОБРАЗОВАТЕЛЬНОГО СТАНДАРТА ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

Направление подготовки дипломированного специалиста
654700 – Информационные системы

Специальность

230201 – Информационные системы и технологии.

Квалификация – *инженер*

| Индекс | Наименование дисциплин и их основные разделы | Всего часов |
|---------|--|-------------|
| ЕН.Ф.01 | <i>Дискретная математика.</i> | |
| | логические исчисления, графы, теория алгоритмов, языки и грамматики, автоматы, комбинаторика; логика высказываний; логическое следование, принцип дедукции; логика предикатов; синтаксис и семантика языка логики предикатов; принцип логического программирования; аксиоматические системы, формальный вывод; метатеория формальных систем; понятие алгоритмической систем; рекурсивные функции; машины Тьюринга; алгоритмически неразрешимые проблемы; меры сложности алгоритмов; легко и трудноразрешимые задач; основы нечеткой логики; элементы алгоритмической логики. | |

2. РАБОЧАЯ ПРОГРАММА

Федеральное агентство по образованию РФ
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО «АмГУ»)

УТВЕРЖДАЮ

Проректор по УНР

_____ Е.С.Астапова

« ____ » _____ 2007 г.

РАБОЧАЯ ПРОГРАММА

по дисциплине "Дискретная математика"
для специальности 230201 «Информационные системы и технологии»
курс 1 семестр 2

Лекции 36 (час.)

Экзамен 2 семестр

Практические (семинарские) занятия 36 (час.)

Зачет __ семестр

Лабораторные занятия __ (час.)

Самостоятельная работа 30 (час.)

Всего часов 102 час.

Составитель Семичевская Наталья Петровна

Факультет Математики и информатики

Кафедра Информационных и управляющих систем

2007 г.

«Дискретная математика»

1. Цели и задачи дисциплины, ее место в учебном процессе

В современной математической науке исследования в областях, традиционно относящихся к дискретной математике, занимают все более заметное место. Это объясняется необходимостью создания и эксплуатации современных электронных вычислительных машин, средств передачи и обработки информации, автоматизированных систем управления и проектирования.

Цель преподавания дисциплины «Дискретная математика» научить студентов основам дискретной математики, где дискретность понимается как противоположность непрерывности. В настоящее время наряду с такими классическими разделами математики, как математический анализ, дифференциальные уравнения в учебных планах специальности «Информационные системы и технологии» появились разделы по математической логике, булевой алгебре, логике высказываний и предикатов, комбинаторике и теории графов.

Курс предназначен для студентов 230201 «Информационные системы и технологии», специализирующихся в области автоматизированного управления и проектирования, вычислительной техники, системного программирования, передачи информации.

В курсе «Дискретная математика» излагаются основные понятия теории графов, основы булевой алгебры, принципы построения формальных теорий: исчисления высказываний и исчисления предикатов, математической логики и теории алгоритмов, с помощью машин Тьюринга и частично рекурсивных функций уточняются понятия алгоритма и вычислимости.

Перечень дисциплин, усвоение которых студентами необходимо при изучении дисциплины: «Математический анализ», «Линейная алгебра и геометрия» (алгебра матриц), «Программирование» (Алгоритмизация, Рекурсивные функции и процедуры).

2. Содержание дисциплины

2.1. Федеральный компонент

Дисциплина «Дискретная математика» является дисциплиной, входящей в блок общематематических и естественно научных дисциплин федерального компонента для

2.2 Содержание лекций

- 36

час.

Основные понятия теории графов

Тема 1. Ориентированные, неориентированные графы, различные виды графов. Локальные характеристики графов. Части графов. (2ч.)

Тема 2. Плоские и неплоские графы. Реализация графов. Теорема о реализации. Изоморфизм графов. (2ч.)

Тема 3. Представление графов. Основные свойства графов. (2ч.)

Математическая логика

Тема 4. История логики. Основные разделы логики. Темпоральные логики. Нечеткая и модальные логики. (2ч.)

Тема 5. Аксиоматические системы. Формальный вывод. Принципы построения формальных теорий. (2ч.)

Тема 6. Исчисление высказываний. Исчисление высказываний и алгебра логики. Логика высказываний. Логическое следование, принцип дедукции. (4ч.)

Тема 7. Исчисление предикатов и теории первого порядка. Выводимость и истинность. Предваренная нормальная форма. Теории первого порядка: исчисление с равенством; исчисление частичного нестрогого порядка. Формальная арифметика. (4ч.)

Теория алгоритмов

Тема 8. Понятие алгоритма. Основные требования к алгоритмам. Блок-схемы алгоритмов. Меры сложности алгоритмов. Понятие алгоритмической системы. Легко и трудно разрешимые задачи. (4ч.)

Тема 9. Машина Тьюринга. Основные определения, представления машины Тьюринга. Примеры машин. Операции над машинами Тьюринга. Композиция машин Тьюринга. Понятие Универсальной машины Тьюринга. Тезис Тьюринга. Проблема остановки. Алгоритмически неразрешимые проблемы. (6ч.)

Тема 10. Рекурсивные функции. Прimitивно-рекурсивные функции. Прimitивно-рекурсивные операторы. Общерекурсивные и частично-рекурсивные функции. Связь рекурсивных функций с машинами Тьюринга. Тезис Черча. (6ч.)

Тема 11. Понятие сложности вычислений; эффективные алгоритмы. Основы нечеткой логики. Элементы алгоритмической логики. (2ч.)

2.3. Практические занятия

- 36час.

1. Ориентированные, неориентированные графы, различные виды графов. Локальные характеристики графов. (4ч.)

2. Части графов. Плоские графы. Реализация графов. Изоморфизм графов. (4ч.)

3. Представление графов матричное и списочное. (2ч.)

4. Логика высказываний. Логическое следование, принцип дедукции. (2ч.)

5. Исчисление высказываний. Аксиомы и правила вывода. (2ч.)

6. Основные метатеоремы исчисления высказываний. Теорема дедукции. (4ч.)

7. Исчисление предикатов и теории первого порядка. Аксиомы и правила вывода. Правила переименования свободных и связанных переменных. (4ч.)

8. Префиксная нормальная форма в исчислении предикатов. (2ч.)

9. Примеры машин Тьюринга. Описание машин T_+ , $T_{\text{коп}}$, T_{2x} , T_{++} . (4ч.)
10. Композиция машин Тьюринга. Пример машин T_{++} , T_x . Проблема остановки. (4ч.)
11. Рекурсивные функции. Примитивно-рекурсивные функции (сложение, умножение, возведение в степень, урезанное вычитание, \min , \max). (4ч.)

2.4. Самостоятельная работа студентов

- 30 ч.

В течение семестра студентами должны быть выполнены и сданы следующие промежуточные виды контрольных работ:

К.р. № 1 «Описание графов»

К.р. № 2 «Исчисление высказываний, исчисление предикатов»

К.р. № 3 «Машины Тьюринга. Протокол машины Тьюринга»

К.р. №4 «Примитивно-рекурсивные функции»

РГР№1 «Теория графов»

РГР№2 «Булева алгебра логики. Логика предикатов»

Коллоквиум

Вопросы к коллоквиуму

1. Понятие формальной системы.
2. Принципы построения формальных теорий.
3. Вывод формулы В. Доказательство формулы В.
4. Теоремы теории Т. Доказательство теорем в теории.
5. Исчисление высказываний. Принципы построения теории.
6. Формулы схемы формул исчисления высказываний.
7. Аксиомы исчисления высказываний. Аксиоматический метод.
8. Правила вывода в исчислении высказываний.
9. Теорема дедукции. Применение теоремы дедукции.
10. Применение теоремы дедукции. Метод доказательства от противного.
11. Теорема о выводимости формулы в исчислении высказываний. Пример.
12. Теоремы о тождественной истинности в исчислении высказываний. Примеры.
13. Исчисление предикатов и теории первого порядка. Принципы построения.
14. Аксиомы исчисления предикатов.
15. Правила вывода в исчислении предикатов.
16. Выводимость и истинность в исчислении предикатов.
17. Эквивалентности, выводимые в исчислении предикатов. Предваренная форма или префиксная нормальная форма ПНФ.
18. Пример теории первого порядка исчисления с равенством.
19. Теория первого порядка исчисления частичного нестроого порядка.
20. Формальная арифметика.

Вопросы к экзамену по дисциплине «Дискретная математика»

Часть I. Теория графов

1. Определение графа. Ориентированные и неориентированные графы.
2. Локальные характеристики графа.

3. Теорема Эйлера о рукопожатиях.
4. Изоморфизм графов.
5. Геометрические графы. Плоские и неплоские графы. Теорема о правильной реализации.
6. Полные графы. Части графа.
7. Матричное представление графов (ориентированных и неориентированных).

Часть II. Математическая логика.

8. Понятие формальной системы.
9. Принципы построения формальных теорий.
10. Вывод формулы В. Доказательство формулы В.
11. Теоремы теории Т.
12. Исчисление высказываний. Принципы построения теории.
13. Формулы схемы формул исчисления высказываний.
14. Аксиомы исчисления высказываний.
15. Правила вывода в исчислении высказываний.
16. Теорема дедукции. Применение теоремы дедукции.
17. Применение теоремы дедукции. Метод доказательства от противного.
18. Теорема о выводимости формулы в исчислении высказываний. Пример.
19. Теоремы о тождественной истинности в исчислении высказываний. Примеры.
20. Полнота и замкнутость теории. Теорема Геделя.
21. Исчисление предикатов и теории первого порядка. Принципы построения.
22. Аксиомы исчисления предикатов.
23. Правила вывода в исчислении предикатов.
24. Выводимость и истинность в исчислении предикатов.
25. Эквивалентности, выводимые в исчислении предикатов. Предваренная форма или префиксная нормальная форма ПНФ.
26. Пример теории первого порядка исчисления с равенством.
27. Теория первого порядка исчисления частичного нестрогого порядка. Формальная арифметика (принцип индукции).

Часть III. Теория алгоритмов.

28. Понятие алгоритма. Основные требования к алгоритмам.
29. Определение машины Тьюринга. Пример.
30. Конфигурация или полное состояние машины Тьюринга. Стандартная начальная конфигурация, Стандартная заключительная конфигурация.
31. Память машины Тьюринга, данные Машины Тьюринга, детерминированность машины Тьюринга.
32. Представления машин Тьюринга. Система команд, построение таблицы переходов, построение диаграммы переходов машин Тьюринга. Примеры.
33. Понятие функции правильно вычислимой по Тьюрингу.
34. Машина Тьюринга вычисляющая сложение (T_+).
35. Машина Тьюринга вычисляющая копирование ($T_{\text{коп}}$).
36. Машина Тьюринга - композиция машин, вычисляющая функцию $f(x)=2x$ ($T_+(T_{\text{коп}})$).
37. Машина Тьюринга с правой полулентой. Пример.
38. Вычисление предикатов на машинах Тьюринга.
39. Алгоритмически неразрешимые проблемы Проблема остановки. Тезис Тьюринга.
40. Рекурсивные функции. Примитивно-рекурсивные функции (f_+ , f_x , f_+).
41. Рекурсивные функции. Примитивно-рекурсивные функции (f_{min} , f_{max}).
42. Примитивно-рекурсивные операторы.
43. Связь рекурсивных функций с машинами Тьюринга. Тезис Черча.

Экзамен принимается у студентов получивших зачет (т.е. у студентов не имеющих семестровых задолженностей) и ставится за знание теории и умение решать практические задачи по всем разделам «Дискретной математики».

Экзаменационный билет состоит из 2-х теоретических вопросов и одной практической задачи.

Критерии оценок на экзамене: В семестре студентом должны быть выполнены работы текущего контроля (контрольные работы, РГР, самостоятельные работы), а также посещение лекций и практических занятий.

Студенту на экзамене предлагается два теоретических и один практический вопрос.

«Отлично» - студент не имеет долгов по семестровым отчетным работам, ответ на теоретический материал полный, хорошо владеет материалом и отвечает на дополнительные вопросы с пониманием, приводит примеры, (освещены два теоретических вопроса), задача решена полностью с пояснениями.

«Хорошо» - студент не имеет долгов по семестровым отчетным работам, ответ на теоретический материал неполный, хорошо отвечает на дополнительные вопросы, приводит примеры, задача решена полностью с пояснениями.

«Удовлетворительно» - студент не имеет задолженностей, ответ на теоретический материал должен быть полный (хотя бы по одному из двух теоретических вопросов), ответы на дополнительные вопросы по теоретическому экзаменационному материалу должны быть близкими к теории, задача решена, хотя бы схематически.

«Неудовлетворительно» - студент не имеет задолженностей, отвечает по билету плохо (ни на один теоретический вопрос не дал ответа), не может ответить ни на один дополнительный вопрос, задача не решена даже схематически.

3. Учебно-методические материалы по дисциплине

Основная литература

1. Москинова Г.И. Дискретная математика. Математика для менеджера в примерах и упражнениях: Уч.пособие. – М.: Логос, 2000. - 240с.
2. Асеев Г. Г. Дискретная математика [Текст]: учеб. пособ. / Г.Г. Асеев, О.М. Абрамов, Д.Э. Ситников. - Ростов н/Д: Феникс; Харьков: Торсинг, 2003. - 144 с.
3. Элементы теории алгоритмов и язык программирования С [Текст]: учеб. пособие: Рек. УМО Моск. физико-техн. ин-та / В.Я. Митницкий. - М.: Изд-во Моск. физико-техн. ин-та, 2001. - 180 с.
4. Игошин В. И. Задачи и упражнения по математической логике и теории алгоритмов [Текст]: учеб. пособ.: рек. Мин. обр. РФ / В. И. Игошин. - М.: Академия, 2005. -304 с.
5. Нефедов В.Н., Осипова В.А. Курс дискретной математики. М., 1992.
6. Яблонский С.В. Введение в дискретную математику. – 3-е изд., стереотипное. – М.: «высшая школа», 2001.- 384 с.
7. Шевелев Ю.П. Высшая математика 6. Дискретная математика. Ч.2: Теория конечных автоматов. Комбинаторика. Теория графов (для автоматизированной технологии обучения): Учебное пособие. – Томск: Том. гос. ун-т систем управления и радиоэлектроники, 1999. – 120 с.
8. Семичевская Н.П. Введение в теорию графов: практикум по дискретной математике. Уч. методич. пособие, Благовещенск, Амурский гос. ун-тет, 2002.
9. Семичевская Н.П. РГР по дискретной математике: учебно-методическое пособие (Эл.вариант) Благовещенск, Амурский гос. ун-тет, 2007.

Дополнительная литература

1. Кузнецов О.П., Адельсон-Вельский Г.М. Дискретная математика для инженера. – 2-е изд., перераб. и доп. – М.: Энергоатомиздат, 1988. - 480с.
2. Нефедов В.Н., Осипова В.А. Курс дискретной математики. М., 1992.
3. Никифоров А. Книга о логике. М.: «Гнозис», 1996.
4. Манин Ю.И. Доказуемое и недоказуемое. М.: «Советское радио», 1979.
5. Смаллиан Р. Алиса в стране смекалки. М.: Мир, 1987.
6. Гарднер М. А ну-ка, догадайся! М.: Мир, 1984.
7. Корилов А.М., Сафьянова Е.Н. Основы системного анализа и теории систем: Учебное пособие. – Томск: изд-во Том. ун-та, 1989. – 207 с.
8. Основы кибернетики. Математические основы кибернетики / Под. ред. К.А. Пупкова. – М.: Высш. школа, 1974. – 416 с.
9. Мальцев А.И. Алгоритмы и рекурсивные функции. 2-е изд. –М.:Наука, 1986.–368 с.
10. Кузин Л.Т. Основы кибернетики.: В 2 т. Т. 2. Основы кибернетических моделей. – М.: Энергия, 1979. – 584 с.
11. Риордан Дж. Введение в комбинаторный анализ. – М. ИЛ, 1963. – 288 с.
12. Рыбников К.А. Введение в комбинаторный анализ. – М.: Изд-во МГУ, 1985.–312 с.
13. Шевелев Ю.П. Высшая математика 5. Дискретная математика. Ч.1: Теория множеств. Булева алгебра (для автоматизированной технологии обучения): Учебное пособие. – Томск: Том. гос. ун-т систем управления и радиоэлектроники, 1998. – 114 с.
14. Шевелев Ю.П. Высшая математика 6. Дискретная математика. Ч.2: Теория конечных автоматов. Комбинаторика. Теория графов (для автоматизированной технологии обучения): Учебное пособие. – Томск: Том. гос. ун-т систем управления и радиоэлектроники, 1999. – 120 с.

4. УЧЕБНО-МЕТОДИЧЕСКАЯ (ТЕХНОЛОГИЧЕСКАЯ) КАРТА ДИСЦИПЛИНЫ

«Математическая логика и теория алгоритмов»

| Номер недели | Номер темы | Вопросы, изучаемые на лекции | Занятия (номера) | | Используемые нагляд. и метод. пособия | Самостоятельная работа студентов | | Форма контроля |
|--------------|------------|------------------------------|---------------------|------------|---------------------------------------|----------------------------------|------|----------------|
| | | | Практич. (семинар.) | Лаборатор. | | Содержание | часы | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | 1 | 1,2,3,4 | 1 | | | | | |
| 2 | 2 | 1,2,3 | | | Мет.1 | | 2 | к/р№1 |
| 3 | 3 | 1,2 | 2 | | Мет.1 | | | |
| 4 | 3 | 3,4,5 | | | Мет.1 | | | |
| 5 | 4 | 1,2 | 3 | | Мет.1 | | 4 | РГР№1 |
| 6 | 4 | 3,4 | | | | | 2 | к/р№2 |
| 7 | 4 | 5,6 | 4 | | | | | |
| 8 | 5 | 1,2 | | | | | | |
| 9 | 5 | 3,4,5 | 5 | | | | | |
| 10 | 6 | 1,2,3 | | | | Коллоқ. | 10 | |
| 11 | 6 | 4,5,6 | 6 | | | | | |
| 12 | 7 | 1,2,3 | | | | | 2 | к/р№3 |
| 13 | 7 | 4,5,6 | 7 | | | | | |
| 14 | 7 | 7,8,9 | | | Мет.2 | | 6 | РГР№2 |
| 15 | 8 | 1,2 | 8 | | Мет.2 | | 2 | К/р№4 |
| 16 | 8 | 3,4 | | | Мет.2 | | | |
| 17 | 9 | 5,6 | 9 | | | | | |
| 18 | 9 | 1,2,3 | | | | | 2 | Тест |

К.р. № 1 «Описание графов»

К.р. № 2 «Исчисление высказываний, исчисление предикатов»

К.р. № 3 «Машины Тьюринга. Протокол машины Тьюринга»

К.р. №4 «Примитивно-рекурсивные функции»

РГР№1 «Теория графов»

РГР№2 «Булева алгебра логики. Логика предикатов»

3. ГРАФИК САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

| Содержание | Объем в часах | Сроки и форма контроля |
|---------------------------|---------------|------------------------|
| Контрольная работа №1 | 2 | 2 неделя |
| Контрольная работа №2 | 2 | 6 неделя |
| Контрольная работа №3 | 2 | 12 неделя |
| Контрольная работа №4 | 2 | 15 неделя |
| РГР №1 | 4 | 5 неделя |
| РГР №2 | 6 | 14 неделя |
| Подготовка к коллоквиуму | 10 | 10 неделя |
| Подготовка к тестированию | 2 | 18неделя |
| Итого | 30ч | |

4. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПРОВЕДЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

По каждой практической работе студенты выполняют домашнюю работу, требования к выполнению домашних работ указаны в методических разработках к практическим работам.

В течение семестра студенты выполняют контрольные работы, расчетно-графические работы по тематике предложенной в рабочей программе. Подготовка к контрольным работам предусматривает изучение материалов лекций и демонстрацию умения решать предложенные задачи в контрольных работах.

5. ПЕРЕЧЕНЬ УЧЕБНИКОВ, УЧЕБНЫХ ПОСОБИЙ

Основная литература

1. Москинова Г.И. Дискретная математика. Математика для менеджера в примерах и упражнениях: Уч.пособие. – М.: Логос, 2000. - 240с.
2. Асеев Г. Г. Дискретная математика [Текст]: учеб. пособ. / Г.Г. Асеев, О.М. Абрамов, Д.Э. Ситников. - Ростов н/Д: Феникс; Харьков: Торсинг, 2003. - 144 с.
3. Элементы теории алгоритмов и язык программирования С [Текст]: учеб. пособие: Рек. УМО Моск. физико-техн. ин-та / В.Я. Митницкий. - М.: Изд-во Моск. физико-техн. ин-та, 2001. - 180 с.
4. Игошин В. И. Задачи и упражнения по математической логике и теории алгоритмов [Текст]: учеб. пособ.: рек. Мин. обр. РФ / В. И. Игошин. - М.: Академия, 2005. -304 с.
5. Нефедов В.Н., Осипова В.А. Курс дискретной математики. М., 1992.
6. Яблонский С.В. Введение в дискретную математику. – 3-е изд., стереотипное. – М.: «высшая школа», 2001.- 384 с.
7. Шевелев Ю.П. Высшая математика 6. Дискретная математика. Ч.2: Теория конечных автоматов. Комбинаторика. Теория графов (для автоматизированной технологии обучения): Учебное пособие. – Томск: Том. гос. ун-т систем управления и радиоэлектроники, 1999. – 120 с.
8. Семичевская Н.П. Введение в теорию графов: практикум по дискретной математике. Уч. методич. пособие, Благовещенск, Амурский гос. ун-тет, 2002.
9. Семичевская Н.П. РГР по дискретной математике: учебно-методическое пособие (Эл.вариант) Благовещенск, Амурский гос. ун-тет, 2007.

Дополнительная литература

10. Кузнецов О.П., Адельсон-Вельский Г.М. Дискретная математика для инженера. – 2-е изд., перераб. и доп. – М.: Энергоатомиздат, 1988. - 480с.
11. Нефедов В.Н., Осипова В.А. Курс дискретной математики. М., 1992.
12. Никифоров А. Книга о логике. М.: «Гнозис», 1996.
13. Манин Ю.И. Доказуемое и недоказуемое. М.: «Советское радио», 1979.
14. Смаллиан Р. Алиса в стране смекалки. М.: Мир, 1987.
15. Гарднер М. А ну-ка, догадайся! М.: Мир, 1984.
16. Кориков А.М., Сафьянова Е.Н. Основы системного анализа и теории систем: Учебное пособие. – Томск: изд-во Том. ун-та, 1989. – 207 с.
17. Основы кибернетики. Математические основы кибернетики / Под. ред. К.А. Пупкова. – М.: Высш. школа, 1974. – 416 с.
18. Мальцев А.И. Алгоритмы и рекурсивные функции. 2-е изд. –М.:Наука, 1986.–368 с.

19. Кузин Л.Т. Основы кибернетики.: В 2 т. Т. 2. Основы кибернетических моделей. – М.: Энергия, 1979. – 584 с.
20. Риордан Дж. Введение в комбинаторный анализ. – М. ИЛ, 1963. – 288 с.
21. Рыбников К.А. Введение в комбинаторный анализ. – М.: Изд-во МГУ, 1985.–312 с.
22. Шевелев Ю.П. Высшая математика 5. Дискретная математика. Ч.1: Теория множеств. Булева алгебра (для автоматизированной технологии обучения): Учебное пособие. – Томск: Том. гос. ун-т систем управления и радиоэлектроники, 1998. – 114 с.
23. Шевелев Ю.П. Высшая математика 6. Дискретная математика. Ч.2: Теория конечных автоматов. Комбинаторика. Теория графов (для автоматизированной технологии обучения): Учебное пособие. – Томск: Том. гос. ун-т систем управления и радиоэлектроники, 1999. – 120 с.

6. КОНСПЕКТ ЛЕКЦИЙ

Тематический план лекций

| № | Содержание лекций | Часы |
|---------------------------------------|--|-------------|
| Основные понятия теории графов | | |
| Тема 1 | Ориентированные, неориентированные графы, различные виды графов. Локальные характеристики графов. Части графов. | 2ч. |
| Тема 2 | Плоские и неплоские графы. Реализация графов. Теорема о реализации. Изоморфизм графов. | 2ч. |
| Тема 3 | Представление графов. Основные свойства графов | 2ч. |
| Математическая логика | | |
| Тема 4 | История логики. Основные разделы логики. Темпоральные логики. Нечеткая и модальные логики. | 2ч. |
| Тема 5 | Аксиоматические системы. Формальный вывод. Принципы построения формальных теорий. | 2ч. |
| Тема 6 | Исчисление высказываний. Исчисление высказываний и алгебра логики. Логика высказываний. Логическое следование, принцип дедукции. | 4ч. |
| Тема 7 | Исчисление предикатов и теории первого порядка. Выводимость и истинность. Предваренная нормальная форма. Теории первого порядка: исчисление с равенством; исчисление частичного нестроого порядка. Формальная арифметика. | 4ч. |
| Теория алгоритмов | | |
| Тема 8 | Понятие алгоритма. Основные требования к алгоритмам. Блок-схемы алгоритмов. Меры сложности алгоритмов. Понятие алгоритмической системы. Легко и трудно разрешимые задачи. | 4ч. |
| Тема 9 | Машина Тьюринга. Основные определения, представления машины Тьюринга. Примеры машин. Операции над машинами Тьюринга. Композиция машин Тьюринга. Понятие Универсальной машины Тьюринга. Тезис Тьюринга. Проблема остановки. Алгоритмически неразрешимые проблемы. | 6ч. |
| Тема 10 | Рекурсивные функции. Примитивно-рекурсивные функции. Примитивно-рекурсивные операторы. Общерекурсивные и частично-рекурсивные функции. Связь рекурсивных функций с машинами Тьюринга. Тезис Черча. | 6ч. |
| Тема 11 | Понятие сложности вычислений; эффективные алгоритмы. Основы нечеткой логики. Элементы алгоритмической логики. | 2ч. |
| ИТОГО | | 36ч. |

ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ ТЕОРИИ ГРАФОВ

Ориентированный граф

Опр1. *Ориентированным графом (орграфом) $G=(X,\Gamma)$ называется упорядоченная пара (X,Γ) , где X есть непустое множество объектов некоторой природы – *вершин* графа, а Γ – многозначное отображение множества X на себя.*

Поскольку многозначное отображение $\Gamma:X\rightarrow X$ полностью определяется перечислением пар вида (x,y) , $y \in \Gamma x$.

Опр2. *Ориентированным графом $G=(X,\Gamma)$ называется упорядоченная пара (X,U) , где X есть непустое множество *вершин* орграфа, а U есть множество упорядоченных пар элементов из $X, U=\{(x,y)\} \subseteq X^*X$, называемых *дугами* орграфа.*

Рис.

Пусть $u=(x,y)$ – дуга орграфа, вершина x называется началом, а вершина y – концом дуги u . Дуга u исходит из вершины x и заходит в вершину y . И в том, и в другом случае говорят, что дуга *инцидентна* соответствующей вершине.

Опр3. Дуга вида (x,x) называется *петлей*.

Опр4. Вершины соединенные дугой называются *смежными*.

Когда пара вершин x,y может быть соединена несколькими параллельными дугами вида (x,y) , полученная конструкция называется *ориентированным мультиграфом*.

Опр5. Граф, у которого существование любой дуги вида (x,y) влечет существование дуги вида (y,x) называется *симметричным*.

Опр6. Ориентированный граф называется *полным*, если каждая пара вершин в нем соединена дугой.

!Полный граф симметричен.

Неориентированный граф

Опр7. *Неориентированным графом* или просто *графом* $G=(X,U)$ называется упорядоченная пара (X,U) , где X есть множество *вершин* графа, а U есть множество неупорядоченных пар элементов из X , называемых

ребрами графа.

В общем случае у ребер не различают начало и конец, и запись (x,y) обозначает то же ребро, что и (y,x) . Иногда при обходах графа или при движении по ребрам из одной вершины в другую мы будем говорить о начале ребра и его конце, подразумевая при этом, что направление движения создает ориентацию ребра. В графе каждая пара вершин соединена не более чем одним ребром; допущение параллельных ребер приводит к конструкции, именуемой *мультиграфом*.

Локальные характеристики графа

Опр. 8-10. Пусть $x \in X$ – вершина графа $G=(X,\Gamma)$, поставим ей в соответствие три числа $\deg^+(x)$, $\deg^-(x)$ и $\deg x$:

Число дуг, заходящих в вершину x , называется *полустепенью захода* вершины x и обозначается $\deg^+(x)$.

Число дуг, исходящих из вершины x , называется *полустепенью исхода* вершины x и обозначается $\deg^-(x)$.

Общее число дуг, инцидентных вершине x , называется *степенью* вершины и обозначается $\deg x$.

$$\deg x = \deg^+(x) + \deg^-(x).$$

Опр.11 Граф $G=(X,U)$ называется *регулярным степени r* , если степени всех вершин одинаковы и равны r ($\deg x=r, \forall x \in X$).

Теорема 1. Для любого конечного графа $G=(X,\Gamma)$ ($|\Gamma|<\infty$) имеют место соотношения (1) и (2):

$$\sum_{x \in X} \deg^+(x) = \sum_{x \in X} \deg^-(x) = |\Gamma|, \quad (1)$$

$$\sum_{x \in X} \deg x = 2|\Gamma|. \quad (2)$$

Без доказательства.

Теорема 2. (теорема Эйлера о рукопожатиях) В любом конечном графе $G=(X,\Gamma)$ ($|\Gamma|<\infty$) число вершин нечетной степени четно или равно нулю.

Доказательство:

Представим X в виде объединения двух множеств X_1 и X_2 : $X=X_1 \cup X_2$. X_1 - множество вершин x , у которых $\deg x = 2k_x + 1$, $k_x \in \mathbf{Z}_+$; а X_2 - множество вершин x , у которых $\deg x = 2k_x$, $k_x \in \mathbf{Z}_+$.

Тогда, воспользовавшись соотношением (2) теоремы 1, получаем:

$$2|\Gamma| = \sum_{x \in X} \deg x = \sum_{x \in X_1} \deg x + \sum_{x \in X_2} \deg x = \sum_{x \in X_1} (2k_x + 1) + \sum_{x \in X_2} 2k_x = 2 \sum_{x \in X_1} k_x + 2 \sum_{x \in X_2} k_x + \sum_{x \in X_1} 1$$

$$= 2 \sum_{x \in X} k_x + |X_1|, \text{ получим выражение для } |X_1|.$$

$$|X_1| = 2|\Gamma| - 2 \sum_{x \in X} k_x = 2 (|\Gamma| - \sum_{x \in X} k_x).$$

Правая часть выражения кратна двум или равна нулю, значит, $|X_1|$ кратно двум или равно нулю.

Обозначим множество дуг, заходящих в вершину x как $B(x)$; т.о. $|B(x)| = \deg^+(x)$. А множество дуг, исходящих из x обозначим как $A(x)$; т.о. $|A(x)| = \deg^-(x)$.

Опр.12. Множество вершин, являющихся началами дуг из $B(x)$, обозначается через $\Gamma^{-1}x$ и называются такие вершины *предшественниками* вершины x .

Опр. 13. Множество вершин, являющихся концами дуг из $A(x)$, обозначается через Γx и называются такие вершины *потомками* вершины x . Если вершина x имеет только одного предшественника, то говорят о *предке* вершины x .

Опр 14. Вершина x , инцидентная только дугам вида (x,x) , называется *голой*.

Опр15. Вершина, не инцидентная ни одной дуге, называется *изолированной*. Deg изолированной вершины равна нулю.

Входом или началом орграфа называется вершина s , у которой $\deg^+(s) = 0$.

Выходом или конечной вершиной орграфа называется вершина t , у которой $\deg^-(t) = 0$.

Опр16. Путем $\mu[a,b]$ из вершины a в вершину b называется последовательность вершин и дуг вида $a(a,x_1)^*x_1(x_1,x_2)^*x_2(x_2,x_3)\dots x_{n-1}(x_{n-1},b)b$. Путь называется *простым*, если ни одна вершина в нем не встречается дважды.

Путь из входа s орграфа в выход t называется *s-t-путем*.

Если в орграфе вершины a и b связаны путем $\mu[a,b]$, то говорят, что вершина b достижима из вершины a или вершина a достигает вершину b .

Опр17. Орграф называется *односторонне связным*, если для любой пары вершин одна достижима из другой.

Опр18. Орграф называется *сильно связным*, если для любой пары вершин каждая из них достижима из другой.

Опр19-20. Путь называется *эйлеровым*, если он содержит все дуги графа по одному разу, и *гамильтоновым*, если он содержит все вершины по одному разу.

Опр21. Путь, начало и конец которого совпадают, т.е. $\mu[a,a]$, называется *контуром* с начальной вершиной a .

Опр22-23. Контур называется *простым*, если ни одна вершина в нем не повторяется дважды, *эйлеровым*, если он содержит все дуги графа по одному разу, и *гамильтоновым*, если он содержит все вершины графа по одному разу.

Опр24. *Длиной пути* или *контюра* называется число дуг, входящих в него.

Опр25-26. Контур длины 1 есть петля. Путь длины 0 есть *тривиальный* или *вырожденный* путь.

!Длина эйлера пути или контюра равна числу дуг в орграфе; длина гамильтонова контюра равна числу вершин в орграфе, длина гамильтонова пути на единицу меньше числа вершин.

Лекция 2

ЧАСТИ ГРАФА

Опр27-28. Орграф $H=(X^*,V)$ называется *частичным графом* графа $G=(X,U)$, если $X^*\subseteq X$ и $V\subseteq U$, и *суграфом*, если $X^*=X$ и $V\subseteq U$.

Опр 29. Орграф $H=(Y,V)$ называется *подграфом* графа $G=(X,U)$, если $Y\subseteq X$ и из того, что $(y_1,y_2)\in U$ и $y_1,y_2\in Y$, следует, что $(y_1,y_2)\in V$.

Другими словами, частичный граф есть орграф, порождаемый некоторым подмножеством дуг исходного орграфа вместе с их концами. Суграф обязательно имеет то же множество вершин, что и исходный (т.е. если порождающее множество дуг инцидентно не всем вершинам исходного графа, то частичный граф дополняется до суграфа за счет изолированных вершин). Подграф порождается некоторым подмножеством вершин исходного графа и теми дугами, оба конца которых принадлежат указанному подмножеству.

Опр30-31. *Зоной* называется сильно связный подграф орграфа. Зона, максимальная относительно включения вершин, называется *компонентой сильной связности* или *бикомпонентой*.

Опр32. Вершина p_0 есть *начальная вершина* подграфа H орграфа G с входом s , если либо p_0 совпадает с s , либо вершина p_0 имеет предшественника в G не принадлежащего H .

Опр33. Вершина q_0 называется *входной вершиной* подграфа H , если существует путь из s в q_0 , не содержащий вершин подграфа, отличных от q_0 .

Опр34. Внешний потомок вершины подграфа H называется его *выходной вершиной*.

На случай неориентированных графов переносятся почти все

введенные выше понятия. Так, две вершины *смежны*, если они соединены ребром; ребро инцидентно своим концам, степень вершины есть число инцидентных ей ребер (полустепени захода и исхода теряют свой смысл).

Говоря о входе или выходе в неориентированном графе, имеют в виду некоторые выделенные вершины; например, откуда начинается и где заканчивается обход.

Для пути и контура в неориентированном графе имеются аналоги: цепь и цикл.

Опр35-36. Цепью $\mu[a,b]$, соединяющей вершины a и b , называется последовательность вершин и ребер вида $a(a, x_1)x_1(x_1, x_2)x_2 \dots x_{n-1}(x_{n-1}, b)b$. Цепь, начало и конец которой совпадают, называется *циклом*.

Иногда термин “цепь” заменяется термином “путь”. Употребление термина “цикл” в случае ориентированных графов, не ведет к путанице, т.к. оно ограничено узкими областями (например, потоки по сетям).

Для цепей и циклов неориентированного графа аналогично тому, как это делалось для орграфов, определяются понятия *простоты*, *эйлеровости*, *гамильтоновости*, а также *длины*. Понятия односторонней и сильной связности теряют смысл, их аналогом служит понятие связности: граф называется *связным*, если каждая пара вершин связана цепью. Сохраняются понятия *частичного графа*, *суграфа* и *подграфа* с естественной заменой множества дуг на множества ребер.

Деревья

Опр37. *Деревом* называется связный неориентированный граф без циклов.

Теорема 1 устанавливает эквивалентность различных свойств дерева, каждое из которых может служить определением.

Теорема 1. Для графа $G=(X,U)$ следующие утверждения эквивалентны: G -дерево;

1. Любые две вершины в графе G соединены единственной простой цепью;
2. Граф G связан и имеет $|X|-1$ ребер;
3. Граф G не содержит циклов и имеет $|X|-1$ ребер;
4. Граф G не содержит циклов, но добавление ребра между любыми двумя несмежными вершинами приводит к появлению одного цикла;
5. Граф G связан, но утрачивает это свойство после удаления любого ребра.

Без доказательства.

Опр 38. Вершина степени 1 называется *висячей* или *листом*.

Из определения дерева и теоремы 1 вытекает, что всякое дерево имеет по крайней мере две висячие вершины.

Опр 39. Определим расстояние $d(x,y)$ между вершинами x и y в дереве как длину из x в y (число ребер).

Опр 40. Расстояние от вершины x до наиболее удаленной от нее вершины называется эксцентриситетом $e(x)$ вершины x , т.е. $e(x)=\max_y d(x,y)$.

Опр 41. Наименьший из эксцентриситетов $\min_{e(x)}$ называется *радиусом* $r(T)$ дерева T .

Опр. 42 *Центральной вершиной* дерева T называется вершина, у которой эксцентриситет равен радиусу.

Опр. 43 Центром дерева называется множество его центральных вершин.

Можно показать, что каждое дерево имеет центр, состоящий или из одной вершины, или из двух смежных вершин.

Опр. 44 *Ветвью к вершине x дерева T* называется максимальное поддерево, содержащее x в качестве висячей вершины ($\deg x = 1$). Т.о., число ветвей к вершине x равно ее степени.

Опр. 45 *Вес* вершины x есть наибольшее число ребер в ветви к вершине x .

!Вес каждой висячей вершины в дереве равен числу ребер в этом дереве.

Опр. 46-47 Вершина x называется *центроидной*, если вершина x имеет наименьший вес. Центроидом дерева называется множество его центроидных вершин.

!Можно показать, что каждое дерево имеет центроид, состоящий из одной вершины или из двух смежных вершин.

Ориентированные деревья

Опр. 48 Ориентированным корневым деревом $T(r)$ или ордеревом с корнем r называется орграф с выделенной вершиной r , который удовлетворяет следующим условиям:

- а) $T(r)$ – дерево, если не принимать во внимание ориентацию дуг;
- б) из корня r достижима любая вершина (или корень r достижим из любой вершины);
- в) в корень r не заходит ни одна дуга (или из корня r не выходит ни одна дуга).

!Любое дерево можно преобразовать в ордерев, если выбрать некоторую вершину в качестве корня и задать ориентацию дуг. Для определенности будем считать, что все дуги ориентированы от корня.

Лекция 3

Дерево-остов или каркас

Опр. 49 *Каркасом* или *остовом* неориентированного графа называется его суграф в виде дерева.

Утверждение 1

Граф G имеет каркас тогда и только тогда, когда он связан.

Алгоритм поиска дерева-остова. В самом деле, если граф связан, то выясним, имеется ли в нем ребро, удаление которого не нарушает связности. В случае, когда таких ребер нет, граф G сам есть дерево в силу свойства 6. Теоремы 1; если же такое ребро есть, то удалим его и выясним, имеется ли в полученном графе ребро, удаление которого не нарушает связности, и т.д. Когда удаление ребер без нарушения связности станет невозможным, получим дерево, множество вершин которого совпадает с множеством вершин G . Проведенное рассуждение дает алгоритм для отыскания некоторого каркаса графа.

Опр. 50 Для орграфа можно определить понятие *оркаркаса*, т.е. суграфа в виде корневого ордерова.

!В орграфе G существует оркаркас с корнем в вершине r , если все вершины графа достижимы из r .

Полные n -вершинные графы

Опр. 51 *Полным n -вершинным графом K_n* называется неориентированный граф, все вершины которого смежны между собой.

!Для K_n графа справедливо следующее утверждение связывающее равенством количество ребер и количество вершин:

$m = (n(n-1)/2)$, где m – количество ребер в графе; n – количество вершин в графе. (Для полного ориентированного графа количество дуг удваивается $m = n(n-1)$).

Для многих приложений важным является наличие в графе полных подграфов, или *клик*, достаточно большого размера, ибо к этой задаче сводится задача об отыскании максимального множества попарно несмежных вершин.

Двудольные графы

Опр.52 *Двудольным графом $G=(X_1, X_2, U)$* называется граф, вершины которого разбиты на два непересекающихся подмножества X_1 и X_2 так, что концы ребер $(x, y) \in U$ принадлежат разным подмножествам, т.е. $x \in X_1, y \in X_2$. Другими словами, граф называют двудольным, если множество его вершин можно разбить на два подмножества X_1 и X_2 , так чтобы каждое ребро соединяло вершину из X_1 с вершиной из X_2 .

Рис. Двудольный граф.

Опр.53 Двудольный граф называется полным $K_{n,m}$ если каждая вершина из X_1 соединена с каждой вершиной из X_2 .

! В двудольном полном графе количество ребер $N = n \times m$.

Опр.54 Двудольный граф $K^{1,m}$ называется *звездным*.

Свойство двудольных графов: Все циклы имеют четную длину. Заметим, что здесь нуль есть четное число, так что к двудольным графам относятся и деревья.

Изоморфизм графов. Геометрические графы. Реализуемость в \mathbf{R}_3 .

Опр. 55 Графы G_1 и G_2 называются *изоморфными*, если существует взаимнооднозначное соответствие (изоморфизм) между множествами их вершин, при котором количество ребер, соединяющих любые две вершины в одном графе, равно количеству ребер, соединяющих соответствующие вершины в другом. Взаимнооднозначное отображение $\alpha: V(G) \rightarrow V(G_1)$, сохраняющее смежность.

Для того чтобы показать изоморфизм достаточно найти взаимное соответствие, а не изоморфизм показать намного сложнее, надо показать структурное несоответствие.

Опр. 56 *Геометрическим графом* называется граф, у которого множество вершин – множество отмеченных точек в \mathbf{R}_2 или в \mathbf{R}_3 , множество дуг – множество параметризованных отрезков непрерывных кривых в \mathbf{R}_2 или в \mathbf{R}_3 , концами которых являются соответствующие им вершины графа.

! Для любого графа существует изоморфный ему геометрический граф (и в \mathbf{R}_2 , и в \mathbf{R}_3) называемый его геометрической реализацией.

Опр. 57 Геометрический граф называется *правильно реализованным* (правильным), если его дуги не имеют общих точек, отличных от вершин графа.

Опр. 58 Граф называется *плоским* (планарным), если у него существует правильная реализация в \mathbf{R}_2 .

Теорема 4. Теорема о правильной реализации. Для любого графа существует его правильная реализация в \mathbf{R}_3 .

Опишем конструкцию, позволяющую построить правильную реализацию. Возьмем в \mathbf{R}_3 произвольную прямую l . Вершинам графа $G(X, U)$ поставим в соответствие отмеченные точки на этой прямой (точки будем обозначать теми же буквами, что и вершины графа G).

Каждой дуге графа G будет соответствовать своя плоскость, проходящая через l . Если дуга $u \in U$ такова, что $u = (x, y)$, $x \neq y$, то в соответствующей плоскости построим на отрезке $[x, y]$ как на диаметре полуокружность; если $u = (z, z)$, то в соответствующей плоскости изобразим единичную окружность, касательную к l в точке z . Эта конструкция дает правильную реализацию графа в \mathbf{R}_3 .

ПРЕДСТАВЛЕНИЕ ГРАФОВ в памяти ЭВМ

Представление графов в памяти ЭВМ существенно зависит от типов структур данных, допускаемых используемым алгоритмическим языком и типом ЭВМ.

Матричное представление

Важной особенностью графов, позволяющей использовать их для моделирования систем и процессов, является возможность приписывать ребрам (дугам) и вершинам веса, в качестве которых чаще всего выступают числа или наборы чисел. Без ограничения общности можно считать, что во взвешенном графе веса приписаны только ребрам (дугам). Во взвешенном графе длина пути (контур, цепи, цикла) определяется как сумма или произведение весов входящих в него дуг. Существуют более сложные способы определения длины пути (в таких случаях это оговаривается особо). Веса дуг часто называют длинами, стоимостью, временем и т.д. в зависимости от конкретной модели.

Представление с помощью матрицы смежности – одно из самых распространенных, т. к. более удобно в описании алгоритмов на графах. Для графов с большим числом дуг это достаточно компактное представление, особенно если есть возможность работать с двоичными битами в машинном слове. К недостаткам следует отнести большой расход памяти при работе с графами, имеющими небольшое число дуг (матрица смежности при этом получается весьма разреженной), а также невозможность уменьшения трудоемкости алгоритмов в том случае, когда она пропорциональна числу дуг, а не числу вершин.

Опр. 58 Матрицей смежности $A(G)$ графа G с n вершинами называется квадратная матрица порядка n с элементами a_{ij} :

$$a_{ij} = \begin{cases} 1, & \text{если существует дуга (ребро) } (x_i, x_j); \\ 0 & \text{в противном случае.} \end{cases}$$

!Матрица смежности неориентированного графа симметрична относительно главной диагонали, поэтому достаточно хранить в памяти только ее половину.

!В случае ориентированного графа входу соответствует столбец из нулей, выходу – строка из нулей, число единиц в i -ой строке равно полустепени исхода вершины x_i ($\deg^-(x_i)$), число единиц в j -м столбце равно полустепени захода вершины x_j ($\deg^+(x_j)$).

Матрица смежности полностью определяет структуру графа (с точностью до изображения на плоскости и нумерации вершин), в том числе и взвешенного (в матрице смежности которого элемент a_{ij} полагаем равным весу дуги (x_i, x_j)).

Пусть $G_1=(X,U_1)$, $G_2=(X,U_2)$ – два графа с одним и тем же множеством вершин, A_1 и A_2 - их матрицы смежности. Тогда матрица $A_1 \otimes A_2$ (\otimes -

логическое сложение), определяет объединение графов, т.е. $G_1 \cup G_2 = (X, U_1 \cup U_2)$. Предположение о равенстве множеств вершин в графах - не ограничение, так как множество вершин графа всегда можно пополнить изолированными вершинами, соответствующими отсутствующим в графе G_1 вершинам из графа G_2 , и наоборот. При этом размерности матриц смежности будут одинаковыми, и i -ой вершине одного графа будет соответствовать i -ая вершина другого.

Пример 1.

Матрицы смежности ориентированного и неориентированного графов.

$$\begin{array}{r}
 \begin{matrix}
 0 & 1 & 1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 1 & 0 & 1 & 1 & 0 \\
 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 0 & 0 & 1 \\
 A(G_1) = 0 & 0 & 0 & 1 & 0 & 0
 \end{matrix}
 \qquad
 \begin{matrix}
 0 & 1 & 0 & 1 & 0 & 0 \\
 1 & 0 & 1 & 1 & 0 & 0 \\
 0 & 1 & 0 & 1 & 0 & 0 \\
 1 & 1 & 1 & 0 & 1 & 1 \\
 0 & 0 & 0 & 1 & 0 & 1 \\
 A(G_2) = 0 & 0 & 0 & 1 & 1 & 0
 \end{matrix}
 \end{array}$$

G1:

G2:

Представление с помощью матрицы инцидентий определяет граф однозначно (с точностью до изоморфизма), но применяется крайне редко в силу большой разреженности матрицы и практического отсутствия алгоритмов, работающих на такой структуре данных.

Опр. 59 Матрицей инцидентности $B(G)$ графа G с n вершинами и m дугами называется прямоугольная матрица размера $n \times m$ с элементами b_{ij} :

$$b_{ij} = \begin{cases} 1, & \text{если вершина } x_i, \text{ есть начало дуги } u_j; \\ -1, & \text{если вершина } x_i, \text{ есть конец дуги } u_j; \\ 0 & \text{в противном случае.} \end{cases}$$

Замечание: В определении отсутствует случай, когда в вершине x_i имеется петля. Большинство авторов используют дополнительное значение для b_{ij} (например, какой-нибудь символ “1 и или выражение -1”).

Опр. 60 Матрицей достижимости $R(G)$ графа G с n вершинами называется квадратная матрица порядка n с элементами r_{ij} :

$$r_{ij} = \begin{cases} 1, & \text{если вершина } x_j \text{ достижима из } x_i; \end{cases}$$

0 в противном случае.

Списочное представление

Представление с помощью списков смежности является главной альтернативой представлению с помощью матрицы смежности.

Опр.1 Список смежности для вершины v есть просто список вершин из множества Γv (потомки вершины v), т.е. это список концов дуг, исходящих из вершины v в случае орграфа, или список смежных с v вершин в случае графа.

Граф представляется с помощью $|X|$ списков смежности, по одному для каждой вершины. Если число дуг в орграфе существенно мало по сравнению с полным графом, то этот способ представления весьма эффективен. Списки смежности занимают объем памяти $|X| + |E|$ и легко реализуются с помощью списочных структур. Часто используются различные варианты списков смежности, например, задаются списки вершин из Γv , совместные списки из Γv и $\Gamma^{-}v$ и т.д. Менее удобен этот способ представления для задания взвешенных графов, ибо тогда возникает дополнительная задача хранить где-то веса дуг и устанавливать соответствующие связи между дугами и весами.

Пример2.

Задание графов с помощью списков смежности.

G1:

$x1: x2, x3;$
 $x2: 0;$
 $x3: x2, x4, x5;$
 $x4: x5;$
 $x5: x6;$
 $x6: x4;$

G2:

$x1: x2, x4;$
 $x2: x1, x3, x4;$
 $x3: x2, x4;$
 $x4: x1, x2, x3, x5, x6;$
 $x5: x4, x6;$
 $x6: x4, x5;$

Представление с помощью списка дуг применяется в тех случаях, когда необходимо иметь отдельную, независимую нумерацию дуг. При этом способе каждой дуге сопоставляется тройка $\langle u, x, y \rangle$, где u - дуга, x - ее начало, y - ее конец. Этот способ представления легко обобщается на случай взвешенных графов. Он более приспособлен для хранения различной информации о дугах.

Представление графов кодами

Пусть G – неориентированный граф, $A(G)$ – его матрица смежности (т.к. матрица симметрична относительно главной диагонали, то будем рассматривать только ее верхний треугольник A^*). Запишем строки A^* последовательно одну за другой и рассмотрим полученную последовательность из нулей и единиц как двоичное число. Меняя нумерацию вершин, будем для одного и того же графа получать разные двоичные числа.

Наибольшее из них определяется для графа однозначно и носит название *кода Харари*. Код Харари определяет граф однозначно, поэтому, например, задачу определения изоморфизма двух графов можно свести к сравнению соответствующих кодов Харари. Этот метод столь же неэффективен, как и другие методы установления изоморфизма двух произвольных графов. Нумерация вершин (и матрица смежности), соответствующая коду Харари, носит название канонической и используется при перечислении (генерировании) графов с заданными свойствами.

Код Прюфера может быть использован для задания деревьев, поскольку допускает реализацию ограниченного числа алгоритмов, работающих с такой структурой данных. Пусть T – дерево с множеством вершин $\{b_1, \dots, b_n\}$. Сопоставим дереву T последовательность $[a_1, \dots, a_{n-2}]$ по следующему правилу:

а) Полагаем $i=1$.

В последовательности $1, 2, \dots, n$ (*) путем просмотра слева направо ищем номер первой висячей вершины. Пусть это будет b_i .

г) Ищем, с какой вершиной смежна вершина b_i . Пусть это будет a_i . Запоминаем a_i .

В последовательности (*) вычеркиваем b_i .

Из дерева T удаляем вершину b_i .

Полагаем $i:=i+1$.

б) Если $i < n-1$, то переходим к шагу 2. Если $i = n-1$, то выдаем последовательность $[a_1, \dots, a_{n-2}]$. Это и есть код Прюфера.

Пример.3

Рис.

Для дерева на рис. Код Прюфера равен $[8, 4, 4, 4, 2, 2]$. Код Прюфера взаимно однозначно кодирует деревья. В случае корневого ордерера процедура получения кода Прюфера аналогична. Необходимо только на последнем месте указывать корневую вершину и при распаковке кода исключать эту

вершину из просмотра последовательности (*).

Пример 4.

Рис.

Код Прюфера [7,4,4,4,2,2,7].

Код Прюфера дает возможность доказать теорему Кэли о числе помеченных деревьев.

Теорема 2. (Кэли) Число помеченных n -вершинных деревьев равно n^{n-2} .

Доказательство теоремы не приводится.

!Код Прюфера является оптимальным по памяти кодированием деревьев.

Опр. 2 Кодирование называется *оптимальным*, если $\lim_{n \rightarrow \infty} (j(n)/H(n))=1$,

где $H(n) = \log |U_n|$, $|U_n|$ - мощность множества U_n . Пусть $U = \bigcup_{n=1}^{\infty} U_n$, где U_n - конечное множество. Предположим, что при каком то способе кодирования элементов из U для запоминания одного элемента из U_n используется самое большее $j(n)$ битов памяти.

Для оценки энтропии $H(n)$ для множества деревьев с n вершинами по теореме 2 имеем $H(n) = (n-2) \log n$. Отсюда следует, что кодирование деревьев кодом Прюфера является оптимальным, т.к. для хранения кода необходимо $j(n) =$

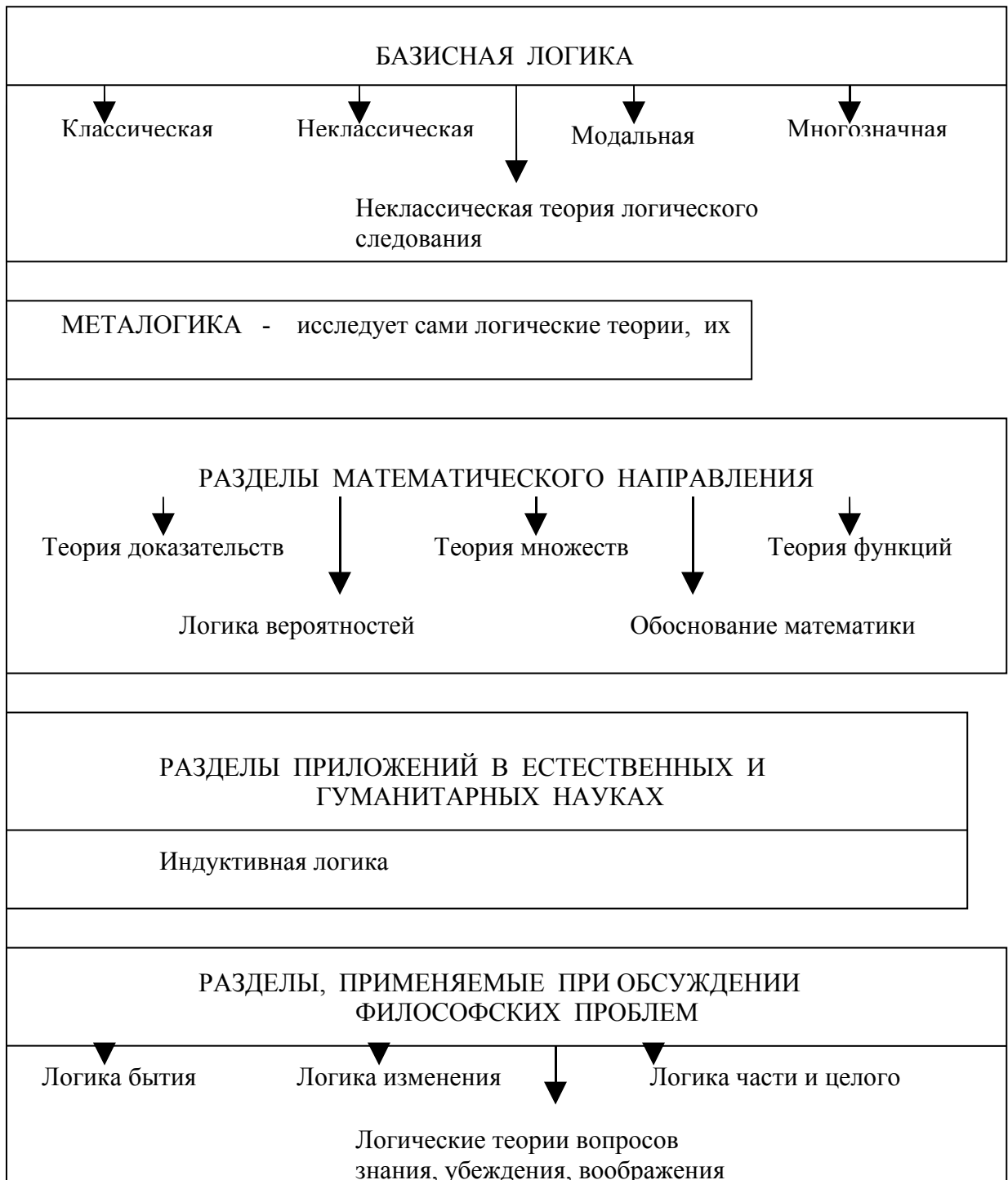
$(n-2) \log n$ битов памяти, поэтому $\lim_{n \rightarrow \infty} (j(n)/H(n))=1$.

При задании дерева списками смежности $\lim_{n \rightarrow \infty} (j(n)/H(n))=3$ для неориентированных графов и $\lim_{n \rightarrow \infty} (j(n)/H(n))=2$ для ориентированных.

Лекция 5

МАТЕМАТИЧЕСКАЯ ЛОГИКА

Основные разделы (ветви) логики



Нечеткая и модальная логики.

Модальная логика

Для классической логики вещь существует или не существует, и нет других вариантов. Язык классической логики слишком беден, чтобы на нем удалось передать рассуждения не только о реальных событиях, имеющих место в действительном мире, но и о возможных событиях или необходимых событиях. В естественном языке имеется большее число логических связей, которые не сводятся к стандартным “и”, “не”, “или”, например “возможно”, “обязательно”, “всегда”, “здесь”, “завтра”. Связки такого рода называются модальностями. Изучение модальностей, начатое Аристотелем, до начала 20-го века оставалось предметом философии и филологии. К середине века

модальная логика стала разделом математической логики. С конца 70-х г.г. эта область начала стремительно развиваться, появились глубокие математические результаты, в информатике, в лингвистике и в других разделах математической логики возникли многочисленные приложения модальных логик.

Модальная логика рассматривает понятия необходимости, возможности, случайности. Исследуются логические связи модальных высказываний, т.е. высказываний, включающих модальности. Теория логических модальностей изучает логическое поведение высказываний, включающих модальные понятия «логически необходимо», «логически возможно», «логически случайно».

Модальная логика и теория доказательств – приложение модальной логики к исследованию аксиоматических теорий и автоматическому построению доказательств. Направление было инициировано Геделем, который заметил, что доказуемость можно рассматривать как модальность.

Временная логика – изучение временных модальностей “будет”, “было”, “сейчас” и т.п.

Понятие логики

Логика (logos- слово, понятие, рассуждение разум)

Формальная логика – наука о законах и операциях правильного мышления. Согласно основному принципу логики правильность рассуждения (вывода) определяется только его логической формой или структурой, и не зависит от конкретного содержания входящих в него утверждений. Различие между формой и содержанием может быть сделанным явным с помощью особого языка, или символики, оно относительно и зависит от выбора языка.

Отличительная особенность *правильного вывода* в том, что от истинных предпосылок он всегда ведет к истинному заключению. Такой вывод позволяет из истин получать новые истины с помощью чистого рассуждения, без обращения к опыту, интуиции и т.д. *Неправильные выводы* могут от истинных посылок вести, как к истинным, так и к ложным заключениям.

Логика занимается не только связями высказываний в правильных выводах, но и

смыслом и значением выражений языка,
отношениями между терминами,
операциями определения и логического деления понятий,
вероятностными и статистическими рассуждениями,
парадоксами и логическими ошибками.

Логика не перечисляет некоторые схемы правильных рассуждений, она выявляет различные типы таких схем;
устанавливает общие критерии их правильности;

выделяет исходные схемы, из которых по определенным правилам могут быть построены другие схемы данного типа;
исследует проблему взаимной совместимости логических схем.

В современной логике логические процессы изучаются путем их отображения в языках *формализованных или логических исчислений*.

Логический синтаксис – это формальное строение логических исчислений, правила образования и преобразования входящих в них выражений.

Логическая семантика – это отношения между исчислениями и содержательными областями, служащими моделями.

Лекция 6

ФОРМАЛЬНЫЕ ТЕОРИИ ИСЧИСЛЕНИЕ ВЫСКАЗЫВАНИЙ

Формальные системы – это системы операций над объектами, понимаемыми как последовательности символов (как слова в фиксированных алфавитах); сами операции также являются операциями над символами. Термин «формальный» подчеркивает, что объекты и операции над ними рассматриваются формально, без каких бы то ни было содержательных интерпретаций символов. Предполагая, что между символами не существует никаких связей и отношений, кроме тех, которые явно описаны средствами самой формальной системы.

Принципы построения формальных теорий.

Всякая точная теория определяется: 1) языком, т.е. множеством высказываний, имеющих смысл с точки зрения этой теории; 2) совокупностью теорем - подмножеством языка, состоящим из высказываний, истинных в данной теории.

Формальная теория (или исчисление) строится следующим образом.

1. Определяется множество формул, или правильно построенных выражений, образующее язык теории. Это множество задается конструктивными средствами (индуктивным определением) и перечислимо. Обычно оно и разрешимо. Множество M разрешимо (рекурсивно), если существует алгоритм A , который по любому объекту a дает ответ, принадлежит a множеству M или нет.
2. Выделяется подмножество формул, называемых аксиомами теории. Множество может быть и бесконечным; во всяком случае, оно должно быть разрешимо.
3. Задаются правила вывода теории. Правило вывода $R(F_1, \dots, F_n, G)$ — это вычислимое отношение на множестве формул.

Опр. 1 Если формулы F_1, \dots, F_n, G находятся в отношении R , то формула G называется непосредственно выводимой из F_1, \dots, F_n по правилу R . Правило $R(F_1, \dots, F_n, G)$ записывается в виде $\frac{F_1, \dots, F_n}{G}$

Формулы F_1, \dots, F_n называются посылками правила R , G — его следствием.

Опр.2 *Выводом* формулы B из формул A_1, \dots, A_n называется последовательность формул F_1, \dots, F_m , такая, что $F_m=B$, а любая F_i ($i=1, \dots, m$) есть либо аксиома, либо одна из исходных формул A_1, \dots, A_n , либо непосредственно выводима из формул F_1, \dots, F_{i-1} (или какого-то их подмножества) по одному из правил вывода. Если существует вывод B из A_1, \dots, A_n , то говорят, что B выводима из A_1, \dots, A_n и обозначается как $A_1, \dots, A_n \vdash B$. Формулы A_1, \dots, A_n называются гипотезами или посылками вывода.

Опр.3 *Доказательством* формулы B в теории T называется вывод B из пустого множества формул, т.е. вывод, в котором в качестве исходных формул используются только аксиомы. Формула B , для которой существует доказательство, называется теоремой теории T ; факт доказуемости B обозначается $\vdash B$.

!Присоединение формул к гипотезам не нарушает выводимости. Поэтому если $\vdash B$, то $A \vdash B$, и если $A_1, \dots, A_n \vdash B$, то $A_1, \dots, A_n, A_{n+1} \vdash B$ для любых A и A_n .

Аксиоматический метод

Аксиоматический метод — способ построения научной теории, при котором какие-то положения теории избираются в качестве исходных, а все

остальные ее положения выводятся из них логическим путем, посредством доказательств.

Положения, доказываемые на основе аксиом, называются теоремами.

Аксиоматический метод зародился еще в античности и приобрел большую известность благодаря «Началам» Евклида, появившимся около 330 – 320 гг. до н.э.. Д. Гильберт (1862 - 1943) рассматривал аксиоматическую теорию как *формальную теорию*, устанавливающую соотношения между ее элементами (знаками) и описывающую любые множества объектов, удовлетворяющих ей.

Опр.1 *Аксиоматические теории* формулируются как формализованные системы, содержащие точное описание логических средств вывода теорем из аксиом. *Доказательство* в такой теории представляет собой последовательность формул, каждая из которых либо является аксиомой, либо получается из предыдущих формул последовательности по одному из принятых *правил вывода*.

К аксиоматической формальной системе предъявляются требования *непротиворечивости, полноты, независимости систем аксиом* и т.д.

Как показал известный математик и логик К. Гедель (1906 - 1978), достаточно богатые научные теории (арифметика натуральных чисел) не допускают полной аксиоматизации. Это свидетельствует об ограниченности аксиоматического метода и невозможности полной формализации научного знания.

Теорема Геделя. (Теорема о неполноте арифметики 1931г.) *Если система Z (содержащая арифметику натуральных чисел) непротиворечива, то в ней существует такое предложение A , что ни само A , ни его отрицание не могут быть доказаны средствами Z .*

Как показывает теорема Геделя даже арифметику натуральных чисел невозможно формализовать полностью, ибо в формализованной арифметике существуют истинные предложения, которые оказываются неразрешимыми. Г.т. показывает, невозможность полной формализации человеческого знания.

Полнота теории.

Опр.2 *Полнота* (в логике и дедуктивных науках) – логико-методологическое требование, предъявляемое к аксиоматической теории и характеризующее достаточность для определенных целей ее выразительных и дедуктивных средств.

Аксиоматическая теория является полной, если все ее формулы, истинные при рассматриваемой интерпретации доказуемы. Полная система содержит все возможные теоремы, не противоречащие интерпретации.

Для уточнения семантического понимания полноты может быть выдвинуто требование, чтобы либо само предложение, либо его отрицание было теоремой, т.е. чтобы предложение было или доказуемо, или опровержимо.

В 1931 г. К. Гедель показал, что достаточно богатые аксиоматические системы (включающие арифметику натуральных чисел) в принципе не могут быть полными: в них имеются предложения, которые не могут быть ни доказаны, ни опровергнуты.

Требование полноты не является необходимым; неполные аксиоматические системы могут представлять и теоретический, и практический интерес.

Непротиворечивость

Опр.3 *Непротиворечивость* – это свойство системы предложений некоторой теории (для аксиоматической теории – система ее аксиом), заключающееся в невыводимости из них противоречия.

Если отрицание какого-то предложения может быть доказано в теории, то о самом предложении говорится, что оно опровержимо в ней.

Непротиворечивость теории означает, что никакое предложение не может быть в ней и доказано, и опровергнуто.

Требование непротиворечивости является обязательным требованием к научной и, в частности, логической теории. Противоречивая теория заведомо несовершенна: наряду с истинными положениями она включает также ложные, в ней что-то одновременно доказывается, и опровергается.

Во многих теориях имеет место закон *Дунса Скота*. В этих условиях доказуемость противоречия означает, что становится «доказуемым» все что угодно и понятие доказательства теряет смысл.

Закон Дунса Скота. Ложное высказывание влечет (имплицирует) любое высказывание.

Закон Дунса Скота выражается формулой:

$\neg p \rightarrow (p \rightarrow q)$ или эквивалентной ей в классической логике формулой:

$(p \& \neg p) \rightarrow q$.

Для простых теорий, таких как исчисление высказываний, доказательство непротиворечивости не представляет труда. В более сложных теориях оно обычно сводится к интерпретации в терминах теории множеств.

Для сложных теорий, например, арифметики и самой теории множеств, отыскание подходящей теории, которая сама была бы непротиворечивой и вместе с тем могла бы использоваться для доказательства их непротиворечивости, представляется задачей скорее всего безнадежной.

Это указывает на нетривиальность проблемы непротиворечивости.

Лекция 7

Исчисление высказываний.

В исчислении высказываний мы будем иметь дело с формулами алгебры логики. Но здесь формулы рассматриваются не как способ

представления функций, а как составные высказывания, образованные из элементарных высказываний с помощью логических операций или, как говорят в логике, связок дизъюнкции, конъюнкции, отрицания, импликации. При этом особое внимание уделяется тождественно-истинным (т.-и.) высказываниям, поскольку они должны входить в любую теорию в качестве общелогических законов.

Исчисление высказываний определяется следующим образом:

1. *Алфавит* вычисления высказываний состоит из переменных высказываний (пропозиционных букв): A, B, C, \dots , знаков логических связок дизъюнкции, конъюнкции, отрицания, импликации ($\neg, \vee, \&, \rightarrow$) и скобок.

2. *Формулы*

а) переменное высказывание есть формула;

б) если \mathfrak{S} и \mathfrak{X} – формулы, то $(\mathfrak{S} \vee \mathfrak{X})$, $(A \& \mathfrak{X})$ ($\mathfrak{S} \rightarrow \mathfrak{X}$) и $\neg \mathfrak{X}$ – формулы;

с) других формул нет.

3. *Аксиомы*. Приведем две системы аксиом.

Система аксиом I.

1. $A \rightarrow (B \rightarrow A)$

2. $(A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C))$

3. $(A \& B) \rightarrow A$

4. $(A \& B) \rightarrow B$

5. $A \rightarrow (B \rightarrow (A \& B))$

6. $A \rightarrow (A \vee B)$

7. $B \rightarrow (A \vee B)$

8. $(A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow ((A \vee B) \rightarrow C))$

9. $(A \rightarrow B) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)$

10. $\neg \neg A \rightarrow A$

Система аксиом II.

1. $A \rightarrow (B \rightarrow A)$

2. $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

3. $(\neg A \rightarrow \neg B) \rightarrow ((\neg A \rightarrow B) \rightarrow A)$.

!Приведенные системы аксиом равносильны в том смысле, что порождают одно и тоже множество формул. Такое утверждение нуждается в доказательстве, которое заключается в том, что показывает выводимость всех аксиом системы II из аксиом системы I и, наоборот системы I из системы II.

Возможны и другие системы аксиом, равносильные первым двум системам.

4. *Правила вывода:*

1) правило подстановки. Если \mathfrak{S} – выводимая формула, содержащая букву A (обозначим $\mathfrak{S}(A)$), то выводима формула $\mathfrak{S}(\mathfrak{X})$, получающаяся из \mathfrak{S} заменой всех вхождений A на произвольную формулу;

2) правило заключения (Modus Ponens). Если \mathcal{S} и $\mathcal{S} \rightarrow \mathcal{K}$ – выводимые формулы, то \mathcal{K} выводима.

В этом описании исчисления высказываний аксиомы являются формулами исчисления; формулы же, использованные в правилах вывода, это «метаформулы», или *схемы формул*. Схема формул, например $\mathcal{S} \rightarrow \mathcal{K}$, обозначает множество всех тех формул исчисления, которые получаются, если ее метаварьируемые заменить формулами исчисления.

Примеры вывода в исчислении высказываний.

Основные метатеоремы исчисления высказываний.

Теорема 1 (дедукции). Если Γ (множество формул), $\mathcal{S} \vdash \mathcal{K}$, то $\Gamma \vdash \mathcal{S} \rightarrow \mathcal{K}$ (\mathcal{S}, \mathcal{K} - формулы).

Будем исходить из системы аксиом II и рассматривать их как схемы аксиом (не пользоваться правилом подстановки).

Следствие 1 (из теоремы дедукции) (правило силлогизма)

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

Построим вывод правила силлогизма:

1. $A \subset B$ - гипотеза;
2. $B \subset C$ - гипотеза;
3. A - гипотеза;
4. В применяем правило *т.р.* 1. 3.
5. C - применяем правило *т.р.* 2. 4.
6. Тогда $A \subset B, B \subset C, A \vdash C$. По теореме о дедукции $A \subset B, B \subset C \vdash A \subset C$.

Следствие 2 (из теоремы дедукции) $A \subset (B \subset C), B \vdash A \subset C$.

В результате двукратного применения правила *т.р.* получим $A, A \supset (B \supset C), B \vdash C$. Отсюда по теореме дедукции $A \supset (B \supset C), B \vdash A \supset C$.

Для любых формул A и B в исчислении высказываний верны утверждения:

Утверждение 1. $\vdash \lceil \rceil A \supset A$.

Утверждение 2. $\vdash A \supset \lceil \rceil A$.

Утверждение 3. $\vdash \lceil A \supset (A \supset B)$.

Утверждение 4. $\vdash (\lceil B \supset \lceil A) \supset (A \supset B)$.

Утверждение 5. $\vdash (A \supset B) \supset (\lceil B \supset \lceil A)$.

Утверждение 6. $\vdash A \supset (\lceil B \supset \lceil (A \supset B))$.

Утверждение 7. $\vdash (A \supset B) \supset ((\lceil A \supset B) \supset B)$.

Применение теоремы дедукции.

Пример 1. Покажем, что аксиома III выводима из системы аксиом I.

1. Подставим в I9 $\neg A$ вместо A . Получим: $(\neg A \supset B) \supset ((\neg A \supset \neg B) \supset \neg \neg A)$.
2. Двойное применение правила *m.p.* к шагу 1 дает:
 $\neg A \supset B, \neg A \supset \neg B \vdash \neg \neg A$.
3. Так как из аксиомы I 10 следует по правилу *m.p.*, что $\neg \neg A \vdash A$, то по транзитивности выводимости получим $\neg A \supset B, \neg A \supset \neg B \vdash A$.
4. Переставим гипотезы в полученной выводимости (их порядок неважен, как видно из определения выводимости): $\neg A \supset \neg B, \neg A \supset B \vdash A$.
5. Применив 2 раза к шагу 4 теорему дедукции, получим аксиому ПЗ:
 $(\neg A \supset \neg B) \supset ((\neg A \supset B) \supset A)$.

Пример 2. Очень распространенным методом математических доказательств является метод доказательства от противного: предположим, что A верно, и покажем, что 1) из A выводится B , 2) что из A выводится $\neg B$, что невозможно и, следовательно, A неверно, т.е. верно $\neg A$.

Этот метод формулируется как правило: «если $\Gamma, A \vdash B$ и $\Gamma, A \vdash \neg B$, то $\Gamma \vdash \neg A$ ». Докажем, что это правило в исчислении высказываний выполняется.

По теореме дедукции, если $\Gamma, A \vdash B$ и $\Gamma, A \vdash \neg B$, то $\Gamma \vdash A \supset B$ и $\Gamma \vdash A \supset \neg B$. Из этих импликаций и аксиомы I9 двойным применением правила заключения $\{m.p.\}$ получаем $\Gamma \vdash \neg A$. Доказанное правило называется также *правилом введения отрицания*.

Пример 3. Докажем закон исключенного третьего $\vdash A \vee \neg A$.

1. $\neg(A \vee \neg A), A \vdash A \vee \neg A$ (аксиома I6 при $B = \neg A$ и *m.p.*).
2. $\neg(A \vee \neg A), A \vdash \neg(A \vee \neg A)$ (очевидно)
3. Применяя к шагам 1 и 2 только что доказанное правило введения отрицания, получаем: $\neg(A \vee \neg A) \vdash \neg A$.
4. Аналогично доказывается $\neg(A \vee \neg A) \vdash \neg \neg A$.
5. Применяя к шагам 3 и 4 введение отрицания, получаем:
 $\vdash \neg \neg(A \vee \neg A)$.
6. С помощью аксиомы I 10 и правила *m.p.* снимаем двойное отрицание в шаге 5 и получаем $\vdash A \vee \neg A$.

Лекция 8

Исчисление высказываний и алгебра логики.

Формула F исчисления высказываний содержательно интерпретируется как составное высказывание, истинность которого зависит от истинности входящих в него элементарных высказываний. Эта зависимость в точности соответствует зависимости значения логической функции, представляемой формулой F , от значений переменных этой функции. Если задана формула $F(A_1, \dots, A_n)$ и распределение истинностей входящих в нее элементарных высказываний A_1, \dots, A_n , то для выяснения истинности ее нужно вычислить как логическую функцию на наборе $(\sigma_1, \dots, \sigma_n)$.

$$A_i^\sigma = \begin{cases} A_i, & \text{если } \sigma_i=1, \text{ т.е. если } A_i \text{ истинно;} \\ \neg A_i, & \text{если } \sigma_i=0, \text{ т.е. если } A_i \text{ ложно.} \end{cases}$$

Если $F(\sigma_1, \dots, \sigma_n)=1$, то высказывание F истинно при данном распределении истинностей A_1, \dots, A_n , если $F(\sigma_1, \dots, \sigma_n)=0$, то высказывание F ложно.

Возникает вопрос, как связано такое содержательное, “истинностное” истолкование формул с их выводимостью в исчислении высказываний?

Теорема 12. Пусть формула $\mathfrak{R}(A_1, \dots, A_n)$ определяет логическую функцию f от n переменных. Тогда, если $f(\sigma_1, \dots, \sigma_n)=\sigma$, то в вычислении высказываний $A_1^{\sigma_1}, \dots, A_n^{\sigma_n} \vdash \mathfrak{R}^\sigma$.

Теорема 13. Всякая теорема исчисления высказываний является тождественно-истинным высказыванием.

Теорема 14. Всякая тождественно-истинная формула является теоремой исчисления высказываний.

Таким образом, исчисление высказываний действительно выполняет задачу порождения общелогических законов – тождественно истинных высказываний.

Полнота и непротиворечивость исчисления высказываний.

Исчисление высказываний построено как аксиоматическая теория.

! Множество теорем исчисления высказываний совпадает с множеством т-и (тождественно-истинная) формул логики высказываний.

Теорема 1. Всякая выводимая (из пустой системы гипотез) формула исчисления высказываний т-и (тождественно-истинная).

Доказательство.

Теорема 2. Любая т-и (тождественно-истинная) формула выводима в исчислении высказываний, т.е. является теоремой.

Теорема о непротиворечивости

Исчисление высказываний непротиворечиво.

Действительно согласно теореме 1 всякая выводимая формула т.-и. Отрицание этой формулы не является т.-и. формулой. Следовательно, ни для какой формулы A невозможно, чтобы одновременно $\vdash A$ и $\vdash \neg A$.

Наряду с полнотой аксиоматической теории в широком смысле рассматривают ее полноту и в узком смысле, если добавление любой

невыводимой формулы в качестве схемы аксиом приводит к противоречивой теории.

Теорема о полноте: Исчисление высказываний полно в узком смысле.

F -произвольная невыводимая формула (согласно теореме о непротиворечивости в качестве F можно взять любую не тождественно-истинную формулу), X_1, \dots, X_n – список ее переменных, а $\langle \sigma_1, \dots, \sigma_n \rangle$ – распределение истинностей, на котором формула F принимает ложное значение.

Расширенная теория оказывается противоречивой.

Во всякой формальной теории возникает вопрос о независимости ее аксиом, т.е. вопрос о том, можно ли какую-нибудь аксиому вывести из остальных, применяя правила вывода данной системы.

! Система аксиом Π исчисления высказываний независима.

ИСЧИСЛЕНИЕ ПРЕДИКАТОВ И ТЕОРИИ ПЕРВОГО ПОРЯДКА

Предикаты.

Опр. 1 Предикатом $P(x_1, \dots, x_n)$ называется функция $P: M^n \rightarrow B$, где M — произвольное множество, а $B = \{0, 1\}$ — двоичное множество. Иначе говоря, n -местный предикат, определенный на M , — это *двузначная функция от n аргументов*, принимающих значения в произвольном множестве M . M называется *предметной областью* предиката, а x_1, \dots, x_n — *предметными переменными*. В принципе ничто не мешает определить предикат в более общем виде как функцию $P: M_1 \times M_2 \times \dots \times M_n \rightarrow B$, т.е. разрешить разным аргументам принимать значения из разных множеств. Иногда это оказывается удобным; однако, как правило, в логике предикатов исходят из первого определения.

Для любых M и n существует взаимно однозначное соответствие между n -местными отношениями и n -местными предикатами на M : а) каждому n -местному отношению R соответствует предикат P , такой, что $P(a_1, \dots, a_n) = 1$, если и только если $(a_1, \dots, a_n) \in R$; б) всякий предикат $P(x_1, \dots, x_n)$ определяет отношение R , такое, что $(a_1, \dots, a_n) \in R$, если и только если $P(a_1, \dots, a_n) = 1$. При этом R задает *область истинности* предиката P .

Всякой функции $f: M^n \rightarrow M$ можно поставить в соответствие $(n+1)$ -местный предикат P , такой, что $P(a_1, \dots, a_n, a_{n+1}) = 1$, если и только если $f(a_1, \dots, a_n) = a_{n+1}$. Поскольку функция должна быть однозначной, то это соответствие требует, чтобы для любого $a_{n+1}^* \neq a_{n+1}$ $P(a_1, \dots, a_n, a_{n+1}^*) = 0$. Поэтому обратное соответствие [от $(n+1)$ -местного предиката к n -местной функции] возможно не всегда, а только при выполнении указанного условия.

Будем употреблять одинаковые обозначения для отношений и соответствующих им предикатов; при этом, помимо функциональных обозначений вида $P(x)$, $P(x_1, x_2)$, для двухместных предикатов будем пользоваться обозначениями вида $x_1 P x_2$, которые уже употреблялись для бинарных отношений.

Пример 1.

а. Предикат $x_1 > x_2$ — это двухместный предикат, предметной областью которого могут служить любые множества действительных чисел. Высказывание $6 > 5$ истинно, а высказывания $7 > 7$ и $3 > 10$ ложны. Различные подстановки чисел вместо одной предметной переменной дают различные одноместные предикаты: $x_1 > 5$, $x_1 > 0$, $7 > x_1$ и т. д.

б. Великая теорема Ферма, не доказанная до сих пор. утверждает, что для любого целого $n > 2$ не существует натуральных чисел x , y , z , удовлетворяющих равенству $x^n + y^n = z^n$. Если этому равенству поставить в соответствие предикат $P_F(x, y, z, n)$, истинный тогда и только тогда, когда оно выполняется, а через $N(x)$ обозначить предикат « x — натуральное число», то теорема Ферма равносильна утверждению «выражение $N(x)$

$\&N(y)\&N(z)\&N(n)\&(n>2) \rightarrow P_F(x, y, z, n)$ верно для любых чисел x, y, z, n ».

в. В описаниях вычислительных процедур и, в частности, в языках программирования часто встречаются указания типа «повторять цикл до тех пор, пока переменные x и y не станут равными, либо прекратить вычисление цикла после 100 повторений». Если обозначить через i счетчик повторений, то описанное здесь условие описывается выражением $(x=y)\vee(i >100)$, а указание в целом принимает вид: «повторять, если $\neg((x=y)\vee(i >100))$ ».

Кванторы.

Пусть $P(x)$ — предикат, определенный на M . Высказывание «для всех x из M , $P(x)$ истинно» обозначается $\forall xP(x)$ (множество M не входит в обозначение и должно быть ясно из контекста).

Опр. 2,3 Знак $\forall x$ называется *квантором общности*; другое его обозначение (x) . Высказывание «существует такой x из M , что $P(x)$ истинно» обозначается $\exists xP(x)$. Знак \exists называется *квантором существования*; другое его обозначение (Ex) .

Опр. 4 Переход от $P(x)$ к $\forall xP(x)$ или к $\exists xP(x)$ называется связыванием переменной x , а также навешиванием квантора на переменную x (или на предикат P), иногда – квантификацией переменной x .

Опр. 5, 6 Переменная, на которую навешен квантор, называется *связанной*; несвязанная переменная называется *свободной*.

Смысл связанных и свободных переменных в предикатных выражениях различен. Свободная переменная — это обычная переменная, которая может принимать различные значения из M ; выражение $P(x)$ — переменное высказывание, зависящее от значения x . Выражение $\forall xP(x)$ не зависит от переменной x и при фиксированных P и M имеет вполне определенное значение. Это, в частности, означает, что переименование связанной переменной, т. е. переход от $\forall xP(x)$ к $\forall yP(y)$, не меняет истинности выражения. Переменные, являющиеся по существу связанными, встречаются не только в логике. Например, в выражениях $\sum_{x=1}^{100} f(x)$ или $\int_a^b f(x)dx$ переменная x связана; при фиксированной f первое выражение равно определенному числу, а второе становится функцией от a и b .

Навешивать кванторы можно и на многоместные предикаты, и вообще на любые логические выражения, которые при этом заключаются в скобки. Выражение, на которое навешивается квантор $\forall x$ или $\exists x$, называется *областью действия квантора*; все вхождения переменной x в это выражение являются связанными. Навешивание квантора на многоместный предикат уменьшает в нем число свободных переменных и превращает его в предикат от меньшего числа переменных.

Пример 2.

а. Пусть $P(x)$ — предикат « x —четное число». Тогда высказывание $\forall xP(x)$ истинно на любом множестве четных чисел и ложно, если M содержит

хотя бы одно нечетное число; высказывание $\exists xP(x)$ истинно на любом множестве, содержащем хотя бы одно четное число, и ложно на любом множестве нечетных чисел.

б. Теорема Ферма (см. пример 1) формулируется следующим образом: $\forall x\forall y\forall z\forall n (N(x) \& N(y) \& N(z) \& N(n) \& (n>2) \rightarrow \neg P_F(x, y, z, n))$.

в. Рассмотрим двухместный предикат $x \geq y$ на множествах M с отношением нестрогого порядка и различные квантификации его переменных. $\forall x(x \geq y)$ — одноместный предикат от y ; если M — множество неотрицательных чисел, то этот предикат истинен в единственной точке: $y=0$. $\forall x\forall y(x \geq y)$ — высказывание, истинное на множестве, состоящем из одного элемента, и ложное на любом другом множестве. $\exists x\exists y(x \geq y)$ истинно на любом непустом множестве. Высказывание $\exists x\forall y(x \geq y)$ («существует x , такой, что для любого y $x \geq y$ ») утверждает, что в M имеется единственный максимальный элемент. Оно истинно на любом конечном множестве целых чисел, но ложно на множестве $\{1/2, 2/3, 3/4, \dots, n/(n+1)\dots\}$ или на множестве двоичных векторов, из которого удален вектор, состоящий из одних единиц. Высказывание $\forall y\exists x(x \geq y)$ утверждает, что для любого элемента y существует элемент x не меньший, чем y ; оно истинно на любом непустом множестве ввиду рефлексивности отношения \geq .

Последние два высказывания говорят о том, что *перестановка кванторов существования и общности меняет смысл высказывания и условия его истинности*.

Истинные формулы и эквивалентные соотношения логики предикатов.

При логической (истинностной) интерпретации формул логики предикатов возможны три основные ситуации.

Опр. 7 Если в области M для формулы F существует такая подстановка констант вместо всех переменных, что F становится истинным высказыванием, то формула F называется *выполнимой* в области M . Если существует область M , где F выполнима, то F называется просто *выполнимой*.

Пример выполнимой формулы: $\exists xP(x,y) \rightarrow \forall xP(x,y)$.

Опр. 8 Если формула F выполнима в M при любых подстановках констант, то она называется *тождественно истинной* в M .

Опр. 9 Если формула F выполнима в M при любых подстановках констант, то она называется *тождественно истинной* в M .

Формула, тождественно истинная в любых M , называется *тождественно истинной или общезначимой*.

Например, формула $\exists xP(x,y) \rightarrow \forall xP(x,y)$ тождественно истинна для всех M , состоящих из одного элемента, а формула $\forall x(P(x) \vee \neg P(x))$ тождественно истинна.

Опр. 10 Если формула F невыполнима в M , она называется *тождественно ложной* в M .

Если F невыполнима ни в каких M , она называется *тождественно ложной*, или *противоречивой*.

Например, формула $P_1(x,y) \& \neg P_1(x,z) \& P_2(x) \& \neg P_2(y) \& \neg P_2(z)$ тождественно ложна на любой области M , если $|M| \leq 2$ (предлагаем это проверить). Формула $\exists x (P(x) \& \neg P(x))$ тождественно ложна.

Опр. 11 Формулы называются *эквивалентными*, если при любых подстановках констант они принимают одинаковые значения.

В частности, *все тождественно истинные формулы (и все ложные формулы) эквивалентны*.

Отметим, что если F_1 и F_2 эквивалентны в соответствии с этим определением, то формула $F_1 \sim F_2$ тождественно истинна.

Множество истинных формул логики предикатов входит в любую теорию, и, следовательно, его исследование является важнейшей целью логики предикатов. В частности, как следует из сделанного ранее замечания, в нем содержатся все эквивалентные соотношения логики предикатов. В этом исследовании прежде всего возникают две проблемы: получение истинных формул и проверка формулы на истинность. Если вспомнить классификацию способов задания множеств, то первая проблема — это проблема построения порождающей процедуры, а вторая — проблема разрешающей процедуры для множества истинных формул. Те же проблемы встают и в логике высказываний. Однако там есть стандартная разрешающая процедура: вычисление формул на наборах значений переменных. С ее помощью порождающую процедуру для множества M_m тождественно истинных высказываний можно организовать следующим образом: строим последовательно все формулы, вычисляем каждую из них на всех наборах и включаем в M_m только те, которые истинны на всех наборах. Аналогичная процедура в логике предикатов сталкивается с большими трудностями, связанными с тем, что предметные и предикатные переменные имеют в общем случае бесконечные области определения. Поэтому прямой перебор всех значений невозможен, и приходится использовать различные косвенные приемы.

Покажем их на примере некоторых эквивалентных соотношений

$$\neg(\exists x P(x)) \sim \forall x \neg P(x). \quad (1)$$

Пусть для некоторого предиката P и области M левая часть истинна. Тогда не существует $a \in M$, для которого $P(a)$ истинно; следовательно, для всех $a \in M$ $P(a)$ ложно, т.е. $\neg P(a)$ истинно, и правая часть истинна. Если же левая часть ложна, то существует $a \in M$, для которого $P(a)$ истинно, и, следовательно, правая часть ложна. Аналогично доказывается

$$\neg(\forall x P(x)) \sim \exists x \neg P(x). \quad (2)$$

Докажем теперь дистрибутивность $\forall x$ относительно конъюнкции и $\exists x$ относительно дизъюнкции

$$\forall x(P_1(x) \& P_2(x)) \sim \forall xP_1(x) \& \forall xP_2(x). \quad (3)$$

Пусть левая часть соотношения истинна для некоторых P_1 и P_2 . Тогда для любого $a \in M$ истинно $P_1(a) \& P_2(a)$, поэтому $P_1(a)$ и $P_2(a)$ одновременно истинны для любых a , и, следовательно, $\forall x P_1(x) \& \forall x P_2(x)$ истинно. Если же левая часть ложна, то для некоторого $a \in M$ ложно либо $P_1(a)$, либо $P_2(a)$, а следовательно, ложно либо $\forall x P_1(x)$, либо $\forall x P_2(x)$, и правая часть ложна. Аналогично доказывается

$$\exists x(P_1(x) \vee P_2(x)) \sim \exists xP_1(x) \vee \exists xP_2(x). \quad (4)$$

Если же $\forall x$ и $\exists x$ в этих соотношениях поменять местами, то получатся соотношения, верные лишь в одну сторону:

$$\exists x(P_1(x) \& P_2(x)) \rightarrow \exists xP_1(x) \& \exists xP_2(x); \quad (5)$$

$$(\forall xP_1(x) \vee \forall xP_2(x)) \rightarrow \forall x (P_1(x) \vee P_2(x)). \quad (6)$$

В таких случаях говорят, что левая часть — более сильное утверждение, чем правая, поскольку она требует для своей истинности выполнения более жестких условий, чем правая. Так, в (5) в левой части требуется, чтобы $P_1(a)$ и $P_2(a)$ были истинны для одного и того же a , тогда как в правой части P_1 и P_2 могут быть истинны при различных a_1 и a_2 . В (6) левая часть требует, чтобы хотя бы один предикат выполнялся для всех $a \in M$; в правой части достаточно, чтобы один предикат был истинен там, где ложен другой. В этих рассуждениях по существу уже содержатся доказательства; окончательное их уточнение предоставляем читателю. Пример, когда (5) и (6) в обратную сторону неверны: $P_1(x)$: « x —четное число», $P_2(x)$: « x —нечетное число».

Приведем без доказательства еще несколько соотношений:

$$\forall x \forall y P(x, y) \sim \forall y \forall x P(x, y) \quad (7)$$

$$\exists x \exists y P(x, y) \sim \exists y \exists x P(x, y) \quad (8)$$

Пример 2, в показывает, что перестановка различных кванторов не является эквивалентностью.

Пусть Y — переменное высказывание или формула, не содержащая x . Тогда

$$\forall x(P(x) \& Y) \sim \forall xP(x) \& Y; \quad (9)$$

$$\forall x(P(x) \vee Y) \sim \forall xP(x) \vee Y; \quad (10)$$

$$\exists x(P(x) \& Y) \sim \exists xP(x) \& Y; \quad (11)$$

$$\exists x(P(x) \vee Y) \sim \exists xP(x) \vee Y; \quad (12)$$

Эти соотношения означают, что формулу, не содержащую x , можно выносить за область действия квантора, связывающего x .

О методах доказательства в логике предикатов.

Метод доказательства формул, содержащих переменные, путем

непосредственной подстановки в них констант называется *методом интерпретаций* или *методом моделей*. Подстановка констант позволяет интерпретировать формулу как осмысленное утверждение об элементах конкретного множества M . Поэтому такой метод, выясняющий истинность формул путем обращения к ее возможному смыслу, называется также семантическим (т. е. смысловым). Метод интерпретаций удобен для доказательства выполнимости формул или их неэквивалентности, поскольку и в том, и в другом случае достаточно найти одну подходящую подстановку. Он удобен также для исследования истинности формул на конечных областях. Дело в том, что если область M конечна, $M = \{a_1, \dots, a_n\}$, то кванторы переходят в конечные формулы логики высказываний:

$$\forall x P(x) \sim P(a_1) \& P(a_2) \& \dots \& P(a_n); \quad (13)$$

$$\exists x P(x) \sim P(a_1) \vee P(a_2) \vee \dots \vee P(a_n) \quad (14)$$

Заменяя все кванторы в формуле по этим соотношениям, любую формулу логики предикатов можно перевести в формулу, состоящую из предикатов, соединенных знаками логических операций. Истинность такой формулы на конечной области проверяется конечным числом подстановок и вычислений.

Для бесконечных же областей в общем случае, и прежде всего при доказательстве тождественной истинности формул метод интерпретаций, как уже отмечалось, связан с большими трудностями. Поэтому для построения множества истинных формул в логике предикатов выбирается другой путь. Это множество порождается из исходных формул (аксиом) с помощью формальных процедур — правил вывода. Слово «формальный» (которое часто противопоставляется слову «содержательный») подчеркивает здесь то обстоятельство, что при переходе от одних выводимых формул к другим не происходит какого-либо обращения к содержанию, смыслу формул. Используются лишь формальные, внешние свойства последовательностей символов, образующих формулы, причем эти свойства полностью описываются правилами вывода.

Лекция 10

Аксиомы и правила вывода

1. А л ф а в и т исчисления предикатов состоит из предметных переменных x_1, x_2, \dots , предметных констант a_1, a_2, \dots , предикатных букв $P^1_1, P^1_2, \dots, P^1_k, \dots$ и функциональных букв $f^1_1, f^1_2, \dots, f^1_k, \dots$, а также знаков логических связок $\neg, \vee, \&, \rightarrow$, кванторов общности и существования (\forall, \exists) и скобок $()$.

Опр.1 *n*-местный предикат – это функция $P(x_1, \dots, x_n)$ от n переменных, принимающих значения из некоторых заданных предметных областей, так,

что x_1 из M_1 , x_2 из M_2 ..., x_n из M_n , а функция P принимает логические значения – «истинно» или «ложно». Предикат $P(x_1, \dots, x_n)$ является функцией типа $P: M_1 \times M_2 \times \dots \times M_n \rightarrow B$, где множества M_1, M_2, \dots, M_n называются предметными областями предиката; x_1, \dots, x_n – предметными переменными предиката; B – бинарное множество $\{И, Л\}$ или $\{0, 1\}$.

Квантор собственные связи логики предикатов.

Верхние индексы предикатных и функциональных букв указывают число аргументов, их нижние индексы служат для обычной нумерации букв.

Переменные высказывания в исчислении предикатов вводятся либо непосредственно как пропозиционные буквы A_1, A_2, \dots , либо как 0-местные предикаты P^0_1, P^0_2, \dots , т.е. как предикаты без переменных.

2. Ф о р м у л ы. Понятие формулы определяется в два этапа.

1) Термы:

а) предметные переменные и константы являются термами;

б) если f^n – функциональная буква, а t_1, \dots, t_n – термы, то $f^n(t_1, \dots, t_n)$ – терм.

2) Формулы:

а) если P^n – предикатная буква, а t_1, \dots, t_n – термы, то $P^n(t_1, \dots, t_n)$ – формула; все вхождения предметных переменных в формулу вида называют свободными переменными;

б) если F_1, F_2 – формулы, то формулами являются $\neg F_1, (F_1 \rightarrow F_2), (F_1 \& F_2), (F_1 \vee F_2)$; все вхождения переменных, свободные в F_1, F_2 , являются свободными и в указанных четырех видах формул;

с) если $F(x)$ – формула, содержащая свободные вхождения переменной x , то $\forall x F(x)$ и $\exists x F(x)$ – формулы; в этих формулах все вхождения переменной x называются связанными; вхождения остальных переменных в F остаются свободными.

3. А к с и о м ы и с ч и с л е н и я п р е д и к а т о в делятся на две группы:

1) аксиомы исчисления высказываний (любая из систем I или II);

2) предикатные аксиомы:

P1. $\forall x F(x) \rightarrow F(y)$

P2. $F(y) \rightarrow \exists x F(x)$

4. П р а в и л а в ы в о д а

1) правило заключения (Modus Ponens)

3) правило обобщения (\forall -введения):

$$\frac{F \rightarrow G(x)}{F \rightarrow \forall x G(x)}$$

4) правило \exists -введения:

$$\frac{G(x) \rightarrow F}{\exists x G(x) \rightarrow F}$$

Нарушение этих требований могут привести к ложным выводам из истинных высказываний.

Пусть, например, $P(x)$ предикат « x – делится на 6», $Q(x)$ предикат « x – делится на 3». Высказывание $P(x) \rightarrow Q(x)$, очевидно, истинно для любого x , однако применение к нему правила обобщения дает высказывание $P(x) \rightarrow \forall x Q(x)$, не являющееся всегда истинным.

Если же к $P(x) \rightarrow Q(x)$ применить правило \exists -введения, получим $\exists x P(x) \rightarrow Q(x)$, из которого путем применения правила обобщения получим высказывание $\exists x P(x) \rightarrow \forall x Q(x)$, ложное на множестве натуральных чисел.

Возможны и другие системы аксиом и правил. Принцип минимизации числа логических операторов в исчислении высказываний оставляет лишь связки \neg и \rightarrow (что отражено в системе аксиом II). В исчислении предикатов этот принцип выражается в том, что квантор $\exists x$ не считается самостоятельным символом, а рассматривается как сокращение выражения $\neg \forall x \neg$.

Примеры вывода в исчислении предикатов.

Правило переименования свободных переменных
Правило переименования связных переменных

Пример 1.

Правило переименования свободных переменных.

В исчислении предикатов из выводимости формулы $F(x)$, содержащей свободные вхождения x , ни одно из которых не находится в области действия квантора по y , следует выводимость $F(y)$.

1. $\neg F(x)$ (по условию)
2. $F(x) \rightarrow (G \rightarrow F(x))$ (акс. II 1; в качестве G выбираем любую доказуемую формулу, не содержащую свободных вхождений x : ее доказуемость понадобится на шаге 5, а ограничение на x на шаге 4)
3. $G \rightarrow F(x)$ (м.р. к шагам 1 и 2)
4. $G \rightarrow \forall x F(x)$ (правило обобщения к шагу 3)
5. $\forall x F(x)$ (м.р. к G и шагу 4)
6. $F(y)$ (м.р. к шагу 5 и аксиоме P1)

Пример 2.

Правило переименования связанных переменных.

В исчислении предикатов из выводимости формулы $\forall xF(x)$, следует выводимость $\forall yF(y)$ при условии, что $F(x)$ не содержит свободных вхождений y и содержит свободные вхождения x , ни одно из которых не входит в область действия квантора по y .

1. $\vdash \forall x F(x)$ (по условию)
2. $\forall xF(x) \rightarrow F(y)$ (акс. P1)
3. $\forall xF(x) \rightarrow \forall yF(y)$ (правило обобщения к шагу 2)
4. $\forall yF(y)$ (т.р. к шагам 1 и 3).

Выводимость и истинность.

Эквивалентные преобразования в исчислении предикатов.

Теорема 5. Всякая доказуемая формула исчисления предикатов тождественно-истинна (общезначима).

Непосредственно проверяется общезначимость аксиом и показывается, что правила вывода сохраняют общезначимость, т.е. их применение к общезначимым формулам дает общезначимые формулы.

Справедлива и обратная теорема.

Теорема 6. Всякая общезначимая предикатная формула доказуема в исчислении предикатов.

! Всякому эквивалентному соотношению $F=G$ в булевой алгебре соответствует доказуемая эквивалентность $\vdash F \sim G$ в исчислении высказываний. Из теорем 5, 6 следует, что между соотношениями содержательной логики предикатов и формальными эквивалентностями в исчислении предикатов имеется аналогичное соответствие ($F \sim G$ рассматривается как сокращение $(F \rightarrow G) \& (G \rightarrow F)$). На нем имеет смысл остановиться подробнее.

Доказательства общезначимости в логике предикатов существенно сложнее, чем в логике высказываний, и поэтому формальный вывод эквивалентностей становится важным способом их получения.

Теорема 7. Пусть $F(A)$ – формула, в которой выделено вхождение формулы A ; $F(B)$ – формула, полученная из $F(A)$ заменой вхождения A формулой B . Тогда если $\vdash A \sim B$, то $\vdash F(A) \sim F(B)$.

Эта теорема формулируется как правило для исчисления предикатов, аналогичное правилу замены эквивалентных подформул в алгебрах. Благодаря этому правилу можно получать доказуемые эквивалентности в исчислении, не строя непосредственного вывода.

При доказательстве теоремы используются следующие производные правила:

- 1) если $\vdash A \sim B$, то $\vdash A \rightarrow C \sim B \rightarrow C$ и $C \rightarrow A \sim C \rightarrow B$;
- 2) если $\vdash A \sim B$, то $\vdash A \vee C \sim B \vee C$ и $\vdash C \vee A \sim C \vee B$;
- 3) если $\vdash A \sim B$, то $\vdash A \& C \sim B \& C$ и $\vdash C \& A \sim C \& B$;
- 4) если $\vdash A \sim B$, то $\vdash \neg A \sim \neg B$;
- 5) если $\vdash F(x) \sim G(x)$, то $\vdash \forall x F(x) \sim \forall x G(x)$;
- 6) если $\vdash F(x) \sim G(x)$, то $\vdash \exists x F(x) \sim \exists x G(x)$.

Первые четыре правила проверяются таблицами истинности.

С помощью этих шести правил теорема 7 доказывается индукцией по построению формулы $F(A)$: показывается, что если $A \sim B$, то эта эквивалентность сохраняется на всех шагах построения $F(A)$ из A и $F(B)$ из B . Продемонстрируем доказательство на примере.

Пример1.

Пусть $F(A) = \forall y (P_1(y) \vee \exists x P_2(x, y))$, $A = \exists x P_2(x, y)$. В качестве эквивалентности $A \sim B$ возьмем эквивалентность $\exists x P_2(x, y) \sim \forall x \neg \neg P_2(x, y)$, верную в логике предикатов и, следовательно, в силу теоремы 7 доказуемую в исчислении предикатов.

1. $\exists x P_2(x, y) \sim \forall x \neg \neg P_2(x, y)$ (исходная эквивалентность)
 2. $(P_1(y) \vee \exists x P_2(x, y)) \sim (P_1(y) \vee \forall x \neg \neg P_2(x, y))$ (правило 2)
 3. $\forall y (P_1(y) \vee \exists x P_2(x, y)) \sim \forall y (P_1(y) \vee \forall x \neg \neg P_2(x, y))$ (правило 5)
- Формула из шага 3 и есть искомая эквивалентность $F(A) \sim F(B)$.

Эквивалентности, выводимые в исчислении предикатов

$$A \& \forall x F(x) \sim \forall x (A \& F(x)); \quad (1)$$

$$A \vee \exists x F(x) \sim \exists x (A \vee F(x)); \quad (2)$$

$$A \& \exists x F(x) \sim \exists x (A \& F(x)); \quad (3)$$

$$A \vee \forall x F(x) \sim \forall x (A \vee F(x)); \quad (4)$$

$$A \rightarrow \forall x F(x) \sim \forall x (A \rightarrow F(x)); \quad (5)$$

$$A \rightarrow \exists x F(x) \sim \exists x (A \rightarrow F(x)); \quad (6)$$

$$\forall x F(x) \rightarrow B \sim \exists x (F(x) \rightarrow B); \quad (7)$$

$$\exists x F(x) \rightarrow B \sim \forall x (F(x) \rightarrow B). \quad (8)$$

Эквивалентности позволяют выносить кванторы вперед. Используя при этом соотношения, полученные в алгебре логики предикатов ($\neg(\exists x P(x)) \sim \forall x(\neg P(x))$), ($\neg(\forall x P(x)) \sim \exists x(\neg P(x))$), позволяющие заменять один квантор другим и

«спускать» отрицание внутрь области действия квантора, а также правила переименования переменных, кванторы можно вынести вперед для любой формулы.

Префиксная нормальная форма

Опр.1 Формула, имеющая вид $Q_1x_1Q_2x_2\dots Q_nx_nF$, где $Q_1, Q_2 \dots Q_n$ – кванторы, а F – формула, не имеющая кванторов (и являющаяся областью действия всех n кванторов), называется *предваренной формой*, или формулой в предваренной форме.

В исчислении предикатов для любой формулы F существует эквивалентная ей предваренная форма F^* , т.е. $\vdash F \sim F^*$.

Этапы приведения к ПНФ:

Пример 2.

Теории первого порядка (прикладные исчисления предикатов).

Построенное ранее исчисление предикатов называется исчислением предикатов первого порядка. В исчислениях второго порядка возможны кванторы по предикатам, т.е. выражения вида $\forall P(P(x))$. Приложения таких исчислений встречаются гораздо реже, нами рассматриваться не будут.

Исчисление предикатов, не содержащее функциональных букв и предметных констант, называется чистым исчислением предикатов. До сих пор рассматривалось именно чистое исчисление предикатов, хотя язык исчисления (формулы) был определен с учетом его использования в прикладных исчислениях.

Прикладные исчисления (теории первого порядка) характеризуются тем, что в них к чисто логическим аксиомам добавляются собственные аксиомы, в которых участвуют конкретные (индивидуальные) предикатные буквы и константы. Примеры индивидуальных предикатных букв – предикаты $=, <$, функциональных букв – знаки арифметических операций, предметных констант – натуральные числа, единица в теории групп, пустое множество в теории множеств.

Важная особенность прикладных исчислений заключается в том, что в схемах аксиом P1 и P2 участвуют уже не предметные переменные, а произвольные термы. Более точно эти схемы аксиом принимают следующий вид:

$$P1^*. \forall x F(x) \rightarrow F(t)$$

$$P2^*. F(t) \rightarrow \exists x F(x),$$

где $F(t)$ – результат подстановки терма t в $F(x)$ вместо всех свободных вхождений x , причем все переменные t должны быть свободными в $F(t)$.

1. Исчисления с равенством.

Большинство прикладных исчислений содержит предикат равенства $=$ и определяющие его аксиомы.

Аксиомы для $=$:

E1. $\forall x (x = x)$; (конкретная аксиома)

E2. $(x = y) \rightarrow (F(x, x) \rightarrow F(x, y))$ (схема аксиом),

где $F(x, y)$ получается из $F(x, x)$ заменой некоторых (необязательно всех) вхождений x на y при условии, что y в этих вхождениях также остается свободным. Всякая теория, в которой E1 и E2 являются теоремами или аксиомами, называется *теорией* (или исчислением) *с равенством*. Из E1 и E2 выводимы основные свойства равенства – рефлексивность, симметричность и транзитивность.

Теорема 8. В любой теории с равенством:

1) $\vdash t = t$ для любого терма t ;

2) $\vdash x = y \rightarrow y = x$;

3) $\vdash x = y \rightarrow (y = z \rightarrow x = z)$.

2. Теория частичного строгого порядка

Пример прикладного исчисления – теория частичного строгого порядка, содержащая две конкретные аксиомы для предиката “ $<$ ”:

NE1. $\forall x \neg (x < x)$;

NE2. $\forall x \forall y \forall z (x < y \rightarrow (y < z \rightarrow x < z))$.

Два сходных утверждения о транзитивности даются в разной форме: теорема 8, без кванторов, в открытой форме, а в NE2 – с кванторами, в замкнутой форме. Эти два способа задания аксиом равносильны: от первого ко второму переходим по правилу обобщений, а от второго к первому – по аксиоме P1* и правилу заключения.

Если к аксиомам NE1 и NE2 добавить аксиому с предметной константой a

NE3. $\forall x \neg (x < a)$, то получим теорию частичного порядка с минимальным элементом a .

3. Формальная арифметика.

Наиболее изученной формальной теорией, которая играет фундаментальную роль в основаниях математики, является формальная арифметика. Перечислим ее схемы аксиом:

A1. $F(0) \& \forall x(F(x) \rightarrow F(x')) \rightarrow F(x)$ (принцип индукции)

A2. $t_1' = t_2' \rightarrow t_1 = t_2$;

A3. $\lceil (t_1' = 0)$;

A4. $t_1 = t_2 \rightarrow (t_1 = t_3 \rightarrow t_2 = t_3)$;

A5. $t_1 = t_2 \rightarrow t_1' = t_2'$;

A6. $t_1 + 0 = t_1$;

A7. $t_1 + t_2' = (t_1 + t_2)'$;

A8. $t_1 \cdot 0 = 0$;

A9. $t_1 \cdot t_2' = t_1 \cdot t_2 + t_1$.

В этих аксиомах использованы три функциональных символа $+$, \cdot , $'$, один индивидуальный предикат (предикатная буква) $=$ и одна предметная константа 0 . Они придают схемам аксиом A2 – A9 конкретный вид: все предикатные и функциональные буквы в них зафиксированы, и единственный способ их варьировать – это подставлять различные термы вместо метапеременных t_1 и t_2 .

Схемы A6 – A9 – это просто предикат равенства, в который подставлены термы определенного вида. Схема A1 имеет вид: $F(x)$ – метаобозначение употребляемое в том же смысле, в каком оно использовалось в чистом исчислении предикатов. Схемы A2 – A9 можно заменить конкретными аксиомами (т.е. формулами арифметики), A2 – A9 можно получить с помощью правила обобщения и схемы аксиом P1.

Лекция 11

МЕТАТЕОРИЯ ЛОГИЧЕСКИХ ИСЧИСЛЕНИЙ

Под метатеорией логических исследований понимается изучение их общих свойств и соответствия этих свойств целям, ради которых исчисления создавались. Некоторые задачи такого рода (связь между доказуемостью и истинностью) уже рассматривались. Здесь они будут систематизированы и изучены более подробно.

Интерпретация и модели.

Ценность всякой формальной теории, в конечном счете, определяется ее способностью описывать какие-то объекты и связи между ними. Поэтому один из первых для любой теории вопросов – это вопрос о том, для описания каких объектов пригодна данная теория. Конечно, если ставить его как общую проблему соответствия научных знаний о мире самому миру, то он не может обсуждаться средствами лишь математики

(поскольку математика в отличие от естественных наук имеет дело не непосредственно с миром, а с формальными описаниями, его различных фрагментов) и становится фундаментальной проблемой философии и методологии науки. Речь пойдет о более простом случае, когда множество объектов, для которого строится формальная теория, само по себе представляет достаточно строго описанный предмет исследований. Более точно проблема адекватности формальной теории и описываемых ею объектов будет рассматриваться как математическая задача о соответствии между содержательно построенной теорией, рассматриваемой как множество объектов с операциями и отношениями на нем (т. е. как алгебраическая система), и множеством высказываний об этой теории, построенном как формальное исчисление. При такой постановке задачи интуитивно ясное понятие интерпретации приобретает точный математический смысл.

Интерпретация формальной теории состоит из множества M и однозначного отображения, которое каждой предикатной букве P_i^n ставит в соответствие n -местное отношение на M (интерпретирует ее как отношение на M), каждой функциональной букве f_j^n – n -местную операцию на M , каждой предметной константе – элемент M . Постоянные термы исчисления (не содержащие предметных переменных) при таком определении также отобразятся в элементы M . Таким образом, множество M (называемое областью интерпретации) рассматривается как основное множество алгебраической системы.

Всякая замкнутая, т. е. не содержащая свободных переменных, формула теории представляет собой высказывание об элементах, отношениях и функциях M , которое может быть истинным или ложным. Значения истинности составных формул вычисляются в соответствии с входящими в них логическими операциями. Открытая формула соответствует некоторому отношению на M , при подстановке предметных констант она превращается в высказывание о том, что между элементами M , соответствующими подставленным константам выполняется данное отношение. Открытая формула называется *выполнимой* в данной интерпретации, если существует такая подстановка предметных констант, при которой она превращается в истинное высказывание. Формула называется *истинной в данной интерпретации*, если она выполняется (т. е. превращается в истинное высказывание) при любой подстановке констант. Формула называется *ложной* в данной интерпретации, если она невыполнима.

Интерпретация (а иногда область интерпретации M) называется *моделью* для множества формул Γ , если любая формула Γ истинна в данной интерпретации. Интерпретация называется *моделью теории T* , если она является моделью множества всех теорем теории T , т. е. если всякая формула, доказуемая в T , истинна в данной интерпретации.

Если в T доказуема некоторая открытая формула F (которая, строго говоря, высказыванием не является), то в модели теории T должны быть

истинными все высказывания, получающиеся из F всеми возможными подстановками констант на место свободных переменных формулы F , и, следовательно, должно быть истинно высказывание $\forall x_1 \dots \forall x_n F$, где x_1, \dots, x_n – свободные переменные формулы F .

Пример. В теории строгого частичного порядка (с аксиомами NE1 и NE2) моделью будет любая интерпретация, при которой предикатная буква $<$ интерпретируется отношением «быть меньше». Однако почти столь же очевидно из аксиом NE1 и NE2 (хотя и выглядит парадоксально), что моделью этой теории будет и интерпретация, при которой символ “ $<$ ” интерпретируется как отношение «быть больше».

Пример иллюстрирует существо формального подхода: в символы исчисления (даже самые привычные) не вкладывается никакого смысла, пока не введена их явная интерпретация. Но и введенная интерпретация, вообще говоря, не относится к числу средств самого исчисления: она позволяет осмыслить формулы исчисления, но не участвует в формальном выводе теорем.

О формальных свойствах самого исчисления, его формул и формальных преобразований над ними принято говорить как о *синтаксисе исчисления*; свойства исчисления, описанные в терминах его интерпретаций, – это *семантика исчисления*.

Непротиворечивость.

Формула называется *общезначимой*, если она истинна в любой интерпретации, и *противоречивой*, если она ложна в любой интерпретации, т. е. если ее отрицание общезначимо. Для любого множества общезначимых формул \mathcal{F} , в частности, для чистого исчисления предикатов (по теореме 6.5) любая алгебраическая система и вообще любое множество является моделью. Напротив, для любого множества формул, содержащего хотя бы одну противоречивую формулу, моделей не существует.

Аксиома E1 теории с равенством не является общезначимой: существуют интерпретации (в смысле, указанном в начале), в которых она ложна. Примером такой интерпретации может служить всякое отображение, которое предикатной букве $=$ ставит в соответствие отношение $<$. (Здесь символы $=$ и $<$ используются на разных уровнях: символ $=$ – как формальная предикатная буква, не имеющая смысла до ее интерпретации, а символ $<$ – как символ отношения на множестве, имеющий всем известный смысл «быть меньше».) Собственные аксиомы прикладных исчислений вообще не общезначимы – иначе по теореме об общезначимой формуле в исчислении предикатов они были бы доказуемы в чистом исчислении предикатов.

Введенное ранее определение противоречивой формулы является семантическим, т.е. связывающим непротиворечивость с истинностью.

Исходя из него, можно сформулировать понятие семантически непротиворечивой теории: теория *семантически непротиворечива*, если ни одна из ее теорем не является противоречивой, т.е. ложной в любой интерпретации. Исчисление высказываний и исчисление предикатов семантически непротиворечивы в силу теорем (о теореме исчисления высказываний и о доказуемой формуле исчисления предикатов): все их теоремы общезначимы и, следовательно, непротиворечивы.

С помощью введенных понятий ответ на общий вопрос, поставленный в начале лекции, формулируется так: теория T пригодна для описания тех множеств, которые являются ее моделями; модель для теории T существует тогда и только тогда, когда T семантически непротиворечива.

Чисто логические теории – *исчисление высказываний* и *исчисление предикатов* – в силу теорем (о теореме исчисления высказываний и о доказуемой формуле исчисления предикатов) пригодны для описания любых множеств, что соответствует общенаучному принципу универсальности законов логики. Лейбниц формулировал его как выполнимость логических законов «во всех мыслимых мирах». Такой критерий пригодности теорий по существу известен уже давно; отыскание модели для теории до возникновения оснований математики было единственным общепризнанным методом доказательства законности теории. Одно из важных достижений оснований математики заключается в том, что аналог этого критерия сформулирован в терминах самих формальных теорий, без привлечения семантических понятий. Таким аналогом является понятие формальной, или дедуктивной непротиворечивости.

Теория T называется *формально непротиворечивой*, если не существует формулы F , такой, что F и $\neg F$ являются теоремами теории T , т. е. в T не выводимы одновременно формула и ее отрицание. Аналогичное определение можно сформулировать и для произвольного множества формул.

Для любой теории, содержащей исчисление высказываний, из определения непосредственно следует, что если она формально противоречива, то в ней доказуема тождественно-ложная формула и, следовательно, она семантически противоречива.

Обратное утверждение (которое также оказывается верным), если понимать его конструктивно, гораздо глубже. Смысл его в том, что по всякому формально непротиворечивому множеству формул можно построить его модель. Доказательство этого факта довольно сложно и заключается в изложении метода такого построения. Вместе оба утверждения образуют важную метатеорему.

Теорема 1. Множество формул формально непротиворечиво, если и только если оно семантически непротиворечиво (т.е. имеет модель).□

Таким образом, понятие формальной непротиворечивости оказывается эквивалентным более привычному в математике понятию семантической непротиворечивости; однако оно сформулировано в синтаксических терминах и с конструктивной точки зрения более надежно.

Неприятие современниками неевклидовых геометрий Лобачевского–Бойаи объясняется именно тем, что законность этих теорий обосновывалась отсутствием в них противоречий – аргументом, по существу совпадающим с современным понятием формальной непротиворечивости. Геометрические модели для этих теорий, доказывающие их семантическую непротиворечивость, были найдены позднее.

Полнота, разрешимость, аксиоматизируемость.

Пусть имеется, с одной стороны, содержательная теория S , сформулированная в семантических терминах, т.е. совокупность истинных высказываний о некоторой алгебраической системе M ; с другой стороны — формальная теория T , т.е. множество выражений, выводимых из аксиом теории T с помощью правил вывода теории T . В каких случаях можно утверждать, что T является удовлетворительной формализацией S ? Каковы признаки удовлетворительности формализации?

Очевидно, необходимым признаком является условие, чтобы S была моделью теории T , т.е. чтобы существовало отображение, при котором всякая теорема теории T отображается в истинное высказывание из S . Однако само по себе это условие явно недостаточно, иначе для формализации любой теории S хватило бы чистого исчисления предикатов: ведь для него любое множество является моделью. Исчерпывающей формализацией S будет служить такая теория T , для которой выполняется и обратное соответствие: каждое истинное высказывание теории S отображается в некоторую теорему теории T . Теория T с таким свойством называется *полной* относительно S , а иногда – *адекватной* S .

Из теорем (исчисления высказываний) следует, что исчисление высказываний полно относительно алгебры высказываний, а теоремы (исчисления предикатов) образуют известную *теорему Геделя о полноте исчисления предикатов* относительно логики предикатов, т.е. множества всех общезначимых предикатных формул.

Если для семантической (содержательной) теории S удастся построить непротиворечивую и полную формальную теорию T , то S естественно назвать *аксиоматизируемой*, или *формализуемой теорией*. Из предыдущих результатов следует, что логика высказываний и логика предикатов аксиоматизируемы с помощью соответствующих исчислений.

Еще одна важная характеристика формальной теории – это ее *разрешимость*. Формальная теория T называется *разрешимой*, если существует алгоритм, который для любой формулы языка определяет,

является она теоремой в T или нет, иначе говоря, если предикат «быть теоремой в теории T » общерекурсивен.

Теорема 2. Исчисление высказываний разрешимо.

Разрешающий алгоритм для формулы F исчисления высказываний заключается в вычислении значений F на всех наборах значений ее переменных. Ввиду полноты исчисления высказываний F является его теоремой, если и только если она истинна на всех наборах.

Теорема 3 (теорема Черча). Исчисление предикатов неразрешимо.

Несмотря на полноту исчисления предикатов, разрешающий алгоритм, связанный с вычислением значений истинности предикатных формул, построить не удастся из-за бесконечности предметной области, которая приводит в общем случае к бесконечным таблицам истинности. *Идея доказательства* теоремы Черча (которое здесь не приводится) состоит в том, чтобы в чистом исчислении предикатов описать предикат $Q(i, a, x)$: «машина Тьюринга с номером i , будучи применена к исходным данным a , закончит вычисление в момент x ». Предикат $\exists x Q(i, a, x)$ – это предикат остановки, а $\exists x Q(a, a, x)$ – предикат самоприменимости; оба предиката неразрешимы.

Отметим, что важный для приложений фрагмент исчисления предикатов разрешим: исчисление одноместных предикатов (т.е. исчисление, допускающее в формулах только одноместные предикатные буквы) разрешимо.

Еще тяжелее обстоит дело с формальной арифметикой, система аксиом которой приведена в конце лекции №__.

Теорема 4 (первая теорема Геделя о неполноте в форме Клини). Любая формальная теория T , содержащая формальную арифметику, неполна: в ней существует (и может быть эффективно построена) замкнутая формула F , такая, что $\neg F$ истинно, но ни F , ни $\neg F$ не выводимы в T .

Теорема 5 (вторая теорема Геделя о неполноте). Для любой непротиворечивой формальной теории T , содержащей формальную арифметику, формула, выражающая непротиворечивость T , недоказуема в T .

Арифметика и теория чисел оказываются неаксиоматизируемыми теориями. В терминах теории алгоритмов сформулированные ранее результаты можно резюмировать как:

- 1) множество всех истинных высказываний логики высказываний перечислимо и разрешимо;
- 2) множество всех истинных высказываний логики предикатов перечислимо (ввиду его полноты), но неразрешимо;
- 3) множество всех истинных высказываний арифметики неперечислимо: если бы для него существовала перечисляющая процедура (и, следовательно, соответствующая машина Тьюринга), то по ней можно было бы построить полную формальную теорию, рассматривая правила перехода машины от одной конфигурации к другой

как правила вывода, процесс вычисления – как вывод, а результаты вычисления – как теоремы.

Две знаменитые теоремы Геделя имеют важное методологическое значение. Из *первой теоремы* следует, что для достаточно богатых математических теорий не существует адекватных формализаций. Правда, любую неполную теорию T можно расширить, добавив, например, к ней в качестве новой аксиомы истинную, но не выводимую в T формулу. Однако по первой теореме Геделя новая теория T также будет неполна. *Вторая теорема* может быть истолкована как невозможность исследования метасвойств теории средствами самой формальной теории (опять невозможность самоприменимости!); иначе говоря, метатеория теории T для того, чтобы иметь возможность доказывать хотя бы непротиворечивость теории T , должна быть богаче T .

По существу при этом ставится под сомнение первоначальная, «максималистская» программа финитного подхода: нельзя построить математику как некоторую фиксированную совокупность средств, которые можно было бы объявить единственно законными и с их помощью строить метатеории любых теорий.

Не следует истолковывать результаты Геделя как крах формального подхода. Наличие алгоритмически неразрешимых проблем вовсе не бросает тень на теорию алгоритмов, а лишь сообщает «суровую правду» об устройстве мира, изучаемого этой теорией. Из этой правды не вытекает, что алгоритмический, конструктивный подход к решению проблемы не пригоден; хотя он чего-то и не может, но лишь потому, что этого не может никто. Точно так же невозможность полной формализации содержательно определенных теорий – это не недостаток подхода или концепции, а объективный факт, неустранимый никакой концепцией. Формальный подход остается основным конструктивным средством изучения множеств высказываний. Невозможность адекватной формализации теории означает, что надо либо искать формализуемые ее фрагменты, либо строить какую-то более сильную формальную теорию, которая, правда, снова будет неполна, но, быть может, будет содержать всю исходную теорию. В частности, методами, не формализуемыми в формальной арифметике, Генцен доказал непротиворечивость формальной арифметики.

Лекция 12

ТЕОРИЯ АЛГОРИТМОВ

1. ПРЕДВАРИТЕЛЬНОЕ ОБСУЖДЕНИЕ

Несколько вступительных слов. С алгоритмами, т. е. эффективными процедурами, однозначно приводящими к результату, математика имела дело

всегда. Школьные методы умножения «столбиком» и деления «углом», метод исключения неизвестных при решении системы линейных уравнений, правило дифференцирования сложной функции, способ построения треугольника по трем заданным сторонам – все это алгоритмы. Однако, пока математика имела дело в основном с числами и вычислениями и понятие алгоритма отождествлялось с понятием метода вычисления, потребности в изучении самого этого понятия не возникало. Традиции организации вычислений складывались веками и стали составной частью общей научной культуры в той же степени, что и элементарные навыки логического мышления. Все многообразие вычислений комбинировалось из 10 – 15 четко определенных операций арифметики, тригонометрии и анализа. Поэтому понятие метода вычисления считалось изначально ясным и не нуждалось в специальных исследованиях.

До середины XIX в. единственной областью математики, работавшей с нечисловыми объектами, была геометрия, и как раз она, не имея возможности опираться на вычислительную интуицию человека, резко отличалась от остальной математики повышенными требованиями к строгости своих рассуждений. До сих пор любой современный шестиклассник, для которого математика — это мир вычислений (и в этом он мало отличается от типичного инженера), мучительно привыкает к понятиям доказательства и математического построения и никак не может понять, зачем доказывать равенство отрезков, когда их проще измерить, и зачем строить перпендикуляр с помощью циркуля и линейки, когда есть угольник с «готовым» прямым углом или транспортир.

Такое же мучительное привыкание к новым, более жестким требованиям строгости началось в математике во второй половине XIX в., оно стимулировалось в основном математикой нечисловых объектов – открытием неевклидовых геометрий, появлением абстрактных алгебраических теорий типа теории групп и т.д. Одним из решающих обстоятельств, приведших к пересмотру оснований математики, т. е. принципов, лежащих в основе математических рассуждений, явилось создание Кантором теории множеств. Довольно быстро стало ясно, что понятия теории множеств в силу своей общности лежат в основе всего здания математики. Однако почти столь же быстро было показано, что некоторые кажущиеся вполне естественными рассуждения в рамках этой теории приводят к неразрешимым противоречиям – парадоксам теории множеств (которые упоминались). Все это потребовало точного изучения принципов математических рассуждений (до сих пор казавшихся интуитивно ясными) математическими же средствами. Возникла особая отрасль математики – основания математики, или метаматематики.

Опыт парадоксов теории множеств научил математику крайне осторожно обращаться с бесконечностью и по возможности даже о бесконечности рассуждать с помощью финитных методов. Существо финитного подхода заключается в том, что он допускает только конечные

комплексы действий над конечным числом объектов. Выяснение того, какие объекты и действия над ними следует считать точно определенными, какими свойствами и возможностями обладают комбинации элементарных действий, что можно и чего нельзя сделать с их помощью, – все это стало предметом теории алгоритмов и формальных систем, которая первоначально возникла в рамках метаматематики и стала важнейшей ее частью. Главным внутриматематическим приложением теории алгоритмов явились доказательства невозможности алгоритмического (т. е. точного и однозначного) решения некоторых математических проблем. Такие доказательства (да и точные формулировки доказываемых утверждений) неосуществимы без точного понятия алгоритма.

Пока техника использовала чисто вычислительные методы, эти высокие проблемы чистой математики ее мало интересовали. В технику термин «алгоритм» пришел вместе с кибернетикой. Если понятие метода вычисления не нуждалось в пояснениях, то понятие процесса управления пришлось выработать практически заново. Понадобилось осознать, каким требованиям должна удовлетворять последовательность действий (или ее описание), чтобы считаться конструктивно заданной, т. е. иметь право называться алгоритмом. В этом осознании огромную помощь инженерной интуиции оказала практика использования вычислительных машин, сделавшая понятие алгоритма ощутимой реальностью. С точки зрения современной практики алгоритм—это программа, а критерием алгоритмичности процесса является возможность его запрограммировать. Именно благодаря этой реальности алгоритма, а также потому, что подход инженера к математическим методам всегда был конструктивным, понятие алгоритма в технике за короткий срок стало необычайно популярным (быть может, даже больше, чем в самой математике).

Однако у всякой популярности есть свои издержки. В повседневной практике слово «алгоритм» употребляется слишком широко, теряя зачастую свой точный смысл. Приблизительные описания понятия «алгоритм» (вроде того, которое приведено в первой фразе этого параграфа) часто принимаются за точные определения. В результате за алгоритм зачастую выдается любая инструкция, разбитая на шаги. Появляются такие дикие словосочетания, как «алгоритм изобретения» (а ведь наличие «алгоритма изобретения» означало бы конец изобретательства как творческой деятельности).

Ясное представление о том, что такое алгоритм, важно, конечно, не только для правильного словоупотребления. Оно нужно и при разработке конкретных алгоритмов, особенно когда имеется в виду их последующее программирование. Однако оно еще важнее при наведении порядка в бурно растущем алгоритмическом хозяйстве. Чтобы ориентироваться в море алгоритмов, захлестнувшим техническую кибернетику, необходимо уметь сравнивать различные алгоритмы решения одних и тех же задач, причем не только по качеству решения, но и по характеристикам самих алгоритмов (числу действий, расходу памяти и т.д.). Такое сравнение невозможно без

введения точного языка для обсуждения всех этих вопросов; иначе говоря, сами алгоритмы должны стать такими же предметами точного исследования, как и те объекты, для работы с которыми они предназначены.

Строгое исследование основных понятий теории алгоритмов начнется в следующем параграфе. Прежде чем приступить к нему, обсудим на неформальном уровне некоторые основные принципы, по которым строятся алгоритмы, и выясним, что же именно в понятии алгоритма нуждается в уточнении.

Основные требования к алгоритмам.

1. Первое, что следует отметить в любом алгоритме, — это то, что он применяется к исходным данным и выдает результаты. В привычных технических терминах это означает, что алгоритм имеет входы и выходы. Кроме того, в ходе работы алгоритма появляются промежуточные результаты, которые используются в дальнейшем. Таким образом, каждый алгоритм имеет дело с *данными* — входными, промежуточными и выходными. Поскольку мы собираемся уточнять понятие алгоритма, нужно уточнить и понятие данных, т. е. указать, каким требованиям должны удовлетворять объекты, чтобы алгоритмы могли с ними работать.

Ясно, что эти объекты должны быть четко определены и отличимы как друг от друга, так и от «необъектов». К алгоритмическим объектам относятся числа, векторы, матрицы смежностей графов, формулы. Изображения (например, рисунок графа) представляются менее естественными в качестве алгоритмических объектов. Если говорить о графе, то дело даже не в том, что в рисунке больше несущественных деталей и два человека один и тот же граф изобразят по-разному (в конце концов, разные матрицы смежности тоже могут задавать один и тот же граф с точностью до изоморфизма), а в том, что матрица смежности легко разбивается на элементы, причем из элементов всего двух видов (нулей и единиц) состоят матрицы любых графов, тогда как разбить на элементы рисунок гораздо труднее. Наконец, с такими объектами, как «хорошая книга» или «осмысленное утверждение», с которыми легко управляется любой человек (но каждый по-своему!), алгоритм работать откажется, пока они не будут описаны как данные с помощью других, более подходящих объектов.

Вместо того чтобы пытаться дать общее словесное определение четкой определенности объекта, в теории алгоритмов фиксируют конкретные конечные наборы исходных объектов (называемых элементарными) и конечный набор средств построения других объектов из элементарных. Набор элементарных объектов образует конечный *алфавит* исходных символов (цифр, букв и т. д.), из которых строятся другие объекты; типичным средством построения являются индуктивные определения, указывающие, как строить новые объекты из уже построенных. Простейшее индуктивное определение — это определение некоторого множества слов, классическим примером которого служит определение идентификатора в АЛГОЛе; идентификатор — это либо

буква, либо идентификатор, к которому приписана справа буква или цифра. Слова конечной длины в конечных алфавитах (в частности, числа) – наиболее обычный тип алгоритмических данных, а число символов в слове (длина слова) – естественная единица измерения объема обрабатываемой информации. Более сложный случай алгоритмических объектов – формулы. Они также определяются индуктивно и также являются словами в конечном алфавите, однако не каждое слово в этом алфавите является формулой. В этом случае обычно основным алгоритмам предшествуют вспомогательные, которые проверяют, удовлетворяют ли исходные данные нужным требованиям. Такая проверка называется синтаксическим анализом.

2. Данные для своего размещения требуют *памяти*. Память обычно считается однородной и дискретной, т. е. состоит из одинаковых ячеек, причем каждая ячейка может содержать один символ алфавита данных. Таким образом, единицы измерения объема данных и памяти согласованы. При этом память может быть бесконечной. Вопрос о том, нужна ли одна память или несколько и, в частности, нужна ли отдельная память для каждого из трех видов данных (входных, выходных и промежуточных), решается по-разному.

3. Алгоритм состоит из отдельных *элементарных* или *действий*, причем множество различных шагов, из которых составлен алгоритм, конечно. Типичный пример множества элементарных действий — система команд. Обычно элементарный шаг имеет дело с фиксированным числом символов (это удобно, например, для измерения времени работы алгоритма числом проделанных шагов), однако это требование не всегда выполняется. Например, в ЭВМ третьего поколения есть команды типа «память - память», работающие с полями памяти переменной длины.

4. Последовательность шагов алгоритма *детерминирована*, т. е. после каждого шага либо указывается, шаг делать дальше, либо дается команда остановки, после чего работа алгоритма считается законченной.

5. Естественно от алгоритма потребовать *результативности* т. е. остановки после конечного числа шагов зависящего от данных) с указанием того, что считать результатом. В частности, всякий, кто предъявляет алгоритм решения некоторой задачи, например вычисления функции $f(x)$, обязан показать, что алгоритм останавливается после конечного числа шагов (как говорят, *сходится*) для, любого x из области задания. Однако проверить результативность (сходимость) гораздо труднее. В отличие от них сходимость обычно не удается установить простым просмотром описания алгоритма; общего же метода проверки сходимости годного для любого алгоритма A и любых данных будет показано далее, вообще не существует. Обсуждение трудностей, связанных с распознаванием сходимости далее.

6. Следует различать: а) описание алгоритма (конструкцию или программу); б) механизм реализации алгоритма (например, ЭВМ), включающий средства остановки, реализации элементарных шагов, выдача результатов и обеспечения детерминированности, т.е. управления ходом вычисления; в)

процесс реализации алгоритма, т. е. последовательность шагов, которая будет порождена при применении алгоритма к конкретным данным.

Будем предполагать, что описание алгоритма и механизм его реализации конечны (память, как уже говорилось, может быть бесконечной, но она не включается в механизм). Требования к конечности процесса реализации совпадают с требованиями результативности.

Необходимо уточнить формы представления данных. При этом нельзя просто заявить, что допустимо любое представление чисел. Ведь для каждого представления существует свой алфавит (который, помимо цифр, может включать запятые, скобки, знаки операций и функций и т. д.) и свой способ сравнения чисел (например, способ перевода в десятичную дробь), тогда как конечность алфавита требует фиксировать его заранее, а конечность описания алгоритма позволяет включить в него лишь заранее фиксированное число способов сравнения. Фиксация представления чисел в виде десятичных дробей так же не решает всех проблем. Сравнение 10—20-разрядных уже не может считаться элементарным действием: сразу нельзя сказать, какое из чисел больше: 90811557001,1 32899901467,0048. В машинных алгоритмах само представление числа еще требует дальнейшего уточнения: нужно во-первых, ограничить число разрядов (цифр) в числе, так как от этого зависит, сколько ячеек памяти будет занимать число, а во-вторых, договориться о способе размещения десятичной запятой в числе, т. е. выбрать представление в виде числа с фиксированной или плавающей запятой, поскольку способы обработки этих двух представлений различны. Наконец, любой, кто имел дело с программированием, отметит, что на шаге 1 требуется узнать две вещи: само наименьшее число (чтобы записать его в R) и его место в P , т. е. его адрес в той части m где хранится P (чтобы вычеркнуть его из P), а следовательно нужно иметь средства указания этого адреса.

Таким образом, даже в этом простом примере несложный анализ показывает, что описанию, которое выглядит вполне ясным, еще далеко до алгоритма. Мы столкнулись здесь с необходимостью уточнить почти все основные характеристики алгоритма, которые отмечались ранее: алфавит данных и форму их представления, память и размещение в ней элементов P и R , элементарные шаги (поскольку шаг 1 явно не элементарен). Кроме того, становится ясным, что выбор механизма реализации (скажем, человека или ЭВМ) будет влиять и на сам характер уточнения: у человека требования к памяти, представлению данных и к элементарности шагов гораздо более слабые и «укрупненные», отдельные незначительные детали он может уточнить сам.

Пожалуй, только два требования к алгоритмам в приведенном описании выполнены в достаточной мере (они то и создают впечатление ясности). Довольно очевидна сходимость алгоритма: после выполнения шагов 1 и 2 либо работа заканчивается, либо из P вычеркивается одно число; поэтому

ровно после n выполнений шагов 1 и 2 из P будут вычеркнуты все числа и алгоритм остановится, а R будет результатом. Кроме того, не вызывает сомнения детерминированность: после каждого шага ясно, что делать дальше, если учесть, что здесь и в дальнейшем используется общепринятое соглашение — если шаг не содержит указаний о дальнейшем переходе, то выполняем шаг, следующий за ним в описании. Поскольку использованные в примере средства обеспечения детерминированности носят довольно общий характер, остановимся на них несколько подробнее.

Блок-схемы алгоритмов. Связи между шагами можно изобразить в виде графа. Для примера 1 граф изображен на рис. 1.



Рис.1

Такой граф, в котором шаги, а ребрам — переходы между шагами, называется блок-схемой алгоритма. Его вершины могут быть двух видов: вершины, из которых выходит одно ребро (их называют операторами), и вершины, из которых выходят два ребра (их называют логическими условиями, или предикатами). Кроме того, имеется единственный оператор конца, из которого не выходит ни одного ребра, и единственный оператор начала. Важной особенностью блок-схемы является то, что связи, которые она описывает, не зависят от того, являются ли шаги элементарными или представляют собой самостоятельные алгоритмы, — как говорят в программировании, блоки (по существу шаг 1 таковым и является). Возможность «разблочивать» алгоритм хорошо известна в программировании и широко там используется: большой алгоритм разбивается на блоки, которые можно раздать для программирования разным лицам. Для данного блока неважно, как устроены другие блоки; для программирования блока достаточно знать, где лежит вся исходная информация, какова форма ее представления, что должен делать блок и куда записать результат.

С помощью блок-схем можно, наоборот, несколько алгоритмов, рассматриваемых как блоки, связать в один большой алгоритм. В частности, если алгоритм A_1 вычисляющий функцию $f_1(x)$, соединен с алгоритмом A_2 , вычисляющим функцию $f_2(x)$ (рис. 2), и при этом исходными данными для A_2 служит результат A_1 , то полученная блок-схема задает алгоритм, вычисляющий

Рис. 2



$f_2(f_1(x))$, т. е. композицию f_1 и f_2 . Такое соединение алгоритм называется композицией алгоритмов.

На блок-схеме хорошо видна разница между описанием алгоритма и процессом его реализации. Описание – это граф; процесс реализации – это путь в графе. Различные пути в одном и том же графе возникают при различных данных, которые создают разные логические условия в точках разветвления. Отсутствие сходимости означает, означает, что в процессе вычисления не появляется условий, ведущих к концу, и процесс идет по бесконечному пути (заиклиивается).

При всей наглядности языка блок-схем не следует, однако, переоценивать его возможности. Он достаточно груб и отражает связи лишь по управлению (что делать в следующий момент, т. е. какому блоку передать управление), а не по информации (где этому блоку брать исходные данные). Например, рис. 2 при сделанной оговорке относительно данных изображает вычисление $f_2(f_1(x))$, однако он же мог изображать последовательное вычисление двух независимых функций $f_1(5)$ и $f_2(100)$, порядок которых несуществен. Блок-схемы не содержат сведений ни о данных, ни о памяти, ни об используемом наборе элементарных шагов. В частности, надо иметь в виду, что если в блок-схеме нет циклов, это еще не значит, что нет циклов в алгоритме (они могут быть в каком-нибудь неэлементарном блоке). По существу блок-схемы – это не язык; а средство, правда, очень удобное, для одной цели — описания детерминизма алгоритма. Оно будет неоднократно использоваться в дальнейшем с одним видоизменением: условия, т.е. точки разветвления, могут быть не только двоичными, но и многозначными; важно лишь, чтобы верным был ровно один из возможных ответов. Примеры таких условий: а) $x < 0$, $x = 0$, $x > 0$; б) $x < 5$, $5 \leq x < 20$, $x = 20$, $x > 20$.

О подходах к уточнению понятия «алгоритм».

Ранее были сформулированы основные требования к алгоритмам. Однако понятия, использованные в формулировках (ясность, четкость, элементарность), сами нуждаются в уточнении. Очевидно, что их словесные определения будут содержать новые понятия, которые потребуют уточнения, и т. д.

Поэтому в теории алгоритмов принимается другой подход: выбирается конечный набор исходных объектов, которые объявляются элементарными, и конечный набор способов построения из них новых объектов. Этот подход был использован при обсуждении вопроса о данных: уточнением понятия «данные» в дальнейшем будем считать множества слов в конечных алфавитах. Для уточнения детерминизма будут использоваться либо блок-схемы и эквивалентные им словесные описания, либо описание механизма реализации алгоритма. Кроме того, нужно зафиксировать набор элементарных шагов и договориться об организации памяти. После этого получится конкретная алгоритмическая модель.

Рассматриваемые алгоритмические модели, претендуют на право считаться формализацией понятия «алгоритм». Это значит, что они должны быть универсальными, т.е. допускать описание любых алгоритмов. Может возникнуть естественное возражение против предлагаемого подхода: не

приведет ли выбор конкретных средств к потере общности формализации? Если иметь ввиду основные цели, стоящие при создании теории алгоритмов, - универсальность и связанную с ней возможность говорить в рамках какой-либо модели о свойствах алгоритмов вообще, то это возражение снимается. Во-первых, доказываемость сводимости одних моделей к другим, т.е. показывается, что всякий алгоритм, описанный средством одной модели, может быть описан средствами другой. Во-вторых, благодаря взаимной сводимости моделей в теории алгоритмов удалось выработать инвариантную по отношению к моделям систему понятий, позволяющую говорить о свойствах алгоритмов независимо от того, какая формализация алгоритма выбрана. Эта система понятий основана на понятии вычислимой функции, т. е. функции, для вычисления которой существует алгоритм. Общее понятие вычислимости и его свойства будут более подробно рассмотрены.

Тем не менее, хотя общность формализации в конкретной модели не теряется, различный выбор исходных средств приводит к моделям разного вида. Можно выделить три основных типа *универсальных алгоритмических моделей* различающихся исходными эвристическими соображениями относительно того, что такое алгоритм.

I. Первый тип связывает понятие алгоритма с наиболее традиционными понятиями математики — вычислениями и числовыми функциями.

Наиболее развитая и изученная модель этого типа *рекурсивные функции* — является исторически первой формализацией понятия алгоритма.

II. Второй тип основан на представлении об алгоритме как о некотором детерминированном устройстве, способном выполнять в каждый отдельный момент лишь весьма примитивные операции. Такое представление не оставляет сомнений в однозначности алгоритма и элементарности его шагов. Кроме того, эвристика этих моделей близка к ЭВМ и, следовательно, инженерной интуиции. Основной теоретической моделью этого типа (созданной в 30-х годах —раньше ЭВМ!) является машина Тьюринга.

III. Третий тип алгоритмических моделей — это преобразования слов в произвольных алфавитах, в которых элементарными операциями являются подстановки, т. е. замены куска слова (подслова) другим словом. Преимущества этого типа — в его максимальной абстрактности и возможности применить понятие алгоритма к объектам произвольной (не обязательно числовой) природы. Впрочем, как будет ясно из дальнейшего, модели второго и третьего типа довольно близки (их взаимная сводимость доказывается просто) и отличаются в основном эвристическими акцентами. Примеры моделей этого типа — канонические системы Поста и нормальные алгоритмы Маркова.

МАШИНЫ ТЬЮРИНГА

Основные определения.

Определение. Машина Тьюринга состоит: 1) управляющего устройства, которое может находиться в одном из состояний, образующих конечное множество $Q = \{q_1, \dots, q_n\}$; 2) ленты, разбитой на ячейки, в каждой из которых может быть записан один из символов конечного алфавита $A = \{a_1, \dots, a_m\}$; 3) устройства обращения к ленте, считывающей и пишущей головки, которая в каждый момент времени обзвевает ячейку ленты, в зависимости от символа в этой ячейке и состояния управляющего устройства, записывает в ячейку символ (быть может, совпадающий с прежним или пустой, т. е. стирает символ), сдвигается на ячейку влево или вправо или остается на месте; при этом управляющее устройство переходит в новое состояние (или остается в старом).

Среди состояний управляющего устройства выделены начальное состояние q_1 и заключительное состояние, которое будем обозначать q_z (z здесь понимается не как числовая переменная, а как мнемонический знак конца). В начальном состоянии машина находится перед началом работы; попав в заключительное состояние, машина останавливается.

Таким образом, память машины Тьюринга – это конечное множество состояний (внутренняя память) и лента (внешняя память). Лента бесконечна в обе стороны, однако в начальный момент времени только конечное число ячеек ленты заполнено непустыми символами, остальные ячейки пусты, т. е. содержат пустой символ λ (пробел). Из характера работы машины следует, что и в любой последующий момент времени лишь конечный отрезок ленты будет заполнен символами. Поэтому важна не фактическая (как говорят в математике, актуальная) бесконечность ленты, а ее неограниченность, т. е. возможность писать на ней сколь угодно длинные, но конечные слова.

Данные машины Тьюринга – это слова в алфавите ленты; на ленте записываются и исходные данные, и окончательные результаты.

Элементарные шаги машины – это считывание и запись символов, сдвиг головки на ячейку влево и вправо, а также переход управляющего устройства в следующее состояние.

Детерминированность машины, т. е. последовательность ее шагов, определяется следующим образом: для любого внутреннего состояния q_i и символа a_j однозначно заданы: а) следующее состояние q'_i ; б) символ a'_j который нужно записать вместо a_j в ту же ячейку (стирание символа будем понимать как запись пустого символа λ); в) направление сдвига головки d_k , обозначаемое одним из трех символов, L (влево), R (вправо), E (на месте). Это задание может описываться либо системой правил (команд), имеющих вид:

$$q_i a_j \rightarrow q'_i a'_j d_k, \quad (1)$$

либо таблицей, строкам которой соответствуют состояния, столбцам — входные символы, а на пересечении строки q_i и столбца a_j записана тройка символов $q'_i a'_j d_k$ и, наконец, блок-схемой, которую будем называть диаграммой переходов. В диаграмме переходов состояниям соответствуют вершины, а правилу вида (1)—ребро, ведущее из q_i в q'_i , на котором написано $a_j \rightarrow a'_j d_k$. Условие однозначности требует, чтобы для любого j и любого $i \neq z$ в системе команд имелась одна команда, аналогичная (1), с левой частью $q_i a_j$; состояние q_z в левых частях команд не встречается. На диаграмме переходов это выражается условием, что из каждой вершины, кроме q_z , выходят ровно m ребер, причем на разных ребрах левые части различны; в вершине q_z нет выходящих ребер. В дальнейшем договоримся опускать символы $q'_i a'_j$, если $q'_i = q_i$, $a'_j = a_j$.

Полное состояние машины Тьюринга, по которому однозначно можно определить ее дальнейшее поведение, определяется ее внутренним состоянием, состоянием ленты (т. е. словом, записанным на ленте) и положением головки на ленте.

Полное состояние будем называть *конфигурацией*, или машинным словом, и обозначать тройкой $\alpha_1 q_i \alpha_2$, где q_i — текущее внутреннее состояние, α_1 — слово слева от головки, а α_2 — слово, образованное символом, обозреваемым головкой, и символами справа от него, причем слева от α_1 и справа от α_2 нет непустых символов. Например, конфигурация с внутренним состоянием q_i , в которой на ленте записано $a b c d e$, а головка обозревает d , запишется как $abcq_i d e$.

Стандартной начальной конфигурацией назовем конфигурацию вида $q_1 \alpha$, т. е. конфигурацию, содержащую начальное состояние, в которой головка обозревает крайний левый символ слова, написанного на ленте. Аналогично *стандартной заключительной конфигурацией* назовем конфигурацию вида $q_z \alpha$. Ко незаключительной конфигурации K машины T применима ровно одна команда вида (1), которая K переводит в конфигурацию K' . Это отношение между конфигурациями обозначим $K \rightarrow_T K'$; если из контекста ясно, о какой машине T идет речь, индекс T будем опускать. Если для K_1 и K_n существует последовательность конфигураций K_1, K_2, \dots, K_n , такая, что $K_1 \rightarrow_T K_2 \rightarrow_T \dots \rightarrow_T K_n$, обозначим это $K_1 \Rightarrow_T K_n$.

Например, если в системе команд машины T имеются команды $q_2 a_5 \rightarrow q_3 a_4 R$ и $q_3 a_1 L \rightarrow q_4 a_2$, то $q_2 a_5 a_1 a_2 \rightarrow a_4 q_3 a_1 a_2 \rightarrow q_4 a_2 q_4 a_4 a_2 a_2$ и, следовательно $q_2 a_5 a_1 a_2 \Rightarrow_T q_4 a_2 q_4 a_4 a_2 a_2$. Последовательность конфигураций $K_1 \rightarrow_T K_2 \rightarrow_T K_3 \rightarrow_T \dots$ однозначно определяется исходной конфигурацией K_1 и полностью описывает работу машины T , начиная с K_1 . Она конечна, если в ней встретится заключительная конфигурация, и бесконечна в противном случае.

Пример 1.

а. Машина с алфавитом $A = \{1, \lambda\}$, состояниями $\{q_1, q_z\}$ и системой команд $q_1 1 \rightarrow q_1 1 R$, $q_1 \lambda \rightarrow q_1 1 R$ из любой начальной конфигурации будет работать бесконечно, заполняя единицами всю ленту вправо от начальной точки.

б. Для любой машины T , если $K_1 \Rightarrow_T K_2 \Rightarrow_T K_3$ и $K_2 = K_3$, последовательность $K_1 \Rightarrow_T K_2 \Rightarrow_T K_3 \Rightarrow_T \dots$ является бесконечной; ее отрезок $K_1 \Rightarrow_T K_3$ будет повторяться (заиклиться).

Если $\alpha_1 q_1 \alpha_2 \Rightarrow \beta_1 q_z \beta_2$, то будем говорить, что машина T перерабатывает слово $\alpha_1 \alpha_2$ в слово $\beta_1 \beta_2$, и обозначать это $T(\alpha_1 \alpha_2) = \beta_1 \beta_2$. Запись $T(\alpha)$ иногда будем употреблять и в другом смысле — просто как обозначение машины T с исходными значениями α .

Действия машины Тьюринга.

Для того чтобы говорить о том, что могут делать машины Тьюринга, необходимо уточнить, как будет интерпретироваться их поведение, и как будут представляться данные. Исходными данными машины Тьюринга будем считать записанные на ленте слова в алфавите исходных данных $A_{исх}$ ($A_{исх} \subseteq A$) и векторы из таких слов (словарные векторы над $A_{исх}$). Это значит, что для каждой машины будут рассматриваться только те начальные конфигурации, в которых на ленте записаны словарные векторы над $A_{исх}$. Запись на ленте словарного вектора $(\alpha_1, \dots, \alpha_k)$ назовем правильной, если она имеет вид $\alpha_1 \lambda \alpha_2 \lambda \dots \alpha_{k-1} \lambda \alpha_k$. (при условии, что $\lambda \notin A_{исх}$) либо $\alpha_1^* \alpha_2^* \dots \alpha_{k-1}^* \alpha_k^*$, где $*$ — специальный символ-разделитель, не входящий в $A_{исх}$. Для любого вектора $V_{исх}$ над $A_{исх}$ машина T либо работает бесконечно, либо перерабатывает его в совокупность слов (разделенных пробелами) в алфавите, который назовем алфавитом результатов и обозначим $A_{рез}$; $A_{исх}$ и $A_{рез}$ могут пересекаться и даже совпадать. В ходе работы на ленте могут появляться символы, не входящие в $A_{исх}$ и $A_{рез}$ и образующие промежуточный алфавит $A_{пр}$ (содержащий, в частности, разделитель). Таким образом, алфавит ленты $A = A_{исх} \cup A_{рез} \cup A_{пр}$. В простейшем случае $A_{исх} = A_{рез}$ и $A = A_{исх} \cup \{\lambda\}$.

Пусть f — функция, отображающая множество векторов над $A_{исх}$ в множество векторов над $A_{рез}$. Машина T правильно вычисляет функцию f , если: 1) для любых V и W , таких, что $f(V) = W$, $q_1 V^* \Rightarrow q_z W^*$, где V^* и W^* — правильные записи V и W соответственно; 2) для любого V , такого, что $f(V)$ не определена, машина T , запущенная в стандартной начальной конфигурации $q_1 V^*$, работает бесконечно. Если для f существует машина T , которая ее правильно вычисляет, функция f называется **правильно вычисляемой по Тьюрингу**.

С другой стороны, всякой правильно вычисляющей машине Тьюринга, т. е. машине, которая, начав со стандартной начальной конфигурации $q_1 \alpha$, может остановиться только в стандартной заключительной конфигурации $q_z \beta$, можно поставить в соответствие вычисляемую ей функцию. Две машины

Тьюринга с одинаковым алфавитом $A_{исх}$ будет называть эквивалентными, если они вычисляют одну и ту же функцию. В частности, машины из примеров 1 а) б) эквивалентны, так как они вычисляют одну и ту же нигде не определенную (пустую) функцию.

Пример 2.

Если машина T содержит команды $q_i a_j \rightarrow q'_i a'_j E$ и $q'_i a'_j \rightarrow q''_i a''_j d_k$, то, заменив эти две команды командой $q_i a_j \rightarrow q''_i a''_j d_k$, получим машину T' , эквивалентную T . Путем таких преобразований можно в машине T убрать все команды, содержащие E , для случая, когда q'_i — незаключительное состояние, при этом может сократиться число состояний (некоторые q'_i не войдут в правые части новых команд и станут недостижимыми из q_i). Если q'_i — заключительное состояние, то, введя новое заключительное состояние q_{n+1} и заменив команду $q_i a_j \rightarrow q'_i a'_j E$ на $m+1$ команду $q_i a_j \rightarrow q'_i a'_j R$, $q'_i a_1 \rightarrow q_{n+1} a_1 L, \dots, q'_i a_m \rightarrow q_{n+1} a_m L$ (m — число букв в A), получим машину, также эквивалентную T . Таким образом, для любой машины T существует эквивалентная ей машина, не содержащая в командах E ; поэтому можно рассматривать машины, головки которых на каждом шаге движутся.

Определения, связанные с вычислением функций, заданных на словарных векторах, даны с явным «запасом общности» и имеют в виду переработку нечисловых объектов. В дальнейшем это понадобится; однако в ближайшее время будут рассматриваться числовые функции, точнее, функции, отображающие N в N . Договоримся представлять натуральные числа в единичном (унарном) коде, т.е. для всех числовых функций $A_{исх} == \{1\}$ либо $A_{исх} == \{1, *\}$ и число x представляется словом $1 \dots 1 == 1^x$, состоящим из x единиц. Таким образом, числовая функция $f(x_1, \dots, x_n)$ правильно вычислима по Тьюрингу, если существует машина T , такая, что $q_1 1^{x_1} * 1^{x_2} * \dots * 1^{x_n} \Rightarrow_T q_z 1^y$, когда $f(x_1, \dots, x_n) = y$, и T работает бесконечно, начиная с $q_1 1^{x_1} * 1^{x_2} * \dots * 1^{x_n}$, когда $f(x_1, \dots, x_n)$ не определена.

Начав с общих определений абстрактных машин, мы затем ввели при определении вычислений на этих машинах ряд ограничений, связанных с правильной записью, правильным вычислением, представлением чисел в весьма неэкономном единичном коде и т.д. Не теряется ли при этом общность? Действительно, возможны другие определения вычисления: например, можно в заключительной конфигурации допускать символы из промежуточного алфавита $A_{пр}$, а результатом считать слово в $A_{рез}$, которое получится, если символы из $A_{пр}$ выкинуть и «сдвинуть» оставшиеся куски. В частности, если $A_{рез} = \{1\}$, то результатом будет число, равное числу единиц на ленте в заключительной конфигурации. Оказывается, что потери общности не происходит; в частности, если функция вычислима в последнем смысле, то она правильно вычислима. Не будем заниматься детальным доказательством этого утверждения в общем виде, однако проиллюстрируем его справедливость на примерах (см. далее пример).

ПРИМЕРЫ МАШИН

Обычно прилагательное «правильный» будем опускать и говорить просто о функциях, *вычислимых по Тьюрингу*.

Пример 3.

а. Сложение. Во введенном ранее представлении чисел сложить числа a и b — это значит слово $1^a * 1^b$ переработать в слово 1^{a+b} , т.е. удалить разделитель $*$ и сдвинуть одно из слагаемых, скажем первое, к другому. Это преобразование осуществляет машина T_+ с четырьмя состояниями и следующей системой команд (первая команда введена для случая, когда $a=0$ и исходное слово имеет вид $*1^b$):

$$q_1 * \rightarrow q_2 \lambda R;$$

$$q_1 1 \rightarrow q_2 \lambda R;$$

$$q_2 1 \rightarrow q_2 1 R;$$

$$q_2 * \rightarrow q_3 1 L;$$

$$q_3 1 \rightarrow q_3 1 R;$$

$$q_3 \lambda \rightarrow q_2 \lambda R;$$

В этой системе команд перечислены не все сочетания состояний машины и символов ленты: опущены те из них, которые при стандартной начальной конфигурации никогда не встретятся. Опускать ненужные команды будем в дальнейшем; в таблицах это будет отмечено прочерками. Диаграмма переходов T_+ приведена на рис. 1; заключительное состояние отмечено двойным кружком.

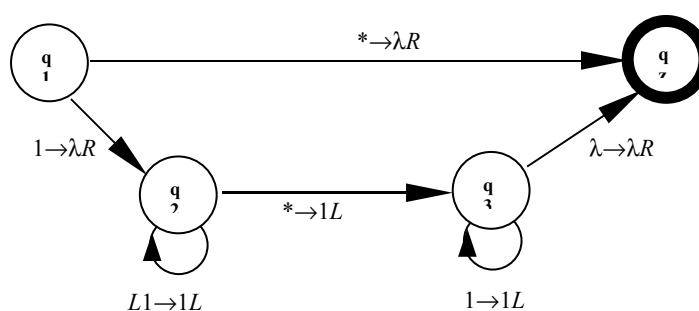


Рис. 1

б. Копирование (перезапись) слова, т. е. переработка слова α в $\alpha*\alpha$. Для чисел эту задачу решает машина $T_{\text{коп}}$, система команд которой приведена в табл. 1.

Таблица 1

| | 1 | λ | * | 0 |
|-------|----------|-----------------|----------|----------|
| q_1 | $q_2 0R$ | $q_2 \lambda R$ | $q_1 *L$ | $q_1 1L$ |
| q_2 | $q_2 1R$ | $q_3 *R$ | $q_3 *R$ | |
| q_3 | $q_3 1R$ | $q_4 1L$ | | |
| q_4 | $q_4 1L$ | | $q_4 *L$ | $q_1 0R$ |

Диаграмма переходов $T_{\text{коп}}$ дана на рис. 2. На этой диаграмме (а также последующих) приняты сокращения:

- 1) если из q_i в q_j ведут два ребра с одинаковой правой частью, то они объединяются в одно ребро, на котором левые части записаны через запятую;
- 2) если символ на ленте не изменяется, то он в правой части команды не пишется. На петле в q_4 использованы одновременно оба сокращения.

Машина $T_{\text{коп}}$ при каждом проходе исходного числа 1^a заменяет левую из его единиц нулем и пишет (в состоянии q_3) одну единицу справа от 1^a в ближайшую пустую клетку. При первом проходе, кроме того, в состоянии q_2 ставится маркер. Таким образом, копия 1^a строится за a проходов. После записи очередной единицы машина переходит в состояние q_4 , которое передвигает головку влево от ближайшего нуля, после чего машина переходит в q_1 и цикл повторяется. Он прерывается, когда q_1 обнаруживает на ленте не единицу, а маркер. Это значит, что все единицы 1^a исчерпаны, т.е. сделано a проходов. Тогда головка возвращается влево в свое исходное положение, заменяя по дороге все нули единицами.

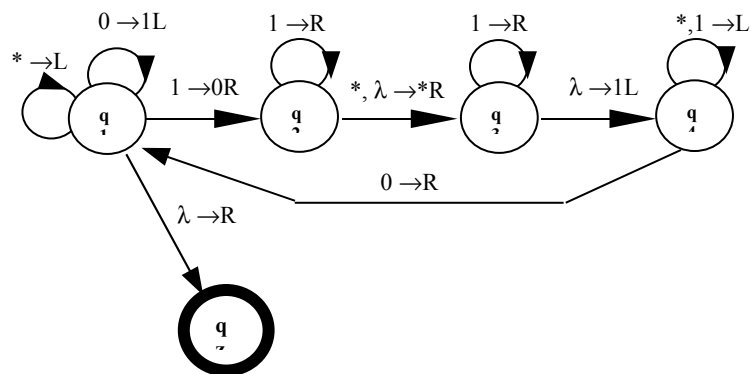


Рис. 2

Операции над машинами Тьюринга.

Работа машины Тьюринга полностью определяется исходными данными и системой команд. Однако для понимания того, как конкретная машина решает данную задачу, как правило, возникает потребность в содержательных пояснениях типа тех, которые приводились для машины $T_{\text{коп}}$. Эти пояснения часто можно сделать более формальными и точными,

если использовать блок-схемы и некоторые операции над машинами Тьюринга.

Композицией двух функций $f_1(x)$ и $f_2(x)$ называется функция $g(x)=f_2(f_1(x))$, которая получается при применении f_2 к результату вычисления f_1 . Для того чтобы $g(x)$ была определена при данном x , необходимо и достаточно, чтобы f_1 была определена на x и f_2 была определена на $f_1(x)$.

Теорема 1. Если $f_1(x)$ и $f_2(x)$ вычислимы по Тьюрингу, то их композиция $g(x)=f_2(f_1(x))$, также вычислима по Тьюрингу.

Пусть T_1 — машина, вычисляющая f_1 , а T_2 — машина, вычисляющая f_2 и множества их состояний соответственно $Q_1=\{q_{11}, \dots, q_{1n_1}\}$ и $Q_2=\{q_{21}, \dots, q_{2n_2}\}$. Построим диаграмму переходов машины T из диаграмм T_1 и T_2 следующим образом: отождествим начальную вершину q_{21} диаграммы машины T_2 с конечной вершиной q_{1z} диаграммы машины T_1 (для систем команд это равносильно тому, что систему команд T_2 приписываем к системе команд T_1 и при этом q_{1z} в командах T_1 заменяем на q_{21}). Получим диаграмму с n_1+n_2-1 состояниями. Начальным состоянием T объявим q_{11} , а заключительным — q_{2z} . Для простоты обозначений будем считать f_1 и f_2 числовыми функциями одной переменной.

Пусть $f_2(f_1(x))$ определена. Тогда $T_1(1^x) = 1^{f_1(x)}$ и $q_{11} \Rightarrow_{T_1} q_{1z} 1^{f_1(x)}$. Машина T пройдет ту же последовательность конфигураций с той разницей, что вместо $q_{1z} 1^{f_1(x)}$ она перейдет в $q_{21} 1^{f_1(x)}$. Эта конфигурация является стандартной начальной конфигурацией для машины T_2 , поэтому $q_{21} 1^{f_1(x)} \Rightarrow_{T_2} q_{2z} 1^{f_2(f_1(x))}$. Но так как все команды T_2 , содержатся в T , то $q_{11} 1^x \Rightarrow_T q_{21} 1^{f_1(x)} \Rightarrow_T q_{2z} 1^{f_2(f_1(x))}$ и, следовательно, $T(1^x) = 1^{f_2(f_1(x))}$. Если же $f_2(f_1(x))$ не определена, то T_1 или T_2 не остановится и, следовательно, машина T не остановится. Итак, машина T вычисляет $f_2(f_1(x))$.

Построенную таким образом машину T будем называть *композицией* машин T_1 и T_2 и обозначать $T_2(T_1)$, а также изображать блок-схемой (рис. 3).

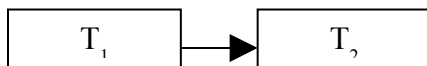


Рис.3

Эта блок-схема имеет более точный смысл, чем изображенная на рис.2 (введение в теорию алгоритмов), так как она всегда предполагает, что исходными данными машины T_2 являются результаты T_1 . При этом они уже «готовы к употреблению», так как благодаря правильной вычислимости (которая существенна при композиции) заключительная конфигурация T_1 легко превращается в стандартную начальную конфигурацию T_2 .

Поскольку машина Тьюринга вектор над $A_{исх}$ воспринимает как слово в алфавите $A_{исх} \cup \{*\}$, определение композиции и теорема 1 остаются в силе, если T_1 и T_2 вычисляют функции от нескольких переменных. Важно лишь,

чтобы данные для T_2 были в обусловленном виде подготовлены машиной T_1 . Это видно на следующем примере.

Пример 4. Машина, диаграмма которой приведена на рис.4, — это машина $T_+(T_{\text{коп}})$. Она вычисляет функцию $f(x)=2x$ для $x \neq 0$; при этом машина $T_{\text{коп}}$ строит двухкомпонентный вектор, а T_+ вычисляет функцию от двух переменных.

Для удобства последующих построений установим следующий важный факт. Оказывается, что для вычисления на машине Тьюринга достаточно, чтобы лента была бесконечной только в одну сторону, например вправо. Такая лента называется *правой полулентой*.

Теорема 2. Любая функция, вычислимая по Тьюрингу, вычислима на машине Тьюринга с правой полулентой; иначе говоря, для любой машины Тьюринга T существует эквивалентная ей машина T_R с правой полулентой. Аналогичная теорема формулируется для левой полуленты.

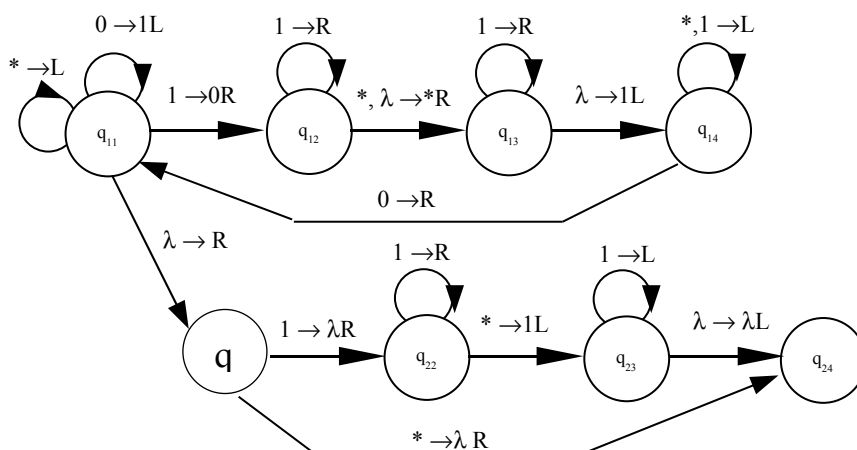


Рис.4

Можно предположить, по крайней мере, две идеи построения такой эквивалентной машины: 1) всякий раз, когда головке прежней машины надо зайти за левый край ленты, новая машина предварительно сдвинет все написанное (вместе с головкой) на клетку вправо; б) лента «перегибается пополам»; при этом ячейки правой половины прежней ленты размещаются, скажем, в нечетных ячейках полуленты, а ячейки левой половины — в четных.

Вычисление предикатов на машинах Тьюринга

Рассмотрим теперь вычисление предикатов на машинах Тьюринга. Машина T вычисляет предикат $P(\alpha)$ (α —слово в $A_{\text{исх}}$), если $T(\alpha) = \bar{\omega}$, где $\bar{\omega} = I$, когда $P(\alpha)$ истинно, и $\bar{\omega} = \perp$, когда $P(\alpha)$ ложно. Если же $P(\alpha)$ не

Так как машина Тьюринга с алфавитом $A_{исх} = \{И, Л\}$ и командами $q_1И \rightarrow q_2ЛЕ$ и $q_1Л \rightarrow q_2ИЕ$ вычисляет отрицание логической переменной, то из вычислимости всюду определенного $P(\alpha)$ следует вычислимость $\neg P(\alpha)$.

Пусть функция $f(\alpha)$ задана описанием: «если $P(\alpha)$ истинно, то $f(\alpha) = g_1(\alpha)$, иначе $f(\alpha) = g_2(\alpha)$ » (под «иначе» имеется в виду «если $P(\alpha)$ ложно»; если же $P(\alpha)$ не определен, то $f(\alpha)$ также не определена). Функция $f(\alpha)$ называется *разветвлением* или *условным переходом* $g_1(\alpha)$ и $g_2(\alpha)$ по условию $P(\alpha)$.

Теорема 3. Если $g_1(\alpha), g_2(\alpha)$ и $P(\alpha)$ вычислимы по Тьюрингу, то разветвление g_1 и g_2 по P также вычислимо.

Пусть T_1 — машина с состояниями $q_{11}, q_{12}, \dots, q_{1n_1}$, и системой команд Σ_1 , вычисляющая g_1 ; T_2 — машина с состояниями $q_{21}, q_{22}, \dots, q_{2n_2}$, и системой команд Σ_2 , вычисляющая g_2 ; Tr вычисляет с восстановлением $P(\alpha)$. Тогда машина T , вычисляющая разветвление g_1 и g_2 по P , — это композиция Tr и машины T_3 , система команд Σ_3 которой имеет следующий вид:

$$\Sigma_3 = \Sigma_1 \cup \Sigma_2 \cup \{q_{31}И \rightarrow \lambda q_{11}R, q_{31}Л \rightarrow \lambda q_{21}R, q_{12}И \rightarrow q_{22}E\}.$$

Первые две из новых команд передают управление системам команд Σ_1 или Σ_2 в зависимости от значения предиката $P(\alpha)$. Третья команда введена для того, чтобы T_3 имела одно заключительное состояние q_{2z} . Отсутствие символа ленты в этой команде означает, что она выполняется при любом символе.

Разветвление встречается в структуре диаграмм переходов многих машин Тьюринга.

Пример 6.

а. В примере 3, а было описано построение машины T^+ для сложения двух чисел. На рис. 6 приведена диаграмма машины T^{++} для сложения n чисел ($n=1, 2, \dots$). Цикл из состояний q_1, q_2, q_3 — это «зацикленная» машина T^+ , в которой заключительное состояние совмещено с начальным.

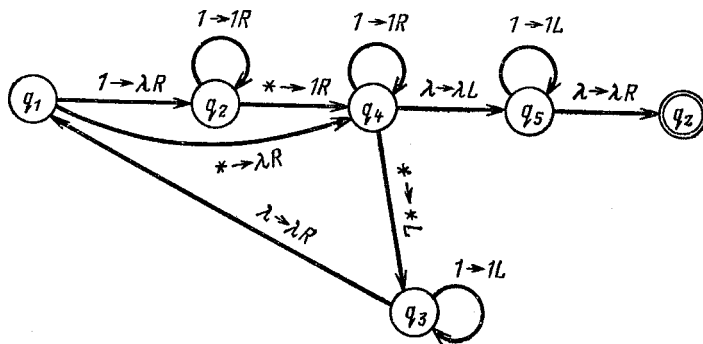


Рис. 6

Сумма, полученная на очередном цикле, является первым слагаемым следующего цикла. Состояние q_4 реализует разветвление. В нем проверяется условие — есть ли второе слагаемое. Если да (о чем говорит наличие маркера *), то происходит переход к следующему циклу; если нет (о чем говорит λ после единиц), то машина выходит из цикла.

б. Пусть машина T с алфавитом A_T и с $\text{Аисх} = \{1\}$ не является правильно вычисляющей и ее заключительная конфигурация стандартна, но может содержать любые символы из A_T (кроме пробелов), а результат интерпретируется как число, равное числу единиц на ленте. Построим для T машину T^{++} , которая работает как T^{++} , а все символы, кроме 1 и λ , она воспринимает как маркеры, т. е. команды для них — те же, что и для *. Тогда T^{++} соберет все единицы в один массив и, следовательно, T^{++} ($T^{++}(\alpha)$) правильно вычисляет функцию, вычисляемую машиной T .

Пример 6, а иллюстрирует важный частный случай разветвления — выход из цикла по условию, хорошо известный в программировании. В словесных описаниях (см. пример 1) и во многих языках программирования (например, в АЛГОЛе) этот случай формулируется так: повторять вычисление f_1 до тех пор, пока истинно условие P ; если P ложно, перейти к вычислению f_2 . Благодаря вычислимости композиции и разветвления словесные описания и язык блок-схем можно сделать вполне точным языком для описания работы машин Тьюринга. Каждый блок — это множество состояний, в котором выделены начальное и заключительное состояния, и система команд. Правильное вычисление для промежуточных блоков не обязательно, поэтому при стыковке блоков важно согласовывать всю конфигурацию. Переход к блоку — это обязательно переход в его начальное состояние. Машина Тьюринга, описываемая блок-схемой, — это объединение состояний и команд, содержащихся во всех блоках. В частности, блоком может быть одно состояние. Блоки, вычисляющие предикаты, обозначим буквой P .

Пример 7.

На рис. 7 приведена блок-схема машины Тьюринга T_x , осуществляющей умножение двух чисел:

$T_x(1^a * 1^b) = 1^{ab}$. Ее заключительное состояние — q_{03} . Блоки реализуют следующие указания и вычисления:

P_1 — вычислить с восстановлением предикат «оба слагаемых больше нуля»;

T_1 — стереть все непустые символы справа;

T_2 — установить головку у ячейки, следующей за маркером *; маркер * заменить на \sim ;

$T_{\text{коп}}$ — см. пример 4,б, в котором команду $q_1\lambda \rightarrow q_2R$, надо заменить на команду $q_1\sim \rightarrow q_2R$;

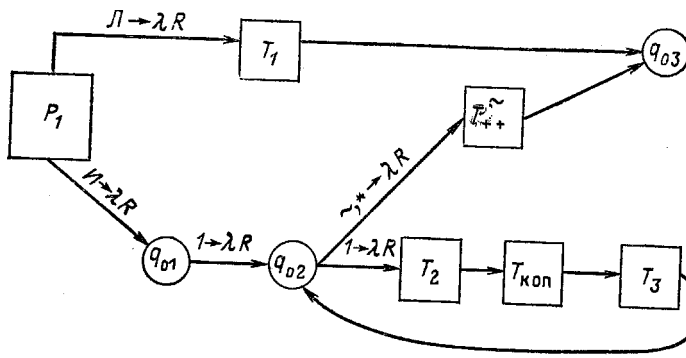


Рис. 7

$T_{\text{кон}}(a-1)$ раз копирует 1^b ; после i -го цикла она останавливается в конфигурации $1^{(a-i-1)} (\sim 1^b)^{i-1} \sim q 1^b * 1^b$,

T_z — вернуть головку к крайнему слева непустому символу. После $(a-1)$ -го цикла этим символом окажется \sim (или $*$, если $a=1$), и происходит выход из цикла и переход к $T_{\sim++}$.

$T_{\sim++}$ работает как T_{++} (см. пример 6, а) с той разницей, что числа, которые она суммирует, разделены двумя видами маркеров: \sim и $*$; для этой цели в нее к командам с маркером $*$ в левой части добавлены такие же команды для маркера \sim .

Аналогично тому, как из машины T_+ была построена T_{++} , можно из T_x построить машину T_{xx} , которая перемножает несколько чисел, и по схеме, аналогичной приведенной на рис. 7, с использованием $T_{\sim xx}$ вместо $T_{\sim++}$ построить машину, осуществляющую возведение в степень. Другой важный пример, который читатель может либо построить, либо найти перевод унарного представления чисел в позиционное — двоичное или десятичное.

Универсальная машина Тьюринга

Построение универсальной машины U

Отметим, что при построении U (как, впрочем, и для многих других машин, описанных в этом параграфе) мы не преследовали целей оптимизации и не жалели ни символов ленты, ни состояний, стремясь к наглядности построения. Нетрудно показать, — а инженер-дискретчик, привыкший к двоичному кодированию, легко в это поверит — что можно построить машину U всего с двумя символами на ленте. **Шеннон установил менее очевидный факт — он построил универсальную машину с двумя состояниями. !В тоже время показано (Боброу и Минский), что универсальная машина с двумя состояниями и двумя символами невозможна.** Вообще в определенных пределах уменьшение числа символов U ведет к увеличению числа состояний, и наоборот, Подробная сводка результатов о минимальных универсальных машинах приведена в [Минский М. Вычисления и автоматы. М.: Мир, 1971].

Существование универсальной машины Тьюринга означает, что систему команд Σ_T любой машины T можно интерпретировать двояким образом: либо как описание работы конкретного устройства машины T , либо как программу для универсальной машины U . Для современного инженера, проектирующего систему управления, это обстоятельство вполне естественно. Он хорошо знает, что любой алгоритм управления может быть

реализован либо аппаратурно – построением соответствующей схемы, либо программно – написанием программы для универсальной управляющей ЭВМ.

Однако важно сознавать, что сама идея универсального алгоритмического устройства совершенно не связана с развитием современных технических средств его реализации (электроники, физики твердого тела и т. д.) и является не техническим, а математическим фактом, описываемым в абстрактных математических терминах, не зависящих от технических средств, и к тому же опирающимся на крайне малое количество весьма элементарных исходных понятий.

Основополагающие работы по теории алгоритмов (и, в частности, работы Тьюринга) появились в 30-х годах, до создания современных ЭВМ.

Указанная двоякая интерпретация сохраняет на абстрактном уровне основные плюсы и минусы двух вариантов инженерной реализации. Конкретная машина T работает гораздо быстрее; кроме того, управляющее устройство машины U довольно громоздко (т.е. велико число состояний и команд). Однако его сложность постоянна, и, будучи раз построено, оно годится для реализации сколь угодно больших алгоритмов, понадобится лишь больший объем ленты, которую естественно считать более дешевой и более просто устроенной, чем управляющее устройство. Кроме того, при смене алгоритма не понадобится строить новых устройств; нужно лишь написать новую программу.

Тезис Тьюринга.

До сих пор нам удавалось для всех процедур, претендующих на алгоритмичность, т. е. конструктивных процедур, строить реализующие их машины Тьюринга. Будет ли это удаваться всегда? Утвердительный ответ на этот вопрос содержится в тезисе Тьюринга, который формулируется так: **«Всякий алгоритм может быть реализован машиной Тьюринга».**

Доказать тезис Тьюринга нельзя, поскольку само понятие алгоритма (или эффективной процедуры) является неточным. Это не теорема и не постулат математической теории, а утверждение, которое связывает теорию с теми объектами, для описания которых она создана. По своему характеру тезис Тьюринга напоминает гипотезы физики об адекватности математических моделей физическим явлениям и процессам. Подтверждением тезиса Тьюринга является, во-первых, математическая практика, а во-вторых, то обстоятельство (уже отмечавшееся в конце §1), что описание алгоритма в терминах любой другой известной алгоритмической модели может быть сведено к его описанию в виде машины Тьюринга.

Тезис Тьюринга позволяет, с одной стороны, заменить неточные утверждения о существовании эффективных процедур (алгоритмов) точными утверждениями о существовании машин Тьюринга, а с другой стороны, утверждения о несуществовании машин Тьюринга истолковывать как утверждения о несуществовании алгоритмов вообще. Однако не следует понимать тезис Тьюринга в том смысле, что вся теория алгоритмов может быть сведена к теории машин Тьюринга. Например, в быстро развивающейся сейчас теории сложности алгоритмов (которая рассматривает сравнительную сложность алгоритмов по памяти, числу действий и т. д.) результаты, верные в рамках одной алгоритмической модели (скажем, о числе действий,

необходимых для вычисления данной функции), могут оказаться неверными в другой модели.

Проблема остановки.

В числе общих требований, предъявляемых к алгоритмам (см. §1), упоминалось требование результативности. Наиболее радикальной формулировкой здесь было бы требование, чтобы по любому алгоритму A и данным α можно было определить, приведет ли работа A при исходных данных α к результату или нет. Иначе говоря, нужно построить алгоритм B , такой, что $B(A, \alpha) = И$, если $A(\alpha)$ дает результат, и $B(A, \alpha) = Л$, если $A(\alpha)$ не дает результата. В силу тезиса Тьюринга эту задачу можно сформулировать как задачу о построении машины Тьюринга; построить машину T_0 , такую, что для любой машины Тьюринга T и любых исходных данных α для машины T $T_0(\Sigma_T, \alpha) = И$, если машина $T(\alpha)$ останавливается, и $T_0(\Sigma_T, \alpha) = Л$, если машина $T(\alpha)$ не останавливается.

Эта задача называется *проблемой остановки*. Ее формулировка несколько напоминает задачу о построении универсальной машины и, в частности, также предполагает выбор подходящего кодирования Σ_T и α в алфавите машины T_0 . Однако в данном случае никакое кодирование не приводит к успеху.

Теорема (проблема остановки) Не существует машины Тьюринга T_0 , решающей проблему остановки для произвольной машины Тьюринга T .

Предположим, что машина T_0 существует. Для определенности будем считать, что маркером между Σ_T и α на ленте машины T_0 служит $*$. Построим машину $T_1(\Sigma_T) = T_0(T_{\text{коп}}(\Sigma_T))$. Исходными данными машины T_1 являются системы команд (точнее, их коды) любой машины T . Запись Σ_T на ленте машины T_1 преобразует в $\Sigma_T * \Sigma_T$ (машина $T_{\text{коп}}$ для чисел описана в примере 4, б), а затем работает как машина T_0 . Таким образом, T_1 также решает проблему остановки для любой машины T , но только в том случае, когда на ленте T в качестве данных α_T написана ее собственная система команд Σ_T . Иначе говоря, $T_1(\Sigma_T) = И$, если машина $T(\Sigma_T)$ останавливается, и $T_1(\Sigma_T) = Л$, если машина $T(\Sigma_T)$ не останавливается. Пусть q_{1n} — заключительное состояние T_1 . Добавим к системе команд T_1 одно состояние $q_{1,n+1}$, объявив его заключительным, и m команд (m — число символов T_1) $q_{1n}L \rightarrow q_{1,n+1}E$, $q_{1n}a_j \rightarrow q_{1,n}R$ для любого a_j , (в том числе $И$), кроме L . Получим машину $T_1'(\Sigma_T)$, которая останавливается, если T не останавливается, и не останавливается, если T останавливается. Запишем теперь на ленте машины T_1' ее собственную систему команд $\Sigma_{T_1'}$. Тогда T_1' остановится, если она не останавливается, и не остановится, если она останавливается. Очевидно, такая машина T_1'

невозможна. Но поскольку она получена из T_0 вполне конструктивными, не вызывающими сомнений средствами и при этом никак не связана с конкретной структурой машины T_0 , остается заключить, что никакая машина T_0 , решающая проблему остановки, невозможна.

В силу тезиса Тьюринга невозможность построения машины Тьюринга означает отсутствие алгоритма решения данной проблемы. Поэтому полученная теорема дает первый пример *алгоритмически неразрешимой проблемы*, а именно, алгоритмически неразрешимой оказывается проблема остановки для машин Тьюринга, т. е. проблема определения результативности алгоритмов.

При истолковании утверждений, связанных с алгоритмической неразрешимостью, следует иметь в виду следующее важное обстоятельство. В таких утверждениях речь идет об отсутствии единого алгоритма, решающего данную проблему; при этом вовсе не исключается возможность решения этой проблемы в частных случаях, но различными средствами для каждого случая. В частности, теорема (проблема остановки) не исключает того, что для отдельных классов машин Тьюринга проблема остановки может быть решена'. Например, она решается для всех машин, приведенных в примерах 2 - 8. Поэтому неразрешимость общей проблемы остановки вовсе не снимает необходимости доказывать сходимость предлагаемых алгоритмов, а лишь показывает, что поиск таких доказательств нельзя полностью автоматизировать.

Неразрешимость проблемы остановки можно интерпретировать как несуществование общего алгоритма для отладки программ, точнее, алгоритма, который по тексту любой программы и данным для нее определял бы, заикнется ли программа на этих данных или нет. Если учесть сделанное ранее замечание, такая интерпретация не противоречит тому эмпирическому факту, что большинство программ, в конце концов, удается отладить, т. е. установить наличие заикливания, найти его причину и устранить ее. При этом решающую роль играют опыт и искусство программиста.

Лекция 15

РЕКУРСИВНЫЕ ФУНКЦИИ

Всякий алгоритм однозначно ставит в соответствие исходным данным (в случае, если он определен на них) результат. Поэтому с каждым алгоритмом однозначно связана функция, которую он вычисляет. Верно ли обратное: для всякой ли функции существует вычисляющий ее алгоритм? Исследование проблемы остановки для машин Тьюринга показывает, что нет: для предиката $P(T, \alpha)$, истинного, если и только если машина Тьюринга T останавливается при исходных данных α , алгоритма его вычисления не

существует. Возникает вопрос: для каких функций алгоритмы существуют? Как описать такие алгоритмические, эффективно вычислимые функции?

Исследование этих вопросов привело к созданию в 30-х годах нашего века *теории рекурсивных функций*. В этой теории, как и вообще в теории алгоритмов, принят конструктивный, финитный подход, основной чертой которого является то, что все множество исследуемых объектов (в данном случае функций) строится из конечного числа исходных объектов базиса – с помощью простых операций, эффективная выполнимость которых достаточно очевидна. Операции над функциями в дальнейшем будем называть операторами.

Примитивно-рекурсивные функции. Определение и примеры. Займемся теперь конкретным выбором средств, с помощью которых будут строиться вычислимые функции. Очевидно, что к вычислимым функциям следует отнести все константы, т. е. 0 и все натуральные числа 1, 2... Однако в прямом включении бесконечного множества констант в базис нет необходимости.

Опр. 1. Достаточно одной константы 0 и *функции следования* $f(x)=x+1$, которую иногда обозначают x' , чтобы получить весь натуральный ряд.

Опр. 2. Кроме того, в базис включим семейство $\{I_m^n\}$ *функций тождества* (или введения фиктивных переменных):

$$I_m^n(x_1, \dots, x_n) = x_m \quad (m \leq n).$$

Весьма мощным средством получения новых функций из уже имеющихся является *суперпозиция*, знакомая нам. *Суперпозицией* называлась любая подстановка функций в функции. Здесь ей для большей обзорности удобно придать стандартный вид.

Опр. 3. *Оператором суперпозиции* S_m^n называется подстановка в функцию от m переменных m функций от n одних и тех же переменных. Она дает новую функцию от n переменных. Например, для $h(x_1, \dots, x_m), g_1(x_1, \dots, x_n), \dots, g_m(x_1, \dots, x_n)$

$$S_m^n(h, g_1, \dots, g_m) = h(g_1(x_1, \dots, x_n), \dots, g_m(x_1, \dots, x_n)) = f(x_1, \dots, x_n).$$

Это определение порождает семейство операторов суперпозиции $\{S_m^n\}$. Благодаря функциям тождества стандартизация суперпозиции не уменьшает ее возможностей: любую подстановку функций в функции можно выразить через S_m^n и I_m^n .

Например, $f(x_1, x_2) = h(g_1(x_1, x_2), g_2(x_1))$ в стандартном виде запишется как $f(x_1, x_2) = S_2^2(h(x_1, x_2), g_1(x_1, x_2), S_1^1(I_1^1(x_1, x_2))), g_2(x_1), g_3(x_1))$, где g_3 — любая функция от x_1 . В свою очередь, используя подстановку и функции тождества, можно переставлять и отождествлять аргументы в функции:

$$\begin{aligned} f(x_2, x_1, x_3, \dots, x_n) &= f(I_2^2(x_1, x_2), I_1^1(x_1, x_2), x_3, \dots, x_n); \\ f(x_1, x_1, x_3, \dots, x_n) &= f(I_1^2(x_1, x_2), I_1^1(x_1, x_2), x_3, \dots, x_n). \end{aligned}$$

Таким образом, если заданы функции Γ_m^n и операторы S_m^n , то можно считать заданными всевозможные операторы подстановки функций в функции, а также переименования, перестановки и отождествления переменных.

Еще одно семейство операторов, которое здесь понадобится, – это операторы примитивной рекурсии.

Опр. 4. Оператор примитивной рекурсии R_n определяет $(n+1)$ -местную функцию f через n -местную функцию g и $(n+2)$ -местную функцию h следующим образом:

$$\begin{aligned} f(x_1, \dots, x_n, 0) &= g(x_1, \dots, x_n); \\ f(x_1, \dots, x_n, y+1) &= h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)). \end{aligned} \quad (1)$$

Пара равенств (1) называется *схемой примитивной рекурсии*. Тот факт, что функция f определена схемой (1), выражается равенством $f(x_1, \dots, x_n, y) = R_n(g, h)$. В случае, когда $n=0$, т. е. определяемая функция f является одно-местной, схема (1) принимает более простой вид:

$$f(0) = c; \quad f(y+1) = h(y, f(y)). \quad (2)$$

где c — константа.

Схемы (1) и (2) определяют f рекурсивно не только через другие функции g и h , но и через значения f в предшествующих точках: значение f в точке $y+1$ зависит от значения f в точке y . Для вычисления $f(x_1, \dots, x_n, k)$ понадобится $k+1$ вычислений по схеме (1) - для $y=0, 1, \dots, k$.

Пример такого определения функции приводился для функции $n!$; здесь оно будет рассмотрено более подробно. Существенным в операторе примитивной рекурсии является то, что независимо от числа переменных в f рекурсия ведется только по одной переменной y ; остальные n переменных x_1, \dots, x_n на момент применения схем (1), (2) зафиксированы и играют роль параметров.

Опр. 5. Функция называется *примитивно-рекурсивной*, если она может быть получена из константы 0, функции x' и функций Γ_m^n с помощью конечного числа применений операторов суперпозиции и примитивной рекурсии. Этому определению можно придать более формальный индуктивный вид.

1. Функции 0, x' и Γ_m^n для всех натуральных n, m , где $m \leq n$, являются примитивно-рекурсивными.
2. Если $h(x_1, \dots, x_m), g_1(x_1, \dots, x_n), \dots, g_m(x_1, \dots, x_n)$ - примитивно-рекурсивные функции, то $S_m^n(h, g_1, \dots, g_m)$ - примитивно-рекурсивные функции для любых натуральных n, m .
3. Если $g_1(x_1, \dots, x_n)$ и $h(x_1, \dots, x_n, y, z)$ — примитивно-рекурсивные функции, то
 $R_n(g, h)$ - примитивно-рекурсивная функция.
4. Других примитивно-рекурсивных функции нет. Из такого

индуктивного описания нетрудно извлечь процедуру, порождающую все примитивно-рекурсивные функции.

Лекция 16

ПРИМЕРЫ ПРИМИТИВНО-РЕКУРСИВНЫХ ФУНКЦИЙ

Пример 1. (сложение, умножение, возведение в степень).

1. Сложение $f_+(x, y) = x + y$ примитивно-рекурсивно:

$$f_+(x, 0) = x = I_1^1(x);$$

$$f_+(x, y+1) = f_+(x, y) + 1 = (f_+(x, y))'.$$

Таким образом, $f_+(x, y) = R_1(I_1^1(x), h(x, y, z))$, где $h(x, y, z) = z' = z + 1$.

2. Умножение $f_\times(x, y) = xy$ примитивно-рекурсивно:

$$f_\times(x, 0) = 0;$$

$$f_\times(x, y+1) = f_\times(x, y) + x = f_+(x, f_\times(x, y)).$$

3. Возведение в степень $f_{\text{exp}}(x, y) = x^y$ примитивно-рекурсивно:

$$f_{\text{exp}}(x, 0) = 1;$$

$$f_{\text{exp}}(x, y+1) = x^y \cdot x = f_\times(x, f_{\text{exp}}(x, y)).$$

Определим функцию $x \div y$ (арифметическое или урезанное вычитание) следующим образом:

Пример 2. (вычитание). Примитивно-рекурсивными являются следующие функции:

1) $f(x) = x \div 1$, определяемая схемой:

$$f(0) = 0 \div 1 = 0;$$

$$f(y+1) = y;$$

2) $f(x, y) = x \div y$, определяемая схемой:

$$f(x, 0) = x;$$

$$f(x, y+1) = x \div (y+1) = (x \div y) \div 1 = f(x, y) \div 1$$

(для определения функции из п.2 использована функция из п.1);

3) $f(x, y) = |x - y| = (x \div y) + (y \div x)$;

4) функция знака (сигнум)

0, если $x = 0$;

$sg(x) = 1$, если $x \neq 0$;

ее схема имеет вид:

$$sg(0) = 0;$$

$$sg(x+1) = 1;$$

$$5) \min(x, y) = x \div (x \div y);$$

$$6) \max(x, y) = y + (x \div y).$$

С помощью функции sg (сигнум) из примера 2 и ее отрицания $\bar{sg}(x) = 1 \div sg(x)$ построим примитивно-рекурсивное описание функций, связанных с делением.

Пример 3. (деление)

а. Функция $r(x, y)$ - остаток от деления y на x :

$$r(x, 0) = 0;$$

$$r(x, y+1) = (r(x, y) + 1) sg(|x - (r(x, y) + 1)|).$$

Смысл второй строки определения в следующем: если $y+1$ не делится на x , то $sg(|x \div (r(x, y) + 1)|) = 1$ и $r(x, y+1) = r(x, y) + 1$; если же $y+1$ делится на x , то $r(x, y+1) = sg(|x - (r(x, y) + 1)|) = 0$.

б. Функция $q(x, y) = [y/x]$ — частное от деления y на x , т. е. целая часть дроби y/x :

$$q(x, 0) = 0;$$

$$q(x, y+1) = q(x, y) + \bar{sg}(|x - (r(x, y) + 1)|).$$

Второе слагаемое, как и в случае $r(x, y)$, зависит от делимости $y+1$ на x . Если $y+1$ делится на x , то $q(x, y+1)$ на единицу больше, чем $q(x, y)$; если нет, то $q(x, y+1) = q(x, y)$.

В рекурсивных описаниях функций примера 3 отчетливо видны логические действия: проверка условия (делимости $y+1$ на x) и выбор дальнейшего хода вычислений в зависимости от истинности условия. Займемся теперь более подробным выяснением того, какие возможности для логических операций имеются в рамках примитивно-рекурсивных функций.

Из функций примера 2 легко получается примитивная рекурсивность «арифметизованных» логических функций, т. е. числовых функций, которые на множестве $\{0, 1\}$ ведут себя как логические функции. Действительно, если $x, y \in \{0, 1\}$, то

$$\bar{x} = 1 \div x;$$

$$x \vee y = \max(x, y);$$

$$x \& y = \min(x, y).$$

Из функциональной полноты (см. §) этого множества функций и того, что суперпозиция является примитивно-рекурсивным оператором (см. далее), следует примитивная рекурсивность всех логических функций.

Отношение $R(x_1, \dots, x_n)$ называется примитивно-рекурсивным, если примитивно-рекурсивна его характеристическая функция χ_R :

$$\chi_R(x_1, \dots, x_n) = \begin{cases} 1, & \text{если } R(x_1, \dots, x_n) \text{ выполняется;} \\ 0 & \text{в противном случае.} \end{cases}$$

Ввиду взаимно однозначного соответствия между отношениями и предикатами χ_R будет характеристической функцией и для соответствующего предиката.

Напомним, что сам предикат принимает логические значения «И» и «Л», с которыми нельзя производить арифметических действий, даже когда эти значения изображаются числами 0 и 1 (см. §). Поэтому следует проводить различие между предикатами и их характеристическими функциями. В алгоритмических языках типа АЛГОЛ предикат будет принимать значения true и false, а его характеристическая функция – значения 0 и 1.

Опр.5. Предикат называется *примитивно-рекурсивным*, если его характеристическая функция примитивно-рекурсивна. В силу соотношений (3), если предикаты $P_1 \dots, P_k$ примитивно-рекурсивны, то и любой предикат, полученный из них с помощью логических операций, примитивно-рекурсивен.

Пример 4. (предикаты и отношения)

а. Предикат $Pd_n(x)$ « x делится на n » примитивно-рекурсивен для любого n :

$$\chi_{Pd_n}(x) = \lceil \text{sg}(r(n,x)).$$

б. Предикат $Pd_{n,m}(x)$ « x делится на m и на n » примитивно-рекурсивен для любых m, n , так как $P_{n,m}(x) = P_m(x) \& P_n(x)$.

в. Отношение $x_1 > x_2$ примитивно-рекурсивно:

$$\chi_{>}(x_1, x_2) = \text{sg}(x_1 \div x_2).$$

г. Если $f(x)$ и $g(x)$ примитивно-рекурсивны, то предикат « $f(x)=g(x)$ » примитивно-рекурсивен, так как его функция χ имеет вид:

$$\chi(x) = \lceil \text{sg}(|f(x) - g(x)|).$$

Лекция 17

Примитивно-рекурсивные операторы.

Оператор называется примитивно-рекурсивным (сокращенно ПР-оператором), если он сохраняет примитивную рекурсивность функций, т. е., если результат его применения к примитивно-рекурсивным функциям дает снова примитивно-рекурсивную функцию. Операторы S_m^n и R_n являются ПР-операторами по определению. Рассмотрим другие ПР-операторы. Их использование позволяет существенно сократить примитивно-рекурсивные описания функций.

Мы встречались с оператором условного перехода (обозначим его здесь B), который по функциям $g_1(x_1, \dots, x_n)$, $g_2(x_1, \dots, x_n)$ и предикату $P(x_1, \dots, x_n)$ строит функцию $f(x_1, \dots, x_n) = B(g_1, g_2, P)$:

$$f(x_1, \dots, x_n) = \begin{cases} g_1(x_1, \dots, x_n), & \text{if } P(x_1, \dots, x_n) \text{ истинно;} \\ g_2(x_1, \dots, x_n), & \text{if } P(x_1, \dots, x_n) \text{ ложно.} \end{cases} \quad (1)$$

Теорема 1. B вычислим по Тьюрингу.

Примитивная рекурсивность B видна из следующего соотношения, эквивалентного (1):

$$f(x_1, \dots, x_n) = g_1(x_1, \dots, x_n)\chi_P(x_1, \dots, x_n) + g_2(x_1, \dots, x_n)\chi_{\bar{P}}(x_1, \dots, x_n)$$

Обобщение оператора B на случай многозначного перехода по предикатам P_1, \dots, P_k , из которых истинен всегда один и только один предикат: $f(x_1, \dots, x_n) = B(g_1, \dots, g_k, P_1, \dots, P_k)$, также примитивно-рекурсивно, поскольку

$$f(x_1, \dots, x_n) = g_1\chi_{P_1} + \dots + g_k\chi_{P_k}$$

(отметим, что это соотношение определяет B и в том случае, когда ни один из P_1, \dots, P_k не истинен: B при этом равно нулю).

С помощью оператора B удобно задавать функции, определенные на конечных множествах. Правда, B , как и любой другой ПР-оператор, всегда определяет функцию на всем натуральном ряде, т. е. производит доопределение функции вне области задания. **Например**, функцию f , определенную на множестве $\{1, 2, 6, 17\}$ равенствами $f(1)=5, f(2)=1, f(6)=0, f(17)=8$, с помощью оператора B можно описать так:

$$f(x) = \begin{cases} 5, & \text{if } x=1; \\ 1, & \text{if } x=2; \\ 0, & \text{if } x=6; \\ 8, & \text{в остальных случаях.} \end{cases}$$

Такое описание полагает $f(x)=8$ вне исходной области задания.

Пусть $f(x_1, \dots, x_n, y)$ —функция от $(n+1)$ -й переменной. Хорошо известные операции суммирования $\Sigma_{y<z}$ и перемножения $\Pi_{y<z}$ по переменной y с пределом z — это операторы, которые из функции $f(x_1, \dots, x_n, y)$

порождают новые функции $g(x_1, \dots, x_n, z) = \sum_{y < z} f(x_1, \dots, x_n, y)$ и $h(x_1, \dots, x_n, z) = \prod_{y < z} f(x_1, \dots, x_n, y)$. Покажем их примитивную рекурсивность:

$$g(x_1, \dots, x_n, 0) = 0 \text{ (по определению);}$$

$$g(x_1, \dots, x_n, z+1) = g(x_1, \dots, x_n, z) + f(x_1, \dots, x_n, z);$$

$$h(x_1, \dots, x_n, 0) = 1;$$

$$h(x_1, \dots, x_n, z+1) = h(x_1, \dots, x_n, z) f(x_1, \dots, x_n, z).$$

Таким образом, g и h могут быть определены по схеме "примитивной рекурсии с использованием сложения и умножения, примитивная рекурсивность которых доказана, а также функции f . Следовательно, g и h примитивно-рекурсивны, если f примитивно-рекурсивна.

С помощью $\sum_{y < z}$ и $\prod_{y < z}$ нетрудно показать, что операторы $\sum_{y=k}^z$ и $\prod_{y=k}^z$ и (т. е. суммирование и умножение от k до z) — так же ПР-операторы.

Рассмотрим теперь оператор, играющий важную роль в теории рекурсивных функций, — *ограниченный оператор наименьшего числа* (μ -оператор), называемый также *ограниченным оператором минимизации*, который применяется к предикатам и определяется как:

$$\mu_{y_{y < z}} P(x_1, \dots, x_n, y) = \begin{cases} \text{наименьшему } y \leq z, \text{ такому, что } P(x_1, \dots, x_n, y) = \text{И,} \\ \text{если такой } y \text{ существует;} \\ z \text{ в противном случае.} \end{cases}$$

Из предиката $P(x_1, \dots, x_n, y)$ с помощью оператора $\mu_{y_{y < z}}$ получается функция $f(x_1, \dots, x_n, z)$. Второй случай в определении μ добавлен для того, чтобы f была всюду определена.

Пример 1. Пусть $P(x_1, x_2, y) = Pd_{x_1, x_2}(y)$ (пример). Тогда $\mu_{y_{y < z}} P(x_1, x_2, y)$ равен наименьшему общему кратному (НОК) x_1 и x_2 , если $z \geq \text{НОК}$, и равен z , если $z < \text{НОК}$.

Ограниченный μ -оператор примитивно-рекурсивен:

$$\mu_{y_{y < z}} P(x_1, \dots, x_n, y) = \sum_{i=0}^z \prod_{j=0}^i (1 - \chi_P(x_1, \dots, x_n, j)).$$

Ограничение z в ограниченном μ -операторе дает гарантию окончания вычислений, поскольку оно оценивает сверху число вычислений предиката P . Возможность оценить сверху количество вычислений является существенной особенностью примитивно-рекурсивных функций. Уже отмечалось, что если $f(x_1, \dots, x_n, k) = R_n(g, h)$, то для вычисления $f(x_1, \dots, x_n, k)$ понадобится $k+1$ вычислений по схеме (?5.2); одно вычисление g и k вычислений h . Правда, каждое из них может, в свою очередь, состоять из некоторого количества вычислений функций, входящих в определение g и h ; но в силу конечности общего числа операторов S_m^n и R_n , использованных для построения f из базисных функций $0, x, ' \text{ и } I_m^n$, для любого k можно оценить количество элементарных действий (т. е. вычислений базисных функций),

необходимых для вычисления $f(x_1, \dots, x_n, k)$. В дальнейшем будет показано, что неограниченный μ -оператор не является примитивно-рекурсивным.

μ -оператор (как ограниченный, так и неограниченный) является удобным средством для построения обратных функций. Действительно, функция $g(x) = \mu y (f(y) = x)$ («наименьший y , такой, что $f(y) = x$ ») является обратной к функции $f(x)$. Поэтому в применении к одноместным функциям μ -оператор иногда называют оператором обращения.

Пример 2.

а. С помощью ограниченного μ -оператора определим деление, точнее функцию $[z/x]$ (частное от деления z на x ;—см. пример), как функцию, обратную умножению:

$$[z/x] = \mu y_{y <= z} (x(y+1) > z).$$

Целая часть от деления — функция, «не совсем обратная» умножению; точнее, обратная ему только в тех точках, где z делится нацело на x . Поэтому в ее описании используется не предикат равенства (как в определении обратной функции), а предикат $>$.

б. Целая часть \sqrt{x} — функция $[\sqrt{x}]$ — примитивно-рекурсивная, так как

$$[\sqrt{x}] = \mu y_{y <= x} ((y+1)^2 > x).$$

в. $[\log_k x] = \mu y_{y <= x} ((k^{y+1} > x))$, следовательно, целая часть логарифма по любому целому основанию k примитивно-рекурсивна.

Еще один ПР-оператор — это оператор совместной или одновременной рекурсии, точнее, целое семейство операторов $\{R_{nk}\}$. С помощью R_{nk} строится рекурсивное описание сразу нескольких функций f_1, \dots, f_k от $(n+1)$ -й переменной, причем значение каждой функции $y+1$ зависит от значения всех функций в точке y :

$$f_1(x_1, \dots, x_n, 0) = g_1(x_1, \dots, x_n);$$

.....

$$f_k(x_1, \dots, x_n, 0) = g_k(x_1, \dots, x_n);$$

$$f_1(x_1, \dots, x_n, y+1) = h_1(x_1, \dots, x_n, y, f_1(x_1, \dots, x_n, y), \dots, f_k(x_1, \dots, x_n, y));$$

.....

$$f_k(x_1, \dots, x_n, y+1) = h_k(x_1, \dots, x_n, y, f_1(x_1, \dots, x_n, y), \dots, f_k(x_1, \dots, x_n, y)).$$

По существу совместная рекурсия дает рекурсивное описание функции-вектора (f_1, \dots, f_k) . Для того чтобы доказать примитивную рекурсивность совместной рекурсии, нужно закодировать этот вектор числом, причем так, чтобы существовала однозначная и примитивно-рекурсивная расшифровка этого кода (т. е. извлечение нужного разряда вектора). В качестве такого кода можно предложить число

$$\varphi(x_1, \dots, x_n, y) = 2^{f_1(x_1, \dots, x_n, y)} \cdot 3^{f_2(x_1, \dots, x_n, y)} \cdot \dots \cdot p_{k-1}^{f_k(x_1, \dots, x_n, y)},$$

где p_i – i -е простое число (2 считается 0-м простым числом). Продемонстрируем эту идею на примере оператора R_{12} .

Пусть $f_1(x, y), f_2(x, y)$ заданы совместной рекурсией:

$$f_1(x, 0) = g_1(x);$$

$$f_2(x, 0) = g_2(x);$$

$$f_1(x, y+1) = h_1(x, y, f_1(x, y), f_2(x, y));$$

$$f_2(x, y+1) = h_2(x, y, f_1(x, y), f_2(x, y)).$$

Определим кодирующую функцию F :

$$F(x, 0) = 2^{g_1(x)} \cdot 3^{g_2(x)};$$

$$F(x, y+1) = 2^{h_1(x, y, f_1(x, y), f_2(x, y))} \cdot 3^{h_2(x, y, f_1(x, y), f_2(x, y))}.$$

Тогда $f_1(x, y) = \mu_{z \leq F(x, y)}(\neg Pd.2^{z+1}(F(x, y)))$ т. е. равна показателю при 2 в разложении $F(x, y)$ на простые множители; $f_2(x, y) = \mu_{z \leq F(x, y)}(Pd.3^{z+1}(F(x, y)))$, т.е. равна показателю при 3 в разложении $F(x, y)$ на простые множители. (Определение предиката Pd см. в примере _).

Следует иметь в виду, что такая кодировка вовсе не предлагается в качестве метода рекурсивного вычисления функций-векторов. Наоборот, предлагается прямой и более простой метод совместной рекурсии, а кодировка является лишь доказательством того, что этот метод относится к числу примитивно-рекурсивных средств вычисления.

Можно предложить полезную схемную интерпретацию примитивной рекурсии вообще и совместной рекурсии в частности. На рис. 1, и изображена схема, состоящая из устройства, вычисляющего за один такт функцию h от двух переменных, и элемента задержки на один такт; по каналам схемы могут передаваться натуральные числа. Время t будем считать дискретным: $t=0, 1, 2, \dots$. Схема имеет один вход x и выход f . Однако выход f зависит не только от значения x , но и от момента t , в который он рассматривается, т. е. представляет собой некоторую функцию $f(x, t)$. Рассмотрим эту зависимость подробнее. Будем считать значение x на все время рассмотрения, начиная с $t=0$, произвольным, но фиксированным. В начальный момент $t=0$ значение второго входа h является константой c , зависящей от начального состояния схемы: $f(x, 0) = h(x, c) = g(x)$. В момент $t=1$ $f(x, 1) = h(x, f(x, 0))$; в общем случае $f(x, t+1) = h(x, f(x, t))$. Таким образом, схема рис. 1,а реализует примитивную рекурсию по переменной t , т.е. по времени. Нетрудно убедиться, например, что если h выполняет умножение, а $c=1$, то $f(x, t) = x^{t+1}$. При аналогичной интерпретации (с начальными константами c_1, c_2 ;) схема на рис.1 реализует схему совместной рекурсии, причем рекурсия также ведется по времени, общему для устройств h_1 и h_2 . Доказательство того, что совместная рекурсия – это ПР-оператор, в терминах таких схем означает, что перекрестные обратные связи на рис.1 можно заменить простыми контурами

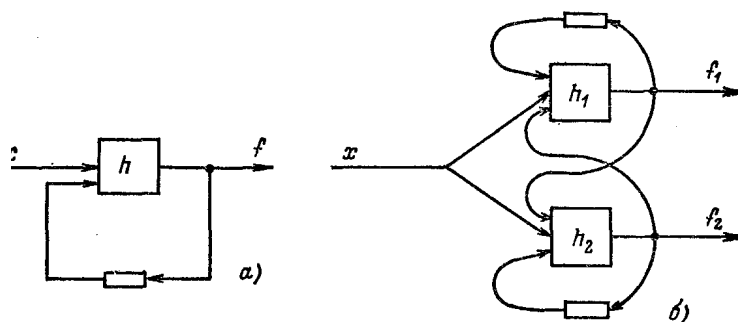


Рис. 1

обратной связи на рис. 1,б можно заменить простыми контурами обратной связи типа приведенных на рис. 1, а. Структура схемы при этом станет проще, однако понадобятся дополнительные вычислительные устройства, формирователь числового кода вектора (f_1, f_2), передаваемого по одному каналу, и декодирующее устройство с двумя выходами f_1 и f_2 .

Описанная схемная интерпретация примитивно-рекурсивных функций проходит при одном предположении, что за один такт любой канал схемы способен передать, а вычислительное устройство — воспринять и переработать любое, сколь угодно большое натуральное число. Такое предположение физически нереализуемо, поэтому теория таких схем не будет представлять самостоятельного интереса. Если же наложить ограничения на возможности каналов и элементов схем, то это приведет к теории конечных автоматов и схем из автоматов.

Подведем некоторые итоги.

Из простейших функций — константы 0, функции $x+1$ и функций тождества I_m^n — с помощью операторов суперпозиции и примитивной рекурсии было получено огромное разнообразие функций, включающих основные функции арифметики, алгебры и анализа (с поправкой на целочисленность). Тем самым выяснено, что эти функции имеют примитивно-рекурсивное описание, которое однозначно определяет процедуру их вычисления; следовательно, их естественно отнести к классу вычислимых функций.

Сделаем два замечания.

Во-первых, все примитивно-рекурсивные функции всюду определены. Это следует из того, что простейшие функции всюду определены, а операторы S_m^n и R_n это свойство сохраняют.

Во-вторых, строго говоря, мы имеем дело не с функциями, а с их примитивно-рекурсивными описаниями. Различие здесь имеет тот же смысл, что и отмечавшееся неоднократно в гл. 3 различие между функциями и их представлениями в виде формул. Примитивно-рекурсивные описания также можно разбить на классы эквивалентности, отнеся в один класс все описания, задающие одну и ту же функцию.

Однако задача распознавания эквивалентности примитивно-рекурсивных описаний, алгоритмически неразрешима.

Тезис Черча. Связь рекурсивных функций с машиной Тьюринга

Определение 1 Функция называется *частично-рекурсивной*, если она может быть построена из простейших функций 0 , $x+1$, I_m^n с помощью конечного числа применений оператора суперпозиции S_m^n , примитивной рекурсии R_n и μ -оператора.

Тезис Черча является аналогом тезиса Тьюринга для рекурсивных функций: ***всякая функция, вычисляемая некоторым алгоритмом, частично-рекурсивна.***

Утверждение: функция вычислима машиной Тьюринга, тогда и только тогда, когда она частично-рекурсивна.

Это утверждение об эквивалентности двух алгоритмических моделей является (в отличие от тезисов) вполне точным математическим утверждением и должно быть доказано.

Разобьем утверждение на две теоремы.

Теорема 1.(достаточности) *Всякая частично-рекурсивная функция вычислима на машине Тьюринга.*

Теорема 2.(необходимости) *Всякая функция вычисляемая на машине Тьюринга, частично-рекурсивна.*

Лекция 18

Трудоёмкость вычислений

В классической теории алгоритмов задача считается разрешимой, если существует решающий ее алгоритм. Однако для реализации некоторых алгоритмов при любых разумных с точки зрения физики предположениях о скорости выполнения элементарных шагов может потребоваться больше времени, чем по современным воззрениям существует вселенная. Поэтому возникает потребность конкретизировать понятие разрешимости и придать ему оценочный, количественный характер, введя такие характеристики алгоритмов, которые позволяли бы судить о возможности и целесообразности их практического применения.

Среди различных характеристик алгоритмов наиболее важными с прикладной точки зрения являются: время и память, расходуемые при вычислении.

Физическое время вычисления алгоритма характеризуется произведением τt , где t – число действий вычисления, τ – среднее физическое время реализации одного шага. Число шагов t определяется

описанием алгоритма в данной алгоритмической модели, это – величина математическая, не зависящая от особенностей физической реализации модели. Напротив, τ зависит от реализации и определяется скоростью обработки сигналов в элементах, записи и считывания информации и т.д. Поэтому число действий t считают математическим временем вычисления алгоритма, определяющим физическое время вычисления с точностью до константы τ , зависящей от реализации.

Память как количественная характеристика алгоритма определяется количеством s единиц памяти (ячеек ленты машины Тьюринга или машинных слов в современных ЭВМ). Ясно, что эта величина по порядку не может превосходить числа шагов вычисления: $s \leq \mu t$, где μ – максимальное число единиц памяти, используемых в данной машине на одном шаге. Напротив, t может существенно превосходить s . С этой точки зрения время более тонко определяет сложность алгоритма, чем память; и это – одна из причин, по которой исследованию временных характеристик алгоритма уделяется большее внимание.

Трудоёмкость алгоритма – это число элементарных действий, выполненных при его вычислении.

Полной характеристикой конкретного варианта задачи является его формальное описание. Характеристикой сложности описания можно считать его объем, который называют *размерностью задачи* (например, для изоморфизма графов размерностью задачи можно считать число символов в матрицах смежности графов); тогда исследование трудоёмкости алгоритма рассматривается как исследование зависимости трудоёмкости вычисления от размерности задачи, решаемой алгоритмом.

И в математике, и на практике в конечном счете нас интересуют не алгоритмы сами по себе, а задачи, которые они решают. Одна и та же задача может решаться различными алгоритмами и на разных машинах. Если машина M зафиксирована, то трудоёмкостью данной задачи относительно машины M называется минимальная из трудоёмкостей алгоритмов, решающих задачу на машине M . Задача, трудоёмкость которых была бы определена точно, довольно мало; хорошим результатом считается определение ее по порядку, т. е. с точностью до множителя, ограниченного некоторой константой. Чаще удается оценить ее сверху или снизу. Оценку сверху получают, указав конкретный алгоритм решения задачи: по определению, трудоёмкость задачи не превосходит трудоёмкости любого из решающих ее алгоритмов. Оценки трудоёмкости снизу – гораздо более трудное дело; их получают обычно из некоторых общих соображений (например, мощностных или информационных).

Можно ли говорить об инвариантности теории трудоёмкости вычислений? Иначе говоря, возможны ли утверждения о трудоёмкости вычислений, сохраняющиеся при переходе к любой алгоритмической модели? Что касается прямых количественных оценок, то инвариантами не являются не только константы, но и степени. Например, доказано, что

трудоемкость распознавания симметричности слова длины n относительно его середины на машине Тьюринга не меньше, чем cn^2 , тогда как для любой ЭВМ, имеющей доступ к памяти по адресу, допускающей операции над адресами, легко написать программу, решающую эту задачу с линейной трудоемкостью.

Таким образом, скорости вычислений на разных моделях различны. Однако строить теорию трудоемкости вычислений, привязываясь к некоторым конкретным моделям, неудобно ни для теории, ни для практики. Для теории – потому, что такая привязка не дает достаточно объективных характеристик трудоемкости задачи, т.е. не позволяет отделить влияние особенностей выбранной модели от специфики самой задачи; для практики — потому, что разнообразие реальных машин растет, и нужны общие понятия и методы оценки трудоемкости решения задач, которые сохраняют свою силу при любых изменениях в мире компьютеров. Поэтому инвариантная теория трудоемкости нужна, и вопрос не в том, возможна ли она, а в том, как ее построить (т.е. какие инварианты найти). Для того чтобы обсуждать этот вопрос, прежде всего следует посмотреть, как меняется трудоемкость при переходе от одной машины к другой. Это рассмотрение мы начнем с некоторого краткого обзора парка абстрактных машин, о которых будет идти речь.

До сих пор в явном виде была описана только одна абстрактная машина – машина Тьюринга. Можно неявно использовать машину другого типа, гораздо более близкую к современным ЭВМ, в которой возможен доступ к памяти по адресам. Такая машина, называемая *машиной с произвольным доступом к памяти*, может на следующем шаге переходить к любой ячейке с указанным адресом (команды условного и безусловного переходов) и реализовать команды-операторы вида $b: =f(a_1, a_2, \dots, a_p)$ (выполнить операцию f над содержимым ячеек a_1, a_2, \dots, a_p и результат положить в ячейку b). Возможны различные варианты моделей машины с произвольным доступом к памяти, в более сложных вариантах допускаются операции над адресами. Здесь мы не будем рассматривать все эти варианты, ограничившись фиксацией лишь одной простой модели – машины элементарных логических операций, или кратко L-машины. Относительно других моделей абстрактных машин ограничимся констатацией их основных свойств, которых будет достаточно при последующих рассмотрениях. Будем считать, что каждая машина имеет конечное число устройств (головок, устройств управления головками, процессоров-устройств, выполняющих элементарные операции, и т.д.), каждое устройство и каждая ячейка памяти могут находиться в одном из конечного числа возможных состояний (состояние ячейки памяти – это записанный в ней символ), и выполнение любого элементарного действия (шага) зависит от информации из конечного числа ячеек памяти, ограниченного некоторой константой μ . Будем говорить, что все ячейки читаются на данном шаге. Полное состояние машины, т. е. набор

состояний устройств, состояний ячеек памяти и указание ячеек, читаемых в настоящий момент, называется, *конфигурацией машины*.

Еще одно вступительное замечание. Алгоритм, осуществляемый машиной, может быть реализован двояким образом: он может быть «встроен» в управляющее устройство или записан в памяти машины. В первом случае машина является специализированной и может выполнять только данный алгоритм; чтобы изменить алгоритм, надо поменять управляющее устройство. Таковы машины Тьюринга в примерах. Во втором случае запись алгоритма в памяти называется программой, а сама машина - *программируемой*; алгоритм, встроенный в управляющее устройство, решает задачу исполнения программ, записанных в памяти машины. Такова универсальная машина Тьюринга и все реальные универсальные ЭВМ. В обоих случаях начальная конфигурация машины - состояния всей памяти и всех устройств - полностью определяет процесс вычисления.

Машина элементарных логических операций (L-машина) - это машина с произвольным доступом к памяти, имеющая следующую систему команд:

$x := 0;$
 $x := 1;$
 $x := y;$
 $x := \neg y;$
 $x := y \& z;$
 $x := y \vee z;$
“конец”;

где x, y, z — адреса ячеек памяти; $:=$ - знак присвоения.

Программа элементарных логических операций (L-программа) с однократной записью - это программа, состоящая из указанных команд, в которой запись в каждую ячейку памяти производится не более одного раза (читать ячейку можно неоднократно).

Можно было бы дополнить систему команд L-машины командами ввода информации в ячейки памяти для данных, ввода программ и вывода результатов решения задач, однако будем считать, что эти элементы памяти доступны для ввода информации и обозрения извне (трудоемкость этих действий следует считать пропорциональной количеству элементов памяти, куда надо ввести или откуда надо вывести информацию). Пусть программа и данные (значения двоичных переменных) находятся в разных секциях памяти, и каждая секция имеет свою адресацию. Тогда можно так перекодировать программу с однократной записью, что номера команд программы и адреса ячеек памяти, в которых эти команды производят запись станут одинаковыми, и будут иметь вид $i:i := U_i$ ($i=1, 2, \dots, t-1$), где выражения U_i имеют вид либо 0, либо 1, либо j , либо $\neg j$, либо $j \& k$, либо $j \vee k$ (в конце программы стоит команда “конец”).

Интерпретация машинных действий

Интерпретация элементарных действий

Теоремы о трудоемкости

Теорема 1. Пусть Манина M_2 полиномиальным образом со степенью C интерпретируема на машине M_1 , и задача p решается на машине M_2 из частичной конфигурации $C_{M_2}^n$ не более чем за t шагов, после чего машина останавливается. Тогда задача p может быть решена на машине M_1 не более чем за Vt^C шагов, где V – положительная константа, т.е. ее сложность относительно машины M_1 не больше Vt^C .

Теорема 2. Если трудоемкость задачи p относительно машины Тьюринга T не больше t , то ее можно решить при помощи не более чем $Vt^2 + Ct = O(t^2)$ элементарных логических операций.

Полиномиальная интерпретируемость машин друг на друга. Аналогичным образом доказывается, что машина с произвольным доступом к памяти и «полным» набором команд – конечным набором операторов вычисления функций от конечного же количества двоичных переменных, условными и безусловными переходами и индексированными командами – полиномиальным образом интерпретируема на машине элементарных логических операций. Однако в этом случае степень больше трех (сложнее всего интерпретация индексированных команд, особенно адресов, которые могут появиться в результате индексации). Обратная полиномиальная интерпретируемость машины элементарных логических операций L на машине с произвольным доступом к памяти тривиальна, так как в системе команд последней машины есть все команды первой.

Так как машина с произвольным доступом к памяти полиномиальным образом интерпретируема на машине элементарных логических операций L , для доказательства ее полиномиальной интерпретируемости на некоторой машине Тьюринга PU достаточно доказать полиномиальную интерпретируемость машины L на машине PU . Кроме того, этим будет доказана полиномиальная интерпретируемость произвольных машин Тьюринга T на машине PU .

Теорема 3. L -машина полиномиальным образом интерпретируется на машине Тьюринга. Для доказательства нужна определенная техника программирования машин Тьюринга, и мы его опускаем.

**7. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ
ПРАКТИЧЕСКИХ РАБОТ
План практических работ**

| № | Содержание практического занятия | Часы |
|-------|---|------|
| 1 | Ориентированные, неориентированные графы, различные виды графов. Локальные характеристики графов. | 4ч. |
| 2 | Части графов. Плоские графы. Реализация графов. Изоморфизм графов. | 4ч. |
| 3 | Представление графов матричное и списочное. | 2ч. |
| 4 | Логика высказываний. Логическое следование, принцип дедукции. | 2ч. |
| 5 | Исчисление высказываний. Аксиомы и правила вывода. | 2ч. |
| 6 | Основные метатеоремы исчисления высказываний. Теорема дедукции. | 4ч. |
| 7 | Исчисление предикатов и теории первого порядка. Аксиомы и правила вывода. Правила переименования свободных и связанных переменных. | 4ч. |
| 8 | Префиксная нормальная форма в исчислении предикатов. | 2ч. |
| 9 | Примеры машин Тьюринга. Описание машин T_+ , $T_{\text{коп}}$, T_{2x} , T_{++} . | 4ч. |
| 10 | Композиция машин Тьюринга. Пример машин T_{++} , T_x . Проблема остановки. | 4ч. |
| 11 | Рекурсивные функции. Примитивно-рекурсивные функции (сложение, умножение, возведение в степень, урезанное вычитание, \min , \max). | 2ч. |
| 12 | Примитивно-рекурсивные функции | 2ч. |
| Итого | | 36 |

Практическое занятие №1

Тема: Ориентированные, неориентированные графы, различные виды графов.

Для практического занятия использовать материалы лекций 1 и 2.

Контрольные вопросы и задания

1. Понятие ориентированного и неориентированного графов. В чем заключается их различие?
2. Какие графы симметричные. Примеры.
3. Полные и пустые графы. Примеры.
4. Двудольные графы. Признаки двудольного графа.
5. Деревья. Признаки графа-дерева.

Практическое занятие №2

Тема: Ориентированные, неориентированные графы. Локальные характеристики графов.

Для практического занятия использовать материалы лекций 3.

Контрольные вопросы и задания

1. Понятие ориентированного и неориентированного графов. В чем заключается их различие?
2. Определить локальные характеристики для различных графов.
3. Проверить условие теоремы Эйлера о рукопожатиях.

Практическое занятие №3

Тема: Части графов. Плоские графы. Реализация графов. Изоморфизм графов.

Для практического занятия использовать материалы лекций 3.

Контрольные вопросы и задания

1. Перечислить части графа.
2. Понятие плоского графа.
3. Теорема о реализации графа в \mathbf{R}^3 .
4. Понятие изоморфизма графов.
5. Определить изоморфизм для различных графов.

Практическое занятие №4

Тема: Представление графов матричное и списочное.

Для практического занятия использовать материалы лекций 4.

Контрольные вопросы и задания

1. Для графов использовать различные описания.
2. Какие графы удобнее описывать матрицами, а какие списками?

Практическое занятие №5

Тема: Логика высказываний. Логическое следование, принцип дедукции. Метод резолюций.

Для практического занятия использовать материалы лекций 5 и 6.

Контрольные вопросы и задания

1. Высказывание и умозаключение в логике высказываний.
2. Логическое следование. Схемы логических умозаключений.

3. Принцип дедукции.
4. Метод резолюций.

Практическое занятие №6

Тема: Исчисление высказываний. Аксиомы и правила вывода.

Для практического занятия использовать материалы лекции 7 и 8. Из логики высказываний использовать понятия: тождественно-истинные формулы, логически верные умозаключения логики высказываний, логические схемы рассуждений, правило утверждения. Из построения исчисления высказываний использовать понятия: вывод и выводимость в исчислении высказываний, доказательство в теории Т.

Контрольные вопросы и задания

1. Как построено исчисление высказываний (основные принципы)?
2. Какие формулы являются аксиомами исчисления высказываний?
3. Какие правила вывода используются в исчислении высказываний?
4. Выводимость в исчислении высказываний, показать на примерах.
5. Выучить систему аксиом П.

Практическое занятие №7

Тема: Основные метатеоремы исчисления высказываний. Теорема дедукции.

Для практического занятия использовать материалы лекции 8, 9. Из исчисления высказываний использовать понятия: метатеорема исчисления, теорема дедукции, применение теоремы дедукции на примерах.

Контрольные вопросы и задания

1. Теорема дедукции.
2. Применение теоремы дедукции. Примеры
3. Теорема о выводимости в исчислении высказываний.
4. Применение теоремы о выводимости. Проверка формул исчисления высказываний на выводимость.

Практическое занятие №8

Тема: Исчисление предикатов и теории первого порядка. Аксиомы и правила вывода. Правила переименования свободных и связанных переменных.

Для практического занятия использовать материалы лекций 9 и 10. Из исчисления предикатов использовать понятия: теория первого порядка,

принципы построения исчисления предикатов, на примере правил переименования показать выводимость в исчислении предикатов.

Контрольные вопросы и задания

1. Как построено исчисление предикатов (основные принципы)?
2. Аксиомами исчисления предикатов?
3. Какие правила вывода используются в исчислении предикатов?
4. Выводимость в исчислении предикатов, показать на примерах.
5. Выучить систему аксиом исчисления предикатов.

Практическое занятие №9

Тема: Доказательства в исчислении предикатов. Префиксная нормальная форма.

Для практического занятия использовать материалы лекции 11, 12. Из исчисления предикатов использовать понятия: доказательства в исчислении предикатов, понятие префиксной нормальной формы, этапы приведения формулы исчисления предикатов к префиксной нормальной форме, эквивалентности, используемые в исчислении предикатов.

Контрольные вопросы и задания

- 1) Метод доказательства от противного. Показать на примере предикатной формулы.
- 2) Понятие префиксной нормальной формы.
- 3) Процедура приведения к ПНФ формулы исчисления предикатов.
- 4) Эквивалентности, используемые в исчислении предикатов.
- 5) Заданные формулы привести к ПНФ.
- 6) Доказательство в исчислении предикатов.

Практическое занятие №10

Тема: Примеры машин Тьюринга. Описание машин T_+ , $T_{\text{коп}}$, T_{2x} , T_{++} .

Для практического занятия использовать материалы лекций 13 и 14. Использовать понятия: машина Тьюринга, конфигурация машины Тьюринга, правильное вычисление функции на машине Тьюринга, протокол машины Тьюринга.

Контрольные вопросы и задания

1. Для машины Тьюринга T_+ использовать разные представления.
2. Для заданных машин T_+ , $T_{\text{коп}}$, T_{++} построить протоколы, описать работу машин.

Практическое занятие №11

Тема: Композиция машин Тьюринга. Примеры машин T_{2x} , T_x . Вычисление предикатов на примере предиката $P(x)$ – “ x четное”. Проблема остановки.

Для практического занятия использовать материалы лекций 15 и 16. Использовать понятия: композиция машин Тьюринга, построение композиции машин Тьюринга, вычисление предикатов на машине Тьюринга, вычисление предикатов с восстановлением и без восстановления, тезис Тьюринга, проблема остановки, тезис Черча для машин Тьюринга.

Контрольные вопросы и задания

1. Ввести понятие композиции машин Тьюринга.
2. Для машины $T_+(T_{\text{коп}})$ построить протокол, определить какую функцию вычисляет эта машина.
3. Пример машины Тьюринга, вычисляющей предикат, построение протокола этой машины.
4. В чем заключается проблема остановки?
5. Тезис Тьюринга и Черча.

Практическое занятие №12

Тема: Рекурсивные функции. Примитивно-рекурсивные функции (сложение, умножение, возведение в степень, урезанное вычитание, \min , \max).

Для практического занятия использовать материалы лекции 17. Использовать понятия: оператора суперпозиции, примитивно-рекурсивной функции, схема рекурсии.

Контрольные вопросы и задания

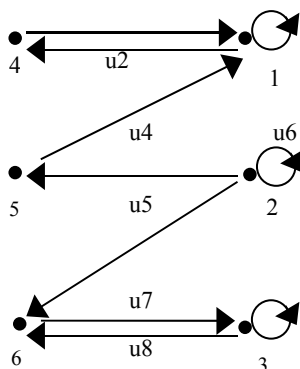
1. Понятие примитивно-рекурсивной функции.
2. Схема рекурсии, привести примеры.
3. Тестирование примитивно-рекурсивных функций.

8. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КОНТРОЛЬНЫХ РАБОТ

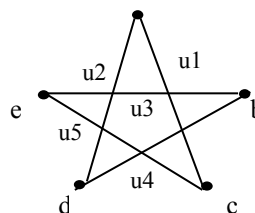
Контрольная работа №1

Вариант 1

G1:



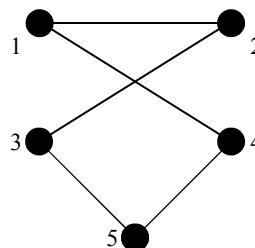
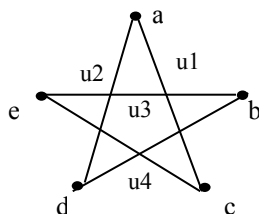
G2:



Задание 1. Для графов G1 и G2 построить матрицы смежности, инцидентности и достижимости.

Задание 2. Для графов G1 и G2 определить локальные характеристики.

Задание 3. Для графов определить изоморфизм.



Контрольная работа №2

Вариант 1

1. Записать таблицу истинности для функции: $(x \oplus y) \vee \neg z$
2. Формулу $x \rightarrow \neg y$ записать в базисе $\{\downarrow\}$.

Вариант 2

1. Записать таблицу истинности для функции: $x \rightarrow (\neg y \oplus z)$

2. Формулу $\neg x \rightarrow y$ записать в базисе $\{\downarrow\}$.

Вариант 3

1. Записать таблицу истинности для функции: $\neg x \oplus (\neg y \downarrow z)$
2. Формулу $\neg(x \rightarrow y)$ записать в базисе $\{\downarrow\}$.

Вариант 4

1. Записать таблицу истинности для функции: $\neg x \wedge (y \mid \neg z)$
2. Формулу $\neg y \oplus z$ записать в базисе $\{\downarrow\}$.

Вариант 5

1. Записать таблицу истинности для функции: $\neg x \wedge y \oplus \neg z$
2. Формулу $x \rightarrow y$ записать в базисе $\{\downarrow\}$.

Контрольная работа №3, №4

«Математическая логика»

Вариант1

I. “Всякая тождественно-истинная формула является теоремой исчисления высказываний.” Указать варианты, содержащие теоремы:

- а) $\Gamma, U \vdash B$, то $\Gamma \vdash U \rightarrow B$;
- б) $\vdash U \rightarrow B$;
- в) $\vdash A \rightarrow (A \rightarrow A)$;
- г) $\vdash (U \rightarrow B) \rightarrow B$.

II. Какие формулы являются аксиомами исчисления предикатов?

- а) $\forall x F(x) \rightarrow F(y)$;
- б) $\exists y F(y) \rightarrow \forall x F(x)$;
- в) нет таких формул.

III. $\forall x((P(x) \rightarrow Q(x)) \& (Q(x) \rightarrow P(x)))$

- а) тождественно-истинная формула
- б) тождественно-ложная формула
- с) формула принимает разные значения на наборах предикатов P, Q.

«Математическая логика»

Вариант2

I. Если \mathfrak{K} – выводимая формула, содержащая букву A ($\mathfrak{K}(A)$), то выводима формула $\mathfrak{K}(B)$, где B произвольная формула: $\frac{\mathfrak{K}(A)}{\mathfrak{K}(B)}$

- а) это правило заключения;
- б) это правило подстановки;
- в) это не есть правило.

II. Какие формулы являются аксиомами исчисления предикатов?

- а) $\forall xF(x) \rightarrow F(y)$;
- б) $F(y) \rightarrow \exists xF(x)$;
- в) нет таких формул.

III. $\forall x((P(x) \rightarrow Q(x)) \vee \neg(P(x) \rightarrow Q(x)))$

- а) тождественно-ложная формула
- б) тождественно-истинная формула
- в) формула принимает разные значения на наборах предикатов P, Q .

«Математическая логика»

Вариант3

I. “Всякая тождественно-истинная формула является теоремой исчисления высказываний.” Указать варианты, содержащие теоремы:

- а) $\Gamma, U \vdash B$, то $\vdash U \rightarrow B$;
- б) $\vdash \neg(A \& B) \sim \neg A \vee \neg B$;
- в) $\vdash A \rightarrow (A \rightarrow A)$;
- г) $\vdash (U \rightarrow B) \rightarrow B$.

II. Какие формулы являются аксиомами исчисления предикатов?

- а) $\forall xF(x) \rightarrow \exists yF(y)$;
- б) $\exists yF(y) \rightarrow \forall xF(x)$;
- в) $F(y) \rightarrow \exists xF(x)$.

III. Если $\Gamma, A \vdash B$ и $\Gamma, A \vdash \neg B$, то $\Gamma \vdash \neg A$.

- а) это теорема дедукции;
- б) это правило введения отрицания.

«Математическая логика»

Вариант4

I. Если \mathfrak{K} – выводимая формула, содержащая букву A ($\mathfrak{K}(A)$), то выводима формула $\mathfrak{K}(B)$, где B тождественно-истинная формула: $\frac{\mathfrak{K}(A)}{\mathfrak{K}(B)}$

$\mathfrak{K}(B)$

- а) это правило заключения;
- б) это правило подстановки;
- в) это не есть правило.

II. Являются ли соотношения эквивалентными?

- а) $\neg(\exists xF(x)) \sim \forall x \neg F(x)$;
- б) $\neg(\forall xF(x)) \sim \exists x \neg F(x)$.

III. $\forall x((P(x) \rightarrow Q(x)) \vee (Q(x) \rightarrow P(x)))$

- а) тождественно-истинная формула
- б) тождественно-ложная формула
- с) формула принимает разные значения на наборах предикатов P,Q.

«Математическая логика»
Вариант5

I. “Всякая тождественно-истинная формула является теоремой исчисления высказываний.” Указать варианты, содержащие теоремы:

- а) $\vdash (U \rightarrow V) \rightarrow V$;
- б) $\vdash (A \& B) \rightarrow V$;
- в) $\vdash \neg \neg A \rightarrow A$;
- г) $\Gamma, U \vdash V$, то $\Gamma \vdash U \rightarrow V$.

II. Являются ли соотношения эквивалентными?

- а) $\neg(\exists xF(x)) \sim \exists x \neg F(x)$;
- б) $\neg(\forall xF(x)) \sim \exists x \neg F(x)$.

III. Если $\Gamma, A \vdash B$, то $\Gamma \vdash A \rightarrow B$.

- а) это теорема дедукции;
- б) это правило введения отрицания.

«Математическая логика»
Вариант6

I. Если \mathfrak{K} и $\mathfrak{K} \rightarrow \mathfrak{N}$ – выводимые формулы, то \mathfrak{N} выводима: $\frac{\mathfrak{K}, \mathfrak{K} \rightarrow \mathfrak{N}}{\mathfrak{N}}$

- а) это правило заключения;
- б) это правило подстановки;
- с) это не есть правило.

II. Какие формулы являются аксиомами исчисления предикатов?

- а) $\forall xF(x) \rightarrow \exists yF(y)$;

- б) $\exists yF(y) \rightarrow \forall xF(x)$;
 в) нет таких формул.

III. $\forall x(\neg (P(x) \rightarrow Q(x)) \rightarrow P(x))$

- а) тождественно-ложная формула
 б) тождественно-истинная формула
 с) формула принимает разные значения на наборах предикатов P,Q.

«Теория алгоритмов»

Вариант1

I. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная таблицей переходов:

- а) суммирование двух чисел, представленных в унарной форме;
 б) копирование чисел, представленных в унарной форме;
 в) функцию $f(x)=2x$, для целого положительного x , представленного в унарной форме;
 г) предикат $P(x)=\{\text{“}x\text{-четное число”}\}$, без восстановления;
 д) нет такой функции.

| | q_1 | q_2 | q_3 | q_4 | q_5 |
|-----------|-----------|-----------------|-----------------|-----------------|-----------------|
| 1 | $1 q_3+1$ | $1 q_3+1$ | $1 q_2+1$ | λq_4-1 | λq_5-1 |
| λ | И q_z0 | λq_4-1 | λq_5-1 | И q_z0 | Л q_z0 |

II. Машина Т правильно вычисляет функцию f, если

- 1) для любых V и W, таких, что $f(V)=W$, $q_1V^* \Rightarrow_T q_zW^*$, где V^* и W^* - правильные записи V и W соответственно;
- 2) для любого V, такого, что f(V) не определена, машина Т, запущенная в стандартной начальной конфигурации q_1V^* , работает бесконечно;
- 3) пункт 1) и 2) вместе.

«Теория алгоритмов»

Вариант2

I. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная таблицей переходов:

- а) суммирование двух чисел, представленных в унарной форме;
 б) копирование чисел, представленных в унарной форме;
 в) функцию $f(x)=4x$, для целого положительного x , представленного в унарной форме;
 г) предикат $P(x)=\{\text{“}x\text{-нечетное число”}\}$, с восстановлением;
 д) нет такой функции.

| | q_1 | q_2 | q_3 |
|-----------|-----------------|-----------|-----------------|
| 1 | λq_2+1 | $1 q_2+1$ | $1 q_3-1$ |
| * | λq_z+1 | $1 q_3-1$ | |
| λ | | | λq_z+1 |

II. “Всякий алгоритм может быть реализован машиной Тьюринга”

- 1) для алгоритмичных конструктивных процедур невозможно строить реализующие их машины Тьюринга;
- 2) это есть тезис Тьюринга;

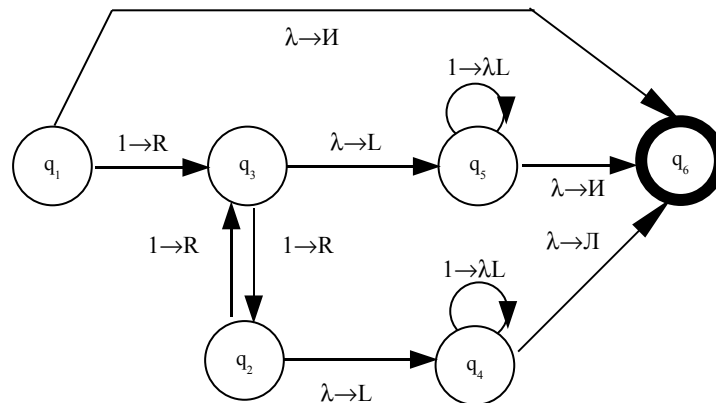
3) это есть проблема остановки.

«Теория алгоритмов»

Вариант 3

I. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная диаграммой переходов:

- а) суммирование двух чисел, представленных в унарной форме;
- б) копирование чисел, представленных в унарной форме;
- в) функцию $f(x)=2x$, для целого положительного x , представленного в унарной форме;
- г) предикат $P(x)=\{\text{“}x\text{-четное число”}\}$, без восстановления;
- д) нет такой функции.



II. Построение машины T_0 , такой, что для любой машины Тьюринга T и любых исходных данных α для машины T $T_0(\Sigma_T, \alpha) = И$, если $T(\alpha)$ останавливается, и $T_0(\Sigma_T, \alpha) = Л$, если $T(\alpha)$ не останавливается:

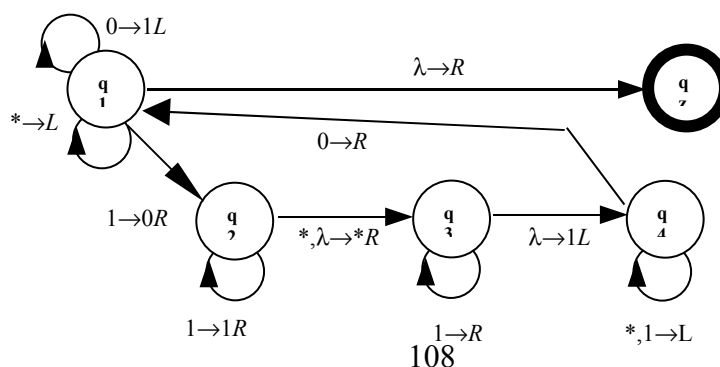
- 1) это есть тезис Тьюринга;
- 2) это есть алгоритмически неразрешимая проблема остановки;
- 3) для подобных алгоритмических процедур можно строить реализующие их машины Тьюринга.

«Теория алгоритмов»

Вариант 4

I. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная диаграммой переходов:

- а) умножение двух чисел, представленных в унарной форме;
- б) копирование чисел, представленных в унарной форме;
- в) функцию $f(x)=x$, для целого положительного x , представленного в унарной форме;
- г) предикат $P(x)=\{\text{“}x\text{-четное число”}\}$, без восстановления;
- д) нет такой функции.



II. Не существует машины Тьюринга Т, реализующей проблему остановки для произвольной машины Тьюринга:

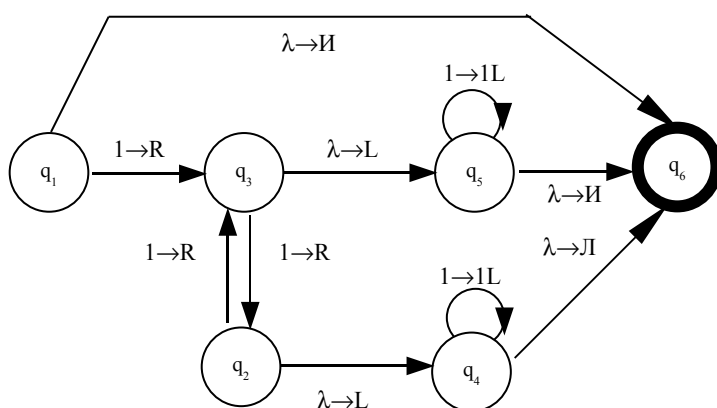
- 1) это есть тезис Тьюринга;
- 2) это есть теорема об алгоритмически неразрешимой проблеме остановки.

«Теория алгоритмов»

Вариант 5

I. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная диаграммой переходов:

- а) суммирование двух чисел, представленных в унарной форме;
- б) копирование чисел, представленных в унарной форме;
- в) предикат $P(x) = \{ \text{“} x \text{ - нечетное число”} \}$, с восстановлением;
- г) функцию $f(x) = 2x$, для целого положительного x , представленного в унарной форме;
- д) нет такой функции.



II. Машина Тьюринга с алфавитом $A_{исх} = \{И, Л\}$ и командами $q_1 И \rightarrow_T q_2 Л0$, $q_1 Л \rightarrow_T q_2 И0$ вычисляет:

- 1) отрицание логической переменной ($\neg P(\alpha)$);
- 2) значение логической переменной ($P(\alpha)$);
- 3) предикат $P(\alpha) = \{ \alpha \text{ - целое число} \}$.

«Теория алгоритмов»

Вариант 6

I. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная таблицей переходов:

- а) суммирование двух чисел, представленных в унарной форме;
- б) копирование чисел, представленных в унарной форме;
- в) функцию $f(x) = 2x$, для целого положительного x , представленного в унарной форме;
- г) предикат $P(x) = \{ \text{“} x \text{ - нечетное число”} \}$, с восстановлением;
- д) нет такой функции.

| | | | | | |
|--|-------|-------|-------|-------|-------|
| | q_1 | q_2 | q_3 | q_4 | q_5 |
|--|-------|-------|-------|-------|-------|

| | | | | | |
|-----------|-------------------|-----------------|-----------------|-------------------|-------------------|
| 1 | $1 q_3+1$ | $1 q_3+1$ | $1 q_2+1$ | λq_4-1 | λq_5-1 |
| λ | $\text{И } q_z 0$ | λq_4-1 | λq_5-1 | $\text{И } q_z 0$ | $\text{Л } q_z 0$ |

II. “Всякий алгоритм может быть реализован машиной Тьюринга”

- 1) для алгоритмичных конструктивных процедур невозможно строить реализующие их машины Тьюринга;
- 2) это есть проблема остановки;
- 3) неверны оба утверждения.

Контрольная работа №

«Математическая логика»

Вариант1

III. “Всякая тождественно-истинная формула является теоремой исчисления высказываний.” Указать варианты, содержащие теоремы:

- а) $\Gamma, U \vdash B$, то $\Gamma \vdash U \rightarrow B$;
- б) $\vdash U \rightarrow B$;
- в) $\vdash A \rightarrow (A \rightarrow A)$;
- г) $\vdash (U \rightarrow B) \rightarrow B$.

IV. Какие формулы являются аксиомами исчисления предикатов?

- а) $\forall x F(x) \rightarrow F(y)$;
- б) $\exists y F(y) \rightarrow \forall x F(x)$;
- в) нет таких формул.

III. $\forall x((P(x) \rightarrow Q(x)) \& (Q(x) \rightarrow P(x)))$

- а) тождественно-истинная формула
- б) тождественно-ложная формула
- с) формула принимает разные значения на наборах предикатов P,Q.

«Математическая логика»

Вариант2

III. Если \mathfrak{X} – выводимая формула, содержащая букву A ($\mathfrak{X}(A)$), то выводима формула $\mathfrak{X}(B)$, где B произвольная формула: $\frac{\mathfrak{X}(A)}{\mathfrak{X}(B)}$

- а) это правило заключения;
- б) это правило подстановки;
- в) это не есть правило.

IV. Какие формулы являются аксиомами исчисления предикатов?

- а) $\forall x F(x) \rightarrow F(y)$;

- б) $F(y) \rightarrow \exists xF(x)$;
 в) нет таких формул.

III. $\forall x((P(x) \rightarrow Q(x)) \vee \neg (P(x) \rightarrow Q(x)))$

- а) тождественно-ложная формула
 б) тождественно-истинная формула
 с) формула принимает разные значения на наборах предикатов P,Q.

«Математическая логика»
 Вариант3

IV. “Всякая тождественно-истинная формула является теоремой исчисления высказываний.” Указать варианты, содержащие теоремы:

- а) $\Gamma, U \vdash B$, то $\vdash U \rightarrow B$;
 б) $\vdash \neg(A \& B) \sim \neg A \vee \neg B$;
 в) $\vdash A \rightarrow (A \rightarrow A)$;
 г) $\vdash (U \rightarrow V) \rightarrow V$.

V. Какие формулы являются аксиомами исчисления предикатов?

- а) $\forall xF(x) \rightarrow \exists yF(y)$;
 б) $\exists yF(y) \rightarrow \forall xF(x)$;
 в) $F(y) \rightarrow \exists xF(x)$.

VI. Если $\Gamma, A \vdash B$ и $\Gamma, A \vdash \neg B$, то $\Gamma \vdash \neg A$.

- а) это теорема дедукции;
 б) это правило введения отрицания.

«Математическая логика»
 Вариант4

III. Если \mathfrak{R} – выводимая формула, содержащая букву A ($\mathfrak{R}(A)$), то выводима формула $\mathfrak{R}(B)$, где B тождественно-истинная формула: \mathfrak{R}
(A)

$\mathfrak{R}(B)$

- а) это правило заключения;
 б) это правило подстановки;
 в) это не есть правило.

IV. Являются ли соотношения эквивалентными?

- а) $\neg(\exists xF(x)) \sim \forall x \neg F(x)$;
 б) $\neg(\forall xF(x)) \sim \exists x \neg F(x)$.

III. $\forall x((P(x) \rightarrow Q(x)) \vee (Q(x) \rightarrow P(x)))$

- а) тождественно-истинная формула
- б) тождественно-ложная формула
- с) формула принимает разные значения на наборах предикатов P,Q.

«Математическая логика»
Вариант5

IV. “Всякая тождественно-истинная формула является теоремой исчисления высказываний.” Указать варианты, содержащие теоремы:

- а) $\vdash (U \rightarrow V) \rightarrow V$;
- б) $\vdash (A \& B) \rightarrow B$;
- в) $\vdash \neg \neg A \rightarrow A$;
- г) $\Gamma, U \vdash V$, то $\Gamma \vdash U \rightarrow V$.

V. Являются ли соотношения эквивалентными?

- а) $\neg(\exists x F(x)) \sim \exists x \neg F(x)$;
- б) $\neg(\forall x F(x)) \sim \exists x \neg F(x)$.

VI. Если $\Gamma, A \vdash B$, то $\Gamma \vdash A \rightarrow B$.

- а) это теорема дедукции;
- б) это правило введения отрицания.

«Математическая логика»
Вариант6

III. Если \mathfrak{X} и $\mathfrak{X} \rightarrow \mathfrak{X}$ – выводимые формулы, то \mathfrak{X} выводима: $\frac{\mathfrak{X}, \mathfrak{X} \rightarrow \mathfrak{X}}{\mathfrak{X}}$

- а) это правило заключения;
- б) это правило подстановки;
- с) это не есть правило.

IV. Какие формулы являются аксиомами исчисления предикатов?

- а) $\forall x F(x) \rightarrow \exists y F(y)$;
- б) $\exists y F(y) \rightarrow \forall x F(x)$;
- в) нет таких формул.

III. $\forall x(\neg(P(x) \rightarrow Q(x)) \rightarrow P(x))$

- а) тождественно-ложная формула
- б) тождественно-истинная формула
- с) формула принимает разные значения на наборах предикатов P,Q.

«Теория алгоритмов»

Вариант1

I. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная таблицей переходов:

- а) суммирование двух чисел, представленных в унарной форме;
- б) копирование чисел, представленных в унарной форме;
- в) функцию $f(x)=2x$, для целого положительного x , представленного в унарной форме;
- г) предикат $P(x)=\{\text{“}x\text{-четное число”}\}$, без восстановления;
- д) нет такой функции.

| | q_1 | q_2 | q_3 | q_4 | q_5 |
|-----------|-------------------|-----------------|-----------------|-------------------|-------------------|
| 1 | $1 q_3+1$ | $1 q_3+1$ | $1 q_2+1$ | λq_4-1 | λq_5-1 |
| λ | $\text{И } q_z 0$ | λq_4-1 | λq_5-1 | $\text{И } q_z 0$ | $\text{Л } q_z 0$ |

II. Машина Т правильно вычисляет функцию f , если

- 3) для любых V и W , таких, что $f(V)=W$, $q_1 V^* \Rightarrow_T q_z W^*$, где V^* и W^* - правильные записи V и W соответственно;
- 4) для любого V , такого, что $f(V)$ не определена, машина Т, запущенная в стандартной начальной конфигурации $q_1 V^*$, работает бесконечно;
- 3) пункт 1) и 2) вместе.

«Теория алгоритмов»

Вариант2

I. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная таблицей переходов:

- а) суммирование двух чисел, представленных в унарной форме;
- б) копирование чисел, представленных в унарной форме;
- в) функцию $f(x)=4x$, для целого положительного x , представленного в унарной форме;
- г) предикат $P(x)=\{\text{“}x\text{-нечетное число”}\}$, с восстановлением;
- д) нет такой функции.

| | q_1 | q_2 | q_3 |
|-----------|-----------------|-----------|-----------------|
| 1 | λq_2+1 | $1 q_2+1$ | $1 q_3-1$ |
| * | λq_z+1 | $1 q_3-1$ | |
| λ | | | λq_z+1 |

II. “Всякий алгоритм может быть реализован машиной Тьюринга”

- 4) для алгоритмичных конструктивных процедур невозможно строить реализующие их машины Тьюринга;
- 5) это есть тезис Тьюринга;
- 6) это есть проблема остановки.

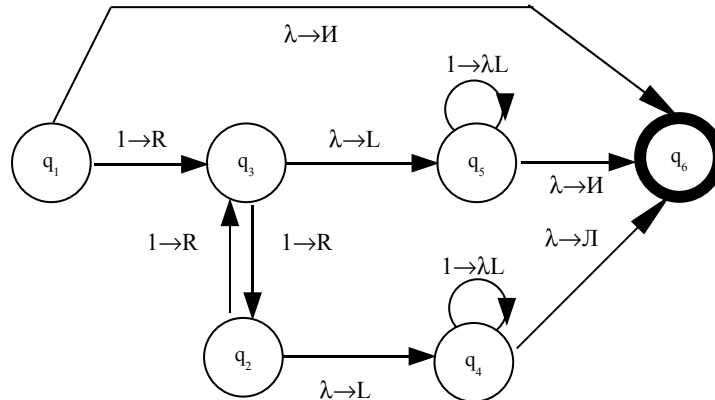
«Теория алгоритмов»

Вариант3

I. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная диаграммой переходов:

- а) суммирование двух чисел, представленных в унарной форме;
- б) копирование чисел, представленных в унарной форме;
- в) функцию $f(x)=2x$, для целого положительного x , представленного в унарной форме;

- г) предикат $P(x) = \{ \text{“}x\text{-четное число”} \}$, без восстановления;
- д) нет такой функции.



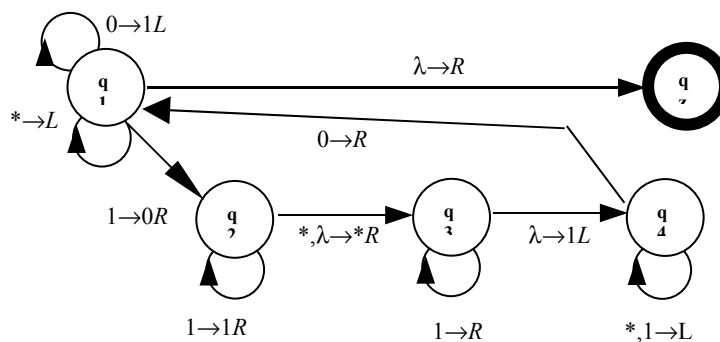
II. Построение машины T_0 , такой, что для любой машины Тьюринга T и любых исходных данных α для машины T $T_0(\Sigma_T, \alpha) = И$, если $T(\alpha)$ останавливается, и $T_0(\Sigma_T, \alpha) = Л$, если $T(\alpha)$ не останавливается:

- 1) это есть тезис Тьюринга;
- 4) это есть алгоритмически неразрешимая проблема остановки;
- 5) для подобных алгоритмичных процедур можно строить реализующие их машины Тьюринга.

«Теория алгоритмов» Вариант 4

I. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная диаграммой переходов:

- а) умножение двух чисел, представленных в унарной форме;
- б) копирование чисел, представленных в унарной форме;
- в) функцию $f(x) = x$, для целого положительного x , представленного в унарной форме;
- г) предикат $P(x) = \{ \text{“}x\text{-четное число”} \}$, без восстановления;
- д) нет такой функции.



II. Не существует машины Тьюринга T , реализующей проблему остановки для произвольной машины Тьюринга:

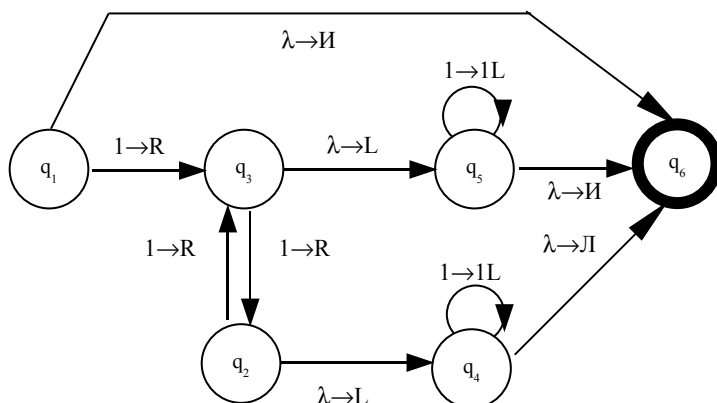
- 1) это есть тезис Тьюринга;
- 3) это есть теорема об алгоритмически неразрешимой проблеме остановки.

«Теория алгоритмов»

Вариант 5

I. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная диаграммой переходов:

- а) суммирование двух чисел, представленных в унарной форме;
- б) копирование чисел, представленных в унарной форме;
- в) предикат $P(x) = \{ \text{“} x \text{ - нечетное число”} \}$, с восстановлением;
- г) функцию $f(x) = 2x$, для целого положительного x , представленного в унарной форме;
- д) нет такой функции.



II. Машина Тьюринга с алфавитом $A_{исх} = \{И, Л\}$ и командами $q_1 И \rightarrow_T q_2 Л0$, $q_1 Л \rightarrow_T q_2 И0$ вычисляет:

- 1) отрицание логической переменной ($\neg P(\alpha)$);
- 2) значение логической переменной ($P(\alpha)$);
- 3) предикат $P(\alpha) = \{ \alpha \text{ - целое число} \}$.

«Теория алгоритмов»

Вариант 6

I. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная таблицей переходов:

- а) суммирование двух чисел, представленных в унарной форме;
- б) копирование чисел, представленных в унарной форме;
- в) функцию $f(x) = 2x$, для целого положительного x , представленного в унарной форме;
- г) предикат $P(x) = \{ \text{“} x \text{ - нечетное число”} \}$, с восстановлением;
- д) нет такой функции.

| | q_1 | q_2 | q_3 | q_4 | q_5 |
|-----------|-------------|-------------------|-------------------|-------------------|-------------------|
| 1 | $1 q_3 + 1$ | $1 q_3 + 1$ | $1 q_2 + 1$ | $\lambda q_4 - 1$ | $\lambda q_5 - 1$ |
| λ | $И q_2 0$ | $\lambda q_4 - 1$ | $\lambda q_5 - 1$ | $И q_2 0$ | $Л q_2 0$ |

II. “Всякий алгоритм может быть реализован машиной Тьюринга”

- 4) для алгоритмичных конструктивных процедур невозможно строить реализующие их машины Тьюринга;
- 5) это есть проблема останковки;
- 6) неверны оба утверждения.

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОРГАНИЗАЦИИ МЕЖСЕССИОННОГО КОНТРОЛЯ ЗНАНИЙ СТУДЕНТОВ

1. Межсессионная аттестация студентов проводится дважды в семестр на 7 и 13 неделях семестра.

2. Аттестационная оценка выставляется по результатам работы в семестре: выполнения домашних заданий, выполнения контрольных работ, сдачи промежуточного коллоквиума по 1-ой части дисциплины, успешного тестирования по 1-ой и 2-ой частям дисциплины, а также посещения практических занятий и посещений лекционных занятий.

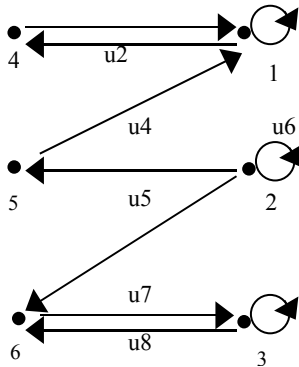
3. Организация аттестации студентов, проводится в соответствии с положением АмГУ о курсовых, экзаменах и зачетах.

10. ЗАДАНИЯ К РАСЧЕТНО-ГРАФИЧЕСКИМ РАБОТАМ

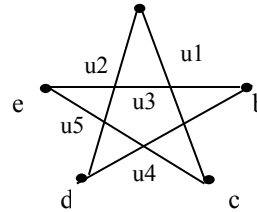
РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА №1

Вариант 1

G1:



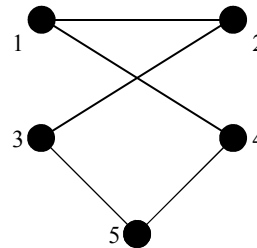
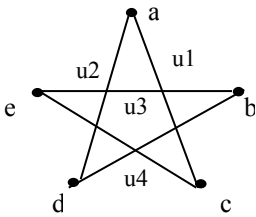
G2:



Задание 1. Для графов G1 и G2 построить матрицы смежности, инцидентности и достижимости.

Задание 2. Для графов G1 и G2 определить локальные характеристики.

Задание 3. Для графов определить изоморфизм.



Задание 4. Для графа Z построить дерево-остов.

Задание 5. Для дерева N записать код Прюфера.

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА №2

Вариант 1

АЛГЕБРА ЛОГИКИ

1. Построить таблицу истинности логической функции $f(x,y,z) = \neg(x \& (z \rightarrow y)) \vee ((y \rightarrow \neg x) \& z)$; для заданной функции f построить логическую схему.
2. Функцию $g(x,y) = x \oplus y$ представить в базисе $\{\&, \vee, \neg\}$.

ЛОГИКА ПРЕДИКАТОВ

3. Предикат $P(x,y)$ задан таблицей на предметной области $D = \{a,b,c\}$, провести квантификацию и определить логический смысл формул:
1) $\forall x \forall y P(x,y)$; 2) $\exists x P(x,a)$; 3) $\forall y P(b,y)$.

| x | y | P(x,y) |
|---|---|--------|
| a | a | 1 |
| a | b | 1 |
| a | c | 1 |
| b | a | 1 |
| b | b | 0 |
| b | c | 1 |
| c | a | 1 |
| c | b | 1 |
| c | c | 1 |

Математическая логика

4. “Всякая тождественно-истинная формула является теоремой исчисления высказываний.” Указать варианты, содержащие теоремы:
а) $\Gamma, U \vdash B$, то $\vdash U \rightarrow B$;
б) $\vdash \neg(A \& B) \sim \neg A \vee \neg B$;
в) $\vdash A \rightarrow (A \rightarrow A)$;
г) $\vdash (U \rightarrow B) \rightarrow B$.
5. Какие формулы являются аксиомами исчисления предикатов?
а) $\forall x F(x) \rightarrow \exists y F(y)$;
б) $\exists y F(y) \rightarrow \forall x F(x)$;
в) $F(y) \rightarrow \exists x F(x)$.

6. Если $\Gamma, A \vdash B$ и $\Gamma, A \vdash \neg B$, то $\Gamma \vdash \neg A$.

а) это теорема дедукции;

б) это правило введения отрицания.

Применение теоремы дедукции показать на примере.

Теория алгоритмов

7. Какую функцию натурального аргумента вычисляет машина Тьюринга, заданная диаграммой переходов:

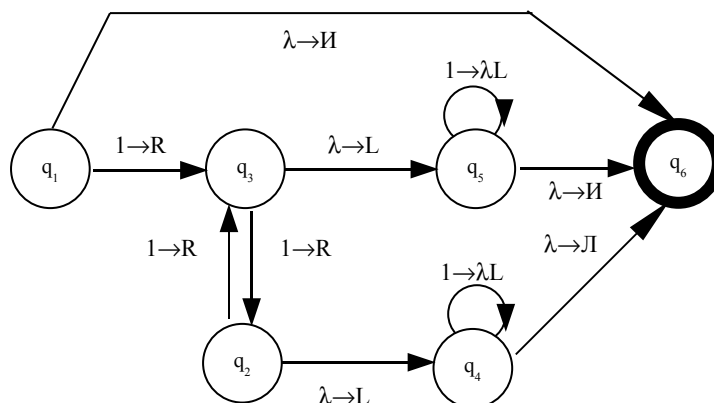
а) суммирование двух чисел, представленных в унарной форме;

б) копирование чисел, представленных в унарной форме;

в) функцию $f(x)=2x$, для целого положительного x , представленного в унарной форме;

г) предикат $P(x)=\{\text{“}x\text{-четное число”}\}$, без восстановления;

д) нет такой функции.



8. Построение машины T_0 , такой, что для любой машины Тьюринга T и любых исходных данных α для машины T $T_0(\Sigma_T, \alpha) = I$, если $T(\alpha)$ останавливается, и $T_0(\Sigma_T, \alpha) = L$, если $T(\alpha)$ не останавливается:

1) это есть тезис Тьюринга;

б) это есть алгоритмически неразрешимая проблема остановки;

7) для подобных алгоритмических процедур можно строить реализующие их машины Тьюринга.

11. КОМПЛЕКТЫ ЭКЗАМЕНАЦИОННЫХ БИЛЕТОВ

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 1

1. Понятие формальной системы. Принципы построения формальных теорий.
2. Понятие функции правильно вычислимой по Тьюрингу.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 2

1. Определение графа. Ориентированные и неориентированные графы.
2. Определение машины Тьюринга. Пример.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 3

1. Определение графа. Локальные характеристики графа.
2. Представления машин Тьюринга. Система команд, построение таблицы переходов, построение диаграммы переходов машин Тьюринга. Примеры.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 4

1. Принципы построения формальных теорий. Теоремы теории Т.
2. Машина Тьюринга вычисляющая сложение (T_+).
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 5

1. Исчисление высказываний. Принципы построения теории.
2. Машина Тьюринга вычисляющая копирование ($T_{\text{коп}}$).
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 6

1. Формулы. Аксиомы исчисления высказываний. Схемы формул исчисления высказываний.
2. Машина Тьюринга - композиция машин, вычисляющая функцию $f(x)=2x$ ($T_+(T_{\text{коп}})$).
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 7

1. Понятие графа. Теорема Эйлера о рукопожатиях.
2. Тезис Тьюринга. Проблема остановки.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 8

1. Геометрические графы. Плоские и неплоские графы. Теорема о правильной реализации.
2. Вычисление предикатов на машинах Тьюринга.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 9

1. Правила вывода в исчислении высказываний.
Теорема дедукции. Применение теоремы дедукции.
2. Машина Тьюринга - композиция машин, вычисляющая функцию $f(x)=2x (T_+(T_{\text{коп}}))$.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 10

1. Применение теоремы дедукции. Метод доказательства от противного.
2. Машина Тьюринга с правой полулентой. Пример машины T_{++} .
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 11

1. Матричное представление графов (ориентированных и неориентированных).
2. Машина Тьюринга вычисляющая копирование ($T_{\text{коп}}$).
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 12

1. Теорема о выводимости формулы в исчислении высказываний. Теоремы о тождественной истинности в исчислении высказываний. Примеры.
2. Машина Тьюринга вычисляющая сложение (T_+).
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 13

1. Исчисление предикатов и теории первого порядка. Принципы построения теории.
2. Понятие функции правильно вычислимой по Тьюрингу.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 14

1. Исчисление предикатов и теории первого порядка. Аксиомы исчисления предикатов.
2. Представления машин Тьюринга. Система команд, построение таблицы переходов, построение диаграммы переходов машин Тьюринга. Примеры.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 15

1. Исчисление предикатов и теории первого порядка. Правила вывода в исчислении предикатов.
2. Память машины Тьюринга, данные Машины Тьюринга, детерминированность машины Тьюринга.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 16

1. Выводимость и истинность в исчислении предикатов. Эквивалентности, выводимые в исчислении предикатов.
2. Конфигурация или полное состояние машины Тьюринга. Стандартная начальная конфигурация, Стандартная заключительная конфигурация.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 17

1. Геометрические графы. Изоморфизм графов.
2. Определение машины Тьюринга. Пример.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“___” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 18

1. Пример теории первого порядка исчисления с равенством.
2. Понятие алгоритма. Основные требования к алгоритмам.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“ ___ ” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 19

1. Применение теоремы дедукции. Метод доказательства от противного.
2. Вычисление предикатов на машинах Тьюринга. Пример.
- 3.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры
“ ___ ” _____ 2006г.
Заведующий кафедрой
Утверждаю: _____

Кафедра ИУС
Факультет МиИ
Курс II
Дисциплина
«Дискретная математика»

Экзаменационный билет № 20

1. Полные графы. Части графа.
2. Машина Тьюринга с правой полулентой. Пример машины T_{++} .
- 3.

12. ТЕСТОВЫЕ ЗАДАНИЯ

Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

«СОГЛАСОВАНО»

Заместитель председателя Совета УМО
к.т.н., доцент

_____ С.В. Коршунов

М.П.

«___» _____ 2007г.

«УТВЕРЖДАЮ»

Ректор

_____ А.Д. Плутенко

М.П.

«___» _____ 2007г.

ТЕСТЫ

По проведению самооценки уровня остаточных знаний студентов по дисциплине
«Дискретная математика»
для специальности
230102 «Автоматизированные системы обработки информации и управления»,

Зав.кафедрой ИУС, к.т.н., доцент

_____ Бушманов А.В.

Благовещенск
2006 г.

Федеральное агентство по образованию РФ
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО «АмГУ»)

Тестовые задания
по проверке остаточных знаний
по дисциплине «Дискретная математика»
(специальности 230201, 230102)
20 заданий
время тестирования 90 минут

Вариант 1

Выполнил: студент факультета математики и информатики

фамилия, имя и отчество

Курс _____ Группа № _____
Дата тестирования: _____

Результат _____ %

Оценка _____

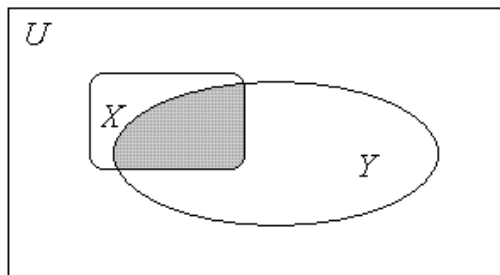
Вариант 1.

Теория множеств

Определить какие из перечисленных множеств пустые:

- a) $(A \cup B) \cap (\bar{A} \cup \bar{B})$;
- b) $(A \cap B) \setminus A$;
- c) $\bar{(X \cup Y)} \cup X \cup Y$;
- d) все множества непустые.

На диаграмме Эйлера-Венна изображено:



- a) $P_{X \cup Y} = \{x \mid x \in X \text{ или } x \in Y\}$;
- b) $P_{X \cap Y} = \{x \mid x \in X \text{ и } x \in Y\}$;
- c) $P_{X \setminus Y} = \{x \mid x \in X \text{ и } x \notin Y\}$;
- d) $P = \{x \mid x \in X\}$;
- e) $P_{\bar{X}} = \{x \mid x \in U \text{ и } x \notin X\}$.

Для отношения $R = \langle X \text{ перпендикуляр к } Y \rangle$ имеют место следующие свойства:

- a) рефлексивность, симметричность, транзитивность;
- b) антирефлексивность, несимметричность, транзитивность;
- c) рефлексивность, симметричность, нетранзитивность;
- d) верных свойств нет.

Указать какие отношения являются отношениями эквивалентности:

- a) $R1 \langle x - y > 0 \rangle$;
- b) $R2 \langle \text{быть подчиненным} \rangle$;
- c) $R3 \langle \text{равночисленность конечных множеств} \rangle$.

Верно ли утверждение: «Между множеством двоичных векторов разрядности n и булеаном конечного множества мощности n существует взаимно-однозначное соответствие».

- a) да;
- b) нет.

Какое число подмножеств содержит множество U , $|U|=n$:

- a) $n!$;
- b) 2^n ;
- c) n .

Какое тождество, из перечисленных является тождеством склеивания?

- a) $(X \cap Y) \cup Y = X \cup Y$;
- b) $\bar{(X \cup Y)} = \bar{X} \cap \bar{Y}$;
- c) $(X \cap \bar{Y}) \cup (X \cap Y) = X$;
- d) такого тождества нет.

Для бинарного отношения $R = \langle x \leq y \text{ и } x \text{ - четное, } y \text{ - нечетное} \rangle$, заданного на конечном множестве $V = \{1, 2, 3, 4\}$ построить матрицу бинарного отношения.

$C_R =$

Алгебра логики и логические схемы

Таблица истинности для функции \oplus (сложение по модулю 2) имеет вид:

a)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

b)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

c)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

с

d) такой таблицы здесь нет.

Формула вида $(x \vee \neg y) \rightarrow (x \& y)$

a) тождественно истинна;

b) тождественно ложна;

c) имеет определенное значение на всех наборах переменных.

Логическая схема высказывания $\underline{a \rightarrow b}, a$

b

a) тождественно истинна;

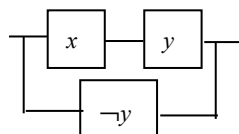
b) тождественно ложна.

Формула $\neg x_1 \neg x_2 \vee x_1 x_2 \vee x_1 \neg x_2 \vee \neg x_1 x_2$

a) СДНФ логической функции константа 1;

b) КНФ логической функции;

c) ДНФ логической функции константа 1.



Какая логическая функция соответствует схеме

a) $\neg x \vee y \vee x \& \neg y$;

b) $x \rightarrow y \vee \neg y$;

c) $x \vee y \& \neg y$;

d) нет.

Указать какие формулы есть законы:

a) $\neg(x \vee y) \equiv \neg x \& \neg y$;

b) $x \vee x \equiv 0$;

c) $\neg(x \& y) \equiv \neg x \vee \neg y$.

15. Для операций $\&$ и \vee в выражении $x_1 \vee (x_2 \vee x_3) = (x_1 \vee x_2) \vee x_3$ записано свойство

a) ассоциативность;

b) коммутативность;

c) дистрибутивность слева;

d) нет.

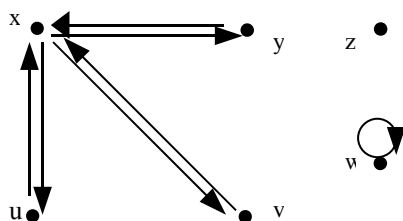
16. Для функции 2-х переменных записать ее СДНФ и СКНФ:

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Теория графов

17.

G:



Граф G ориентированный, с противоположными дугами и одной изолированной вершиной;

Граф G ориентированный, симметричный;

Граф G ориентированный, с противоположными дугами и одной голой вершиной.

Граф G ориентированное корневое дерево.

18.

Полустепень захода вершины x ($\deg_+ x$) в графе G равна:

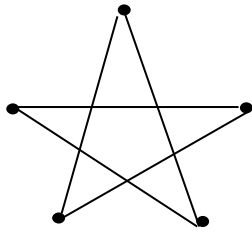
a) 5;

b) 6;

c) 3;

d) 0.

19.
W:



а) Матрица смежности для графа W:

$$A(W) = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

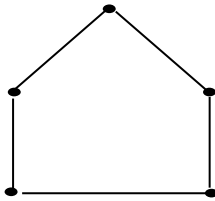
б) Матрица достижимости для графа W:

$$R(W) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

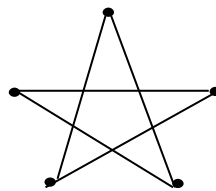
с) Матрица инцидентности для графа W:

$$B(W) = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

19.
Графы G_1 и G_2 изоморфны:
 G_1



G_2



а) да;
б) нет.

Федеральное агентство по образованию РФ
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО «АмГУ»)

Тестовые задания
по проверке остаточных знаний
по дисциплине «Дискретная математика»
(специальности 230201, 230102)
20 заданий
время тестирования 90 минут

Вариант 2

Выполнил: студент факультета математики и информатики

фамилия, имя и отчество

Курс _____ Группа № _____
Дата тестирования: _____

Результат _____%

Оценка _____

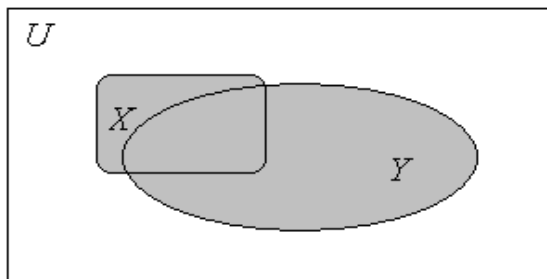
Вариант 2.

Теория множеств

1. Определить какие из перечисленных множеств пусты:

- a) $(A \cap B) \cup \bar{(A \cup B)}$;
- b) $B \setminus (A \cap B)$;
- c) $\bar{(X \cup Y)} \cap (X \cup Y)$;
- d) все множества непустые.

2. На диаграмме Эйлера-Венна изображено:



- c) $P_{X \cup Y} = \{x \mid x \in X \text{ или } x \in Y\}$;
- d) $P_{X \cap Y} = \{x \mid x \in X \text{ и } x \in Y\}$;
- e) $P_{X \setminus Y} = \{x \mid x \in X \text{ и } x \notin Y\}$;
- f) $P = \{x \mid x \in X\}$;
- e) $P_{\bar{X}} = \{x \mid x \in U \text{ и } x \notin X\}$.

3. Для отношения $R = \langle a + b \text{ четное число} \rangle$ имеют место следующие свойства:

- a) рефлексивность, симметричность, транзитивность;
- b) антирефлексивность, несимметричность, транзитивность;
- c) антирефлексивность, антисимметричность, нетранзитивность;
- d) верных свойств нет.

4. Указать какие отношения являются отношениями строгого порядка:

- a) $R1 \langle x - y > 0 \rangle$;
- b) $R2 \langle \text{жить этажом ниже} \rangle$;
- c) $R3 \langle \text{равночисленность конечных множеств} \rangle$.

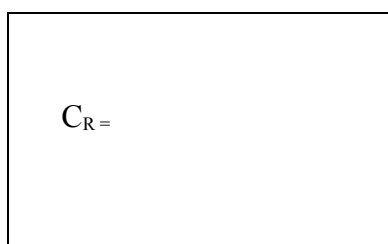
5. Верно ли утверждение: «Между множеством двоичных векторов разрядности n и множеством логических функций от n переменных существует взаимно-однозначное соответствие».

- a) да;
- b) нет.

6. Какое тождество, из перечисленных является тождеством де-Моргана?

- a) $(X \cap \bar{Y}) \cup Y = X \cup Y$;
- b) $\bar{(X \cap Y)} = \bar{X} \cup \bar{Y}$;
- c) $(X \cap \bar{Y}) \cup (X \cap Y) = X$;
- d) такого тождества нет.

7. Для бинарного отношения $R = \langle x + y = 4 \rangle$, заданного на конечном множестве $V = \{1, 2, 3, 4\}$ построить матрицу бинарного отношения.



$C_R =$

Алгебра логики и логические схемы

8. Таблица истинности для функции \equiv (эквиваленция) имеет вид:

a)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

b)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

c)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

d)

d) такой таблицы здесь нет.

9. Формула вида $(x \vee \neg x) \rightarrow (x \& \neg x)$

a) тождественно истинна;

b) тождественно ложна;

c) имеет определенное значение на всех наборах переменных.

10. Каково количество логических функций 3-х переменных:

a) 2^8 ;

b) 3^2 ;

c) 3^3 ;

d) нет такой формулы.

11. Логическая схема высказывания $\frac{a \vee b, \neg a}{b}$

a) тождественно истинна;

b) тождественно ложна.

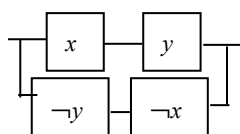
12. Формула $(\neg x_1 \vee x_2) \& (x_1 \vee x_2) \& (\neg x_1 \vee \neg x_2)$

a) СДНФ логической функции;

b) СКНФ логической функции;

c) КНФ логической функции.

13. Какая логическая функция соответствует схеме



a) $\neg x \vee y \vee x \& \neg y$;

b) $x \sim y$;

c) $x \vee y \& \neg y$.

14. Указать эквивалентное выражение для $x \oplus y$:

- a) $\neg x \vee \neg y$;
- b) $x \sim y$;
- c) $x \vee y \& \neg y$.

15. Для операций $\&$ и \vee в выражении $x_1 \& x_2 = x_2 \& x_1$ записано свойство

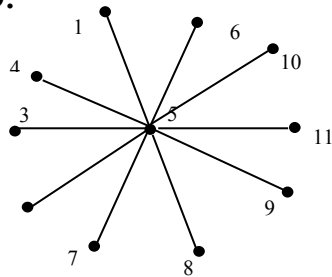
- a) ассоциативность;
- b) коммутативность;
- c) дистрибутивность слева;
- d) нет.

16. Для функции 2-х переменных записать ее СДНФ и СКНФ:

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Теория графов.

17. D:



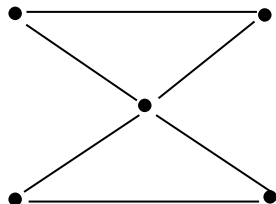
- Граф D неориентированный, полный, регулярный граф;
- Граф D двудольный граф $K_{5,1}$, дерево, звездный граф;
- Граф D двудольный граф $K_{5,1}$.

18.

Степень вершины 5 (deg 5) в графе D равна:

- a) 10;
- b) 6;
- c) 11;
- d) 2.

19. W:



Для графа W матрица смежности имеет вид:

- a)
$$A(W) = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

b)

$$A(W) = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

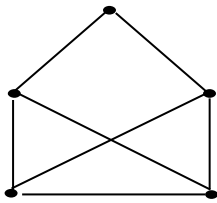
c)

$$A(W) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

20.

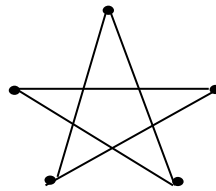
Графы G_1 и G_2 изоморфны:

G_1



нет;
да.

G_2



Федеральное агентство по образованию РФ
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО «АмГУ»)

Тестовые задания
по проверке остаточных знаний
по дисциплине «Дискретная математика»
(специальности 230201, 230102)
20 заданий
время тестирования 90 минут

Вариант 3

Выполнил: студент факультета математики и информатики

фамилия, имя и отчество

Курс _____ Группа № _____
Дата тестирования: _____

Результат _____%

Оценка _____

Вариант 3.

Теория множеств

Определить какие из перечисленных множеств пустые:

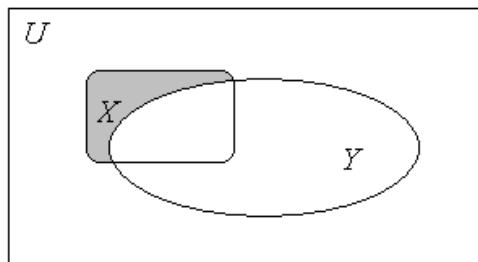
a) $(A \cup B) \setminus (A \cap B)$;

b) $B \cup (A \cap B)$;

c) $\overline{(X \cap Y) \cap (X \cup Y)}$;

все множества непустые.

На диаграмме Эйлера-Венна изображено:



a) $P_{X \cup Y} = \{x \mid x \in X \text{ или } x \in Y\}$;

b) $P_{X \cap Y} = \{x \mid x \in X \text{ и } x \in Y\}$;

c) $P_{X \setminus Y} = \{x \mid x \in X \text{ и } x \notin Y\}$;

d) $P = \{x \mid x \in X\}$;

e) $P_{\overline{X}} = \{x \mid x \in U \text{ и } x \notin X\}$.

Для отношения $R = \text{«}a \text{ делитель } b\text{»}$ имеют место следующие свойства:

a) рефлексивность, симметричность, транзитивность;

b) антирефлексивность, несимметричность, транзитивность;

c) рефлексивность, антисимметричность, транзитивность;

d) верных свойств нет.

Указать какие отношения являются отношениями нестрогого порядка:

a) R_1 «быть братом»;

b) R_2 «жить этажом выше»;

c) R_3 «равночисленность конечных множеств»;

d) нет верных вариантов.

Верно ли утверждение: «Между множеством двоичных векторов разрядности n и множеством из n наборов логических переменных логической функции существует взаимно-однозначное соответствие».

a) да;

b) нет.

Какое тождество, из перечисленных является тождеством Порецкого?

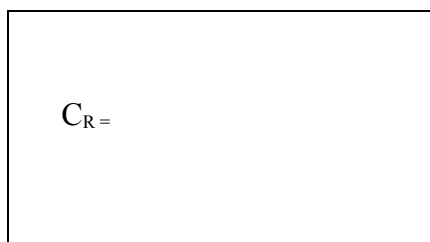
a) $(X \cap \overline{Y}) \cup Y = X \cup Y$;

b) $\overline{(X \cup Y)} = \overline{X} \cap \overline{Y}$;

c) $(X \cap \overline{Y}) \cup (X \cap Y) = X$;

d) такого тождества нет.

7. Для бинарного отношения $R = \text{«}x+y \text{ кратно } 3\text{»}$, заданного на конечном множестве $V = \{1, 2, 3, 4\}$ построить матрицу бинарного отношения.



8. Таблица истинности для функции \rightarrow (импликация) имеет вид:

a)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

b)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

c)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

e

d) такой таблицы здесь нет.

9. Формула вида $\neg(x \& y) \sim (\neg x \vee \neg y)$

a) тождественно истинна;

b) тождественно ложна;

c) имеет определенное значение на всех наборах переменных.

10. Каково количество логических бинарных операций:

a) $3!$;

b) 2^2 ;

c) 16;

d) нет такой формулы.

Логическая схема высказывания $\frac{a \vee b, a \rightarrow b}{b}$

a) тождественно истинна;

b) тождественно ложна.

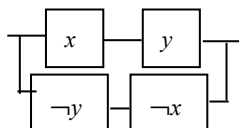
Формула $(\neg x_1 x_2) \vee (x_1 x_2) \vee (\neg x_1 \neg x_2)$

a) СДНФ логической функции;

b) СКНФ логической функции;

c) КНФ логической функции.

Какая логическая функция соответствует схеме



a) $\neg(x \oplus y)$;

b) $\neg(x \sim y)$;

c) $\neg(x \rightarrow y)$;

d) нет.

14. Указать эквивалентное выражение для $x \& y$:

- a) $\neg x y \vee x \neg y$;
- b) $x \sim y$;
- c) $(x \vee y)(x \vee \neg y)(\neg x \vee y)$.

15. Для операций $\&$ и \vee в выражении $x_1 \vee (x_2 \& x_3) = x_1 \& x_2 \vee x_1 \& x_3$ записано свойство:

- a) ассоциативность;
- b) коммутативность;
- c) дистрибутивность слева;
- d) нет.

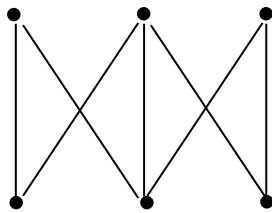
16. Для функции 2-х переменных записать ее СДНФ и СКНФ:

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

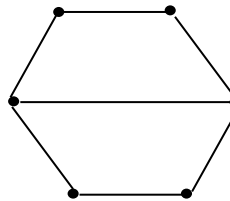
Теория графов

17. Для графа V его правильной реализацией на плоскости (в \mathbf{R}^2) является граф W:

V:

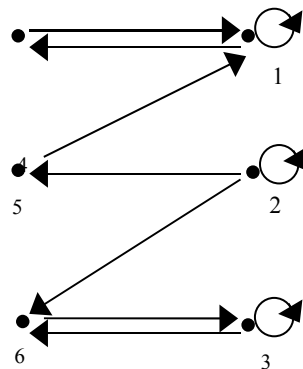


W:



- a) да;
- b) нет.

18. G:



- a) Граф G ориентированный, с противоположными дугами и одной изолированной вершиной;
- b) Граф G ориентированный, симметричный, три вершины инцидентны петлям;
- c) Граф G ориентированный, с противоположными дугами и тремя петлевыми дугами;
- d) Граф G ориентированный, двудольный граф;
- e) Граф G ориентированный, с противоположными дугами.

19.

Полустепень исхода вершины 4 (\deg_4) в графе G равна:

- a) 1;
- b) 6;
- c) 3;
- d) 2.

20. Полный ориентированный граф K_n с n вершинами без петель, кратных дуг, у которого любые две вершины инцидентны дуге имеет:

- $(n-1)!$ дуг;
- n^2-n дуг;
- $(n^2-n)/2$ дуг.

Федеральное агентство по образованию РФ
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО «АмГУ»)

Тестовые задания
по проверке остаточных знаний
по дисциплине «Дискретная математика»
(специальности 230201, 230102)
20 заданий
время тестирования 90 минут

Вариант 4

Выполнил: студент факультета математики и информатики

фамилия, имя и отчество

Курс _____ Группа № _____
Дата тестирования: _____

Результат _____%

Оценка _____

Вариант 4.

Теория множеств

2. Определить какие из перечисленных тождеств верны:

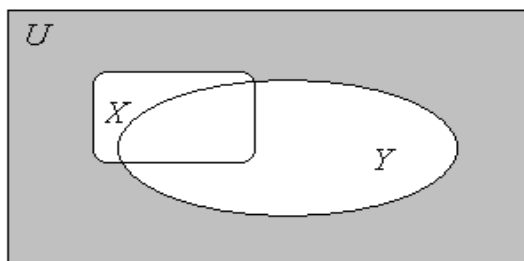
a) $(A \cup B) \setminus (A \cap B) = (A \setminus B) \cup (B \setminus A)$;

b) $B \cup (A \cap B) = B \cup A$;

c) $\neg X \cap \neg Y \cap \neg Z = \neg (X \cup Y \cup Z)$;

все тождества неверны.

3. На диаграмме Эйлера-Венна изображено:



a) $P_{X \cup Y} = \{x \mid x \in X \text{ или } x \in Y\}$;

b) $P_{X \cap Y} = \{x \mid x \in X \text{ и } x \in Y\}$;

c) $P_{X \setminus Y} = \{x \mid x \in X \text{ и } x \notin Y\}$;

d) $P = \{x \mid x \in X\}$;

e) $P_{\neg X} = \{x \mid x \in U \text{ и } x \notin X\}$;

f) нет формулы.

3. Для отношения $R = \text{«учиться в одном вузе»}$ имеют место следующие свойства:

a) рефлексивность, симметричность, транзитивность;

b) антирефлексивность, несимметричность, транзитивность;

c) антирефлексивность, несимметричность, нетранзитивность;

d) верных свойств нет.

4. Указать какие отношения являются отношениями строгого порядка:

a) R_1 «быть братом»;

b) R_2 «жить в одном городе»;

c) R_3 «a делитель b»;

d) Нет верных вариантов.

5. Верно ли утверждение: «Между множеством целых чисел и множеством всех положительных четных чисел существует взаимно-однозначное соответствие».

a) да;

b) нет.

Какое тождество, из перечисленных является тождеством поглощения?

a) $(X \cap Y) \cup Y = X \cup Y$;

b) $(X \cup Y) \cap Z = (X \cap Z) \cup (Y \cap Z)$;

c) $(X \cap Y) \cup (X \cap Y) = X$;

d) такого тождества нет.

7. Для бинарного отношения $R = \text{«x делится на y без остатка»}$, заданного на конечном множестве $V = \{1, 2, 3, 4\}$ построить матрицу бинарного отношения.

| |
|---------|
| $C_R =$ |
|---------|

8. Таблица истинности для функции \vee (дизъюнкция) имеет вид:

a)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

b)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

c)

f

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

d) такой таблицы здесь нет.

9. Формула вида $\neg(x \& y) \rightarrow (\neg x \& \neg y) \sim (x \& y) \vee (\neg x \& \neg y)$

a) тождественно истинна;

b) тождественно ложна;

c) имеет определенное значение на всех наборах переменных.

10. Каково количество логических функций 3-х переменных:

a) $3!$;

b) n^3 ;

c) 3^n ;

d) нет такой формулы.

11. Логическая схема высказывания $\frac{a}{b \rightarrow a}$

a) тождественно истинна;

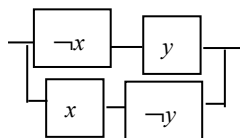
b) тождественно ложна.

12. Формула $(\neg x_1 x_2) \vee (x_1 x_2) \vee (\neg x_1 \neg x_2)$

a) СДНФ логической функции;

b) СКНФ логической функции;

КНФ логической функции.



13. Какая логическая функция соответствует схеме

a) $\neg(x \oplus y)$;

b) $\neg(x \sim y)$;

c) $\neg(x \& y)$.

14. Указать эквивалентное выражение для $x|y$ (| штрих Шеффера $f(x,y) = (1110)$):

- a) $\neg x \vee \neg y$;
- b) $x \sim y$;
- c) $(x \vee y)(x \vee \neg y)(\neg x \vee y)$;
- d) эквивалентных выражений нет.

15. Для операций $\&$ и \vee в выражении $x_1 \& (x_2 \vee x_3) = (x_1 \& x_2) \vee x_3$ записано свойство

- a) ассоциативность;
- b) коммутативность;
- c) дистрибутивность слева;
- d) нет.

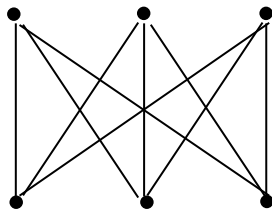
16. Для функции 2-х переменных записать ее СДНФ и СКНФ:

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

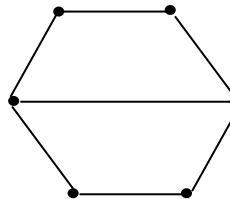
Теория графов

17. Для графа V его правильной реализацией на плоскости (в \mathbf{R}^2) является граф W:

V:

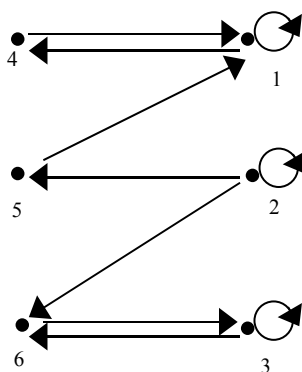


W:



- a) да;
- b) нет;
- c) правильная реализация для графа V в \mathbf{R}^2 не существует.

18. G:



a) Матрица смежности для графа G:

$$A(G) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

b) Матрица достижимости для графа G:

$$R(W) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

c) Матрица инцидентности для графа G:

$$B(G) = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

d) нет матричного представления для графа G.

19.

Степень вершины 5 ($\deg 5$) в графе G равна:

- a) 1;
- b) 2;
- c) 3;
- d) 0.

20. Полный неориентированный граф K_n с n вершинами без петель, кратных дуг, у которого любые две вершины инцидентны дуге имеет:

- a) $(n-1)!$ дуг;
- b) n^2-n дуг;
- c) $(n^2-n)/2$ дуг;
- d) верной формулы нет.

Федеральное агентство по образованию РФ
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО «АмГУ»)

Тестовые задания
по проверке остаточных знаний
по дисциплине «Дискретная математика»
(специальности 230201, 230102)
20 заданий
время тестирования 90 минут

Вариант 5

Выполнил: студент факультета математики и информатики

фамилия, имя и отчество

Курс _____ Группа № _____
Дата тестирования: _____

Результат _____%

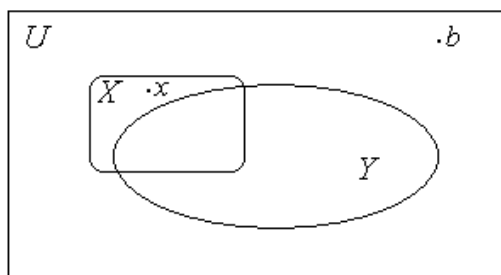
Оценка _____

Вариант 5.

Теория множеств

1. Проверить всегда ли справедливо тождество $(A \setminus B) \cup C = (A \cup C) \setminus (B \cup C)$:
 а) нет;
 да.

2. На диаграмме Эйлера-Венна изображено:



$$P_{X \cup Y} = \{x \mid x \in X \text{ или } x \in Y\};$$

$$P_{X \cap Y} = \{x \mid x \in X \text{ и } x \in Y\};$$

$$P_{X \setminus Y} = \{x \mid x \in X \text{ и } x \notin Y\};$$

$$P = \{x \mid x \in X\};$$

$$е) P_{\setminus X} = \{x \mid x \in U \text{ и } x \notin X\}.$$

3. Для отношения $R = \text{«а отец б»}$ имеют место следующие свойства:

- а) рефлексивность, симметричность, транзитивность;
 б) антирефлексивность, несимметричность, транзитивность;
 в) антирефлексивность, несимметричность, нетранзитивность;
 верных свойств нет.

4. Указать какие отношения являются отношениями эквивалентности:

- а) R_1 «быть братом»;
 б) R_2 «жить в одном городе»;
 в) R_3 «а делитель б»;
 г) Нет верных вариантов.

5. Верно ли утверждение: «Между множеством целых чисел и множеством всех вещественных чисел отрезка $(0, 1]$ существует взаимно-однозначное соответствие».

- а) да;
 б) нет.

6. Какое число подмножеств содержит множество U , $|U|=n$:

- а) $n!$;
 б) n^2 ;
 в) n ;
 г) нет такой формулы.

7. Какое тождество, из перечисленных является тождеством идемпотентности?

$$(X \cap Y) \cup Y = X \cup Y;$$

$$(X \cup Y) \cap Z = (X \cap Z) \cup (Y \cap Z);$$

$$(X \cap Y) \cup (X \cap Y) = X;$$

такого тождества нет.

8. Для бинарного отношения $R = \text{«x делится на y с остатком 1»}$, заданного на конечном множестве $V = \{1, 2, 3, 4\}$ построить матрицу бинарного отношения.

$C_R =$

9. Таблица истинности для функции $|$ (штрих Шеффера) имеет вид:

a)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

b)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

c)

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

d) такой таблицы здесь нет.

10. Формула вида $(x \& y) \sim (\neg(\neg x \vee \neg y))$

- a) тождественно истинна;
- b) тождественно ложна;
- c) имеет определенное значение на всех наборах переменных.

11. Логическая схема высказывания $\frac{a \vee b, a \rightarrow b}{a}$

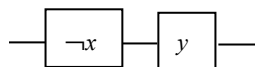
- a) тождественно истинна;
- b) тождественно ложна.

12. Формула $(\neg x_1 x_2) \vee (x_1 x_2) \vee \neg x_2$

- a) СДНФ логической функции;
- b) СКНФ логической функции;
- c) КНФ логической функции;

h) такой формы нет.

13. Какая логическая функция соответствует схеме



- a) $(x \oplus y)$;
- b) $\neg(x \vee \neg y)$;
- c) $(x \rightarrow y)$;
- d) нет.

14. Указать эквивалентное выражение для $x \rightarrow y$:

- a) $\neg x y \vee x \neg y$;
- b) $\neg(x \sim y)$;
- c) $\neg x \neg y \vee \neg x y \vee x y$;
- d) $x y \vee \neg x \neg y$;
- e) нет.

15. Для операций $\&$ и \vee в выражении $x_1 \vee (x_2 \vee x_3) = x_1 \vee x_2 \vee x_1 \vee x_3$ записано свойство

- a) ассоциативность;
- b) коммутативность;
- c) дистрибутивность слева;
- d) нет.

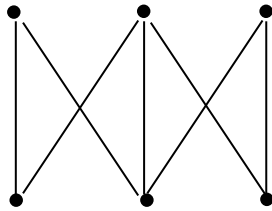
16. Для функции 2-х переменных записать ее СДНФ и СКНФ:

| x1 | x2 | f |
|----|----|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

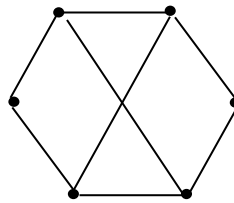
Теория графов

17. Для графа V его правильной реализацией на плоскости (в \mathbf{R}^2) является граф W:

V:

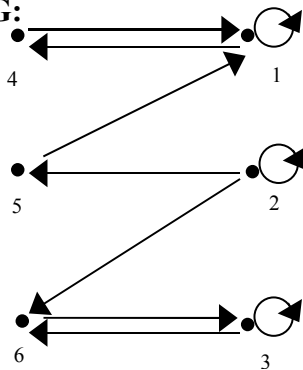


W:



- a) да;
- b) нет.

18. G:



- a) Граф G ориентированный, с противоположными дугами и одной изолированной вершиной;
- b) Граф G ориентированный, симметричный, три вершины инцидентны петлям;
- c) Граф G ориентированный, с противоположными дугами и тремя петлевыми дугами;
- d) Граф G ориентированный, двудольный граф;
- e) Граф G ориентированный, с противоположными дугами.

19.

Полустепень исхода вершины 3 (\deg_3) в графе G равна:

- a) 1;
- b) 6;
- c) 3;
- d) 2.

20. Полный двудольный ориентированный граф $K_{n,m}$ без петель, кратных дуг, у которого любые две вершины инцидентны дуге имеет:

- a) $2nm$ дуг;
- b) n^2-n дуг;
- c) $(n^2-m)/2$ дуг.

13. КАРТА КАДРОВОЙ ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ

| | | |
|-----------------------------------|----------------|------------------|
| Лектор | к.т.н., доцент | Семичевская Н.П. |
| Руководитель практических занятий | к.т.н., доцент | Семичевская Н.П. |

Наталья Петровна СЕМИЧЕВСКАЯ
доцент кафедры информационных управляющих систем АмГУ

Редактор О.К. Мамонтова

Учебно-методический комплекс дисциплины
ДИСКРЕТНАЯ МАТЕМАТИКА

Издательство АмГУ. Подписано к печати _____.____.07. Формат _____. Усл.
печ.л. _____, уч.-изд. л. _____. Тираж _____. Заказ _____.