ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

Кафедра «Информатика и прикладная математика»

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Б1. В. ДВ.12.02

МАТЕМАТИЧЕСКАЯ ЛОГИКА И ТЕОРИЯ АЛГОРИТМОВ

Направление подготовки: 27.03.04 Управление в технических системах

Профиль: Интеллектуальные системы обработки информации и управления

Квалификация: бакалавр

Форма обучения: очная

СОДЕРЖАНИЕ

1. КОНСПЕКТ ЛЕКЦИЙ

1. 1 Лекция №1 (2 часа).

Тема: «Основные операции алгебры высказываний. Формулы алгебры высказываний»

1.1.1 Вопросы лекции:

1. Основные понятия алгебры высказываний.
2. Основные операции алгебры высказываний.

1.1.2 Краткое содержание вопросов:

1. Основные понятия алгебры высказываний.
2. Основные операции алгебры высказываний.

Основные операции алгебры высказываний. Таблицы истинности.

Рассмотрим логические операторы.

1) Оператор, соответствующий союзу "и", называется *конъюнкцией*. Высказывание *A&B* истинно тогда и только тогда, когда истинны оба высказывания *A* и *B* (табл. 1).

Таблица 1

A	B	A & B
И	И	И
И	Л	Л
Л	И	Л
Л	Л	Л

2) Оператор, соответствующий частице "не", называется *отрицание* (табл.2.)

Таблица 2

A	¬A
И	Л
Л	И

Эта операция одноместна – в том смысле, что из одного данного простого высказывания строится новое высказывание *¬A*.

3) Операция, соответствующая союзу "или", называется *дизъюнкцией* (табл.3).

Таблица 3

A	B	A V B
И	И	И
И	Л	И
Л	И	И
Л	Л	Л

Абсолютная истинность *A∨B* означает, что в каждой ситуации, хотя бы одно из высказываний − истинно.

4) Операция, соответствующая обороту "если..., то...", называется *импликацией*. *A* называется посылкой импликации, *B* – её заключением (табл.4).

Таблица 4

A	B	A → B
И	И	И
И	Л	Л
Л	И	И
Л	Л	И

В случае импликации несоответствие между обычным пониманием истинности и идеализированной точкой зрения алгебры высказываний еще заметнее, чем для других логических операций. Главное отличие в том, что при учете смыслового содержания высказываний оборот "если..., то..." подразумевает причинную связь между посылкой и заключением. С точки же зрения алгебры высказываний истинность импликации в некоторой ситуации означает лишь, что если в этой ситуации истинна посылка, то истинно заключение.

В результате истинными могут оказаться импликации "если в доме пять этажей, то в квартире № 3 живет Иванов", "если в Воронеже идет дождь, то книга серого цвета".
5) Операция, соответствующая оборотам типа "тогда и только тогда", "для того чтобы...", "необходимо и достаточно", называется *эквивалентностью* (табл.5). К эквивалентности в той же мере относится замечание о том, что её использование в алгебре высказываний не учитывает смыслового содержания высказываний.

Таблица 5

A	B	A ~ B
И	И	И
И	Л	Л
Л	И	Л
Л	Л	И

Таким образом, имеется некоторое количество логических операций, позволяющих получать из простых высказываний сложные. При этом вместо простых высказываний можно брать сложные, уже построенные. Т.е. появляется возможность применять многоступенчатые конструкции, многократно используя введенные логические операции.

1. 2 Лекция №2 (2 часа).

Тема: «Булевы функции. Элементарные булевы функции. Представление булевых функций формулами**»**

1.3.1 Вопросы лекции:

1. Булевы функции. Элементарные булевы функции.
2. Представление булевых функций формулами

1.3.2 Краткое содержание вопросов:

1. Булевы функции. Элементарные булевы функции.
2. Представление булевых функций формулами

1. Булевы функции. Элементарные булевы функции.

Булевы функции, булевы константы. **Булевыми функциями (или функциями алгебры логики или истинностными функциями) называются функции, значения которых равны 0 или 1 и аргументы которых принимают только два значения 0 и 1.** Булевы функции могут быть заданы специальными таблицами истинности или аналитически в виде специальных высказывательных форм, называемых иногда булевыми формами. Выражения, содержащие одну или несколько переменных (аргументов), соединенных знаками логических операций, называются *логическим формами*. Высказывания, не содержащие ни одной переменной, называются константами. В логике, в отличие от арифметики, только две константы 0 - false и 1- true.

Напомним, что *форма называется числовой,* если при допустимом значении своих аргументов, она обозначает число (является числом). Булева форма является частным случаем числовой формы. Т.о. при помощи суперпозиции, исходя из логических операций над логическими переменными, можно строить сложные составные высказывания и затем вычислять их. Такого рода составные высказывания являются частным случаем так называемых булевых функций, которые являются предметом изучения математической логики. Обобщая все сказанное, можно дать определение булевых функций:

Булевыми функциями, называются предикаты, все аргументы которых определены на множестве {0, 1}, интерпретируемые как {ложь, истина}.

Можно сказать, что понятие булевой функции является частным случаем понятия предиката. Отличие состоит лишь в том, что у булевой функции четко фиксирована как область определения {0, 1}, так и область значений функции {0, 1}, в то время как у пре-

диката четко фиксирована только одна область значений {0, 1}, в то время как область определения задана произвольным множеством.

В свою очередь понятие предиката является частным случаем понятия функции, отличие состоит в том, что у предиката четко фиксирована область значений {0, 1}, а у функции это может быть вся числовая ось.

2. Представление булевых функций формулами.

Булевы функции и формулы. *ФАЛ называются также* булевыми *функциями,* двоичными *функциями или* переключательными *функциями. Аргументы булевой функции являются булевыми переменными. Булеву функцию можно задать таблицей истинности.*

Утверждение Для булевой функции от n аргументов существует 2^n различных наборов аргументов.

Булева функция $f(x_1, x_2, \ldots, x_n)$ называется *полностью определенной*, если ее значения определены на всех 2^n наборах переменных. В противном случае функция *частично определенная*.

Функция $f(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_n)$ *существенно зависит от переменной* x_i, (или переменная x_i – *существенная*), если $\exists$ такой набор значений $x_1, x_2, \ldots, x_n$ $(\sigma_1,\ldots,\sigma_{i-1},\sigma_i,\sigma_{i+1},\ldots,\sigma_n)$, что

$f(\sigma_1,\ldots,\sigma_{i-1},0,\sigma_{i+1},\ldots,\sigma_n) \neq f(\sigma_1,\ldots,\sigma_{i-1},1,\sigma_{i+1},\ldots,\sigma_n)$. В противном случае переменная x_i – *несущественная (фиктивная)*.

x_1	x_2	f_1	f_2
0	0	0	1
0	1	0	1
1	0	1	0
1	1	1	0

Пусть две булевы функции заданы таблицей истинности. Для них переменная x_1 существенная, а x_2 – несущественна. По определению булевы функции равны, если одна из другой получается введением или удалением несущественных переменных.

Одна и та же функция может иметь множество реализаций формулами над данным базисом (т.е. множеством логических операций). Формулы, реализующие одну и ту же функцию, называются *равносильными* (т.е. на всех наборах переменных их значение истинности совпадает). Отношение равносильности формул является отношением эквивалентности.

Формулы алгебры логики, при образовании которых используются только операции отрицания, конъюнкции и дизъюнкции, называются *булевыми формулами*.

Для любой формулы алгебры логики существует равносильная ей булева формула.

Способы представления булевых функций. Нормальные формы

Табличный способ определения истинности сложного выражения имеет ограниченное применение, т.к. с увеличением числа логических переменных число вариантов становится слишком большим. Тогда может быть использован способ приведения формул к *нормальной форме*.

Аналитическое выражение функции (или формула) находится в *нормальной форме*, если в ней отсутствуют знаки эквивалентности, импликации, двойного отрицания, а знаки отрицания находятся только при переменных.

Элементарной дизъюнкцией (*произведением*) называется дизъюнкция (произведение) переменных или их отрицаний, в котором каждая переменная встречается только один раз.

ДНФ – это дизъюнкция элементарных произведений. *КНФ* – это произведение элементарных дизъюнкций. Как ДНФ, так и КНФ функции не единственна. Обычно предполагают, что входящие в ДНФ (КНФ) элементарные конъюнкции (дизъюнкции) попарно различны.

ДНФ (КНФ) называется *совершенной*, если каждая переменная формулы входит в каждую элементарную конъюнкцию (дизъюнкцию) ровно один раз.
СДНФ (СКНФ) функции единственна.

Элементарные дизъюнкции: $x \vee \bar{y}$, z. Элемент. конъюнкции: $x \cdot \bar{y} \cdot z$, x.
$f(x,y,z) = xyz \vee \bar{x}y$ – ДНФ ; $f(x,y,z) = (x \vee \bar{y}) \cdot z$ – КНФ.

Введем обозначения: $x^{\alpha} = \begin{cases} x, & \alpha = 1 \\ \bar{x}, & \alpha = 0 \end{cases}$

О разложении булевой функции по k переменным (знак $\cup \equiv \vee$**).**

$$f(x_1,\ldots,x_n) = \bigvee_{\alpha_1,\ldots,\alpha_k} x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_k^{\alpha_k} \cdot f(\alpha_1,\alpha_2,\ldots,\alpha_k,x_{k+1},\ldots,x_n)$$

n=3, k=2.

Доказательство:
Выберем какой-либо набор значений для переменных $x_1, \ldots, x_n$. Пусть это будет $\sigma_1, \ldots, \sigma_n$..
Заметим, что $\sigma_i^{\alpha_i} = \begin{cases} 1, & \sigma_i = \alpha_i \\ 0, & \sigma_i \neq \alpha_i \end{cases}$ ($1^1=1$, $0^0=1$, $1^0=\neg 1=0$, $0^1=0$)
Подставим в правую часть формулировки теоремы вместо $x_1, \ldots, x_n$ набор $\sigma_1, \ldots, \sigma_n$. Получим $\bigvee_{(\alpha_1,\ldots,\alpha_k)} \sigma_1^{\alpha_1}\sigma_2^{\alpha_2}\ldots\sigma_k^{\alpha_k} \cdot f(\alpha_1,\alpha_2,\ldots,\alpha_k,\sigma_{k+1},\ldots,\sigma_n)$. Поскольку коэффициент перед функцией равен 1 только при равных значениях σ_i и α_i, в разложении останется только один член: $\sigma_1^{\alpha_1}\ldots\sigma_k^{\alpha_k} \cdot f(\alpha_1,\ldots,\alpha_k,\sigma_{k+1},\ldots,\sigma_n)$, и $\sigma_i=\alpha_i$, т.е. $f(\sigma_1,\ldots,\sigma_k,\sigma_{k+1},\ldots,\sigma_n)$. Получена левая часть формулы теоремы 4.6. Поскольку набор был выбран произвольно, получаем, что утверждение верно $\forall$ набора $x_1, \ldots, x_n$. ■

Следствие 1: Разложение Шеннона

$$f(x_1,x_2,\ldots,x_n) = \bar{x}_1 \cdot f(0,x_2,\ldots,x_n) \vee x_1 \cdot f(1,x_2,\ldots,x_n)$$

Следствие 2: При k=n получаем: $f(x_1,\ldots,x_n) = \bigvee_{f=1} x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$, т.е. выбираем те слагаемые, на которых функция равна 1. Полученная формула представляет собой *СДНФ*.

Построение совершенных нормальных форм

Построение СДНФ

1. Построение по ТИ.

Найти строки в ТИ, где $f = 1$.

1) ∀ найденному набору $\sigma_1, ..., \sigma_n$. поставить в соответствие произведение $\tilde{x}_1 \cdot \tilde{x}_2 \cdot \ldots \cdot \tilde{x}_n$, где $\tilde{x}_i = \begin{cases} x_i, & \text{если } \sigma_i = 1 \\ \overline{x_i}, & \text{если } \sigma_i = 0 \end{cases}$

2) Составить дизъюнкцию из произведений п.2.

2. Получение из ДНФ.

Если некоторое произведение ДНФ не содержит какой-либо переменной, то необходимо помножить это произведение на дизъюнкцию этой переменной и ее отрицания и применить дистрибутивный закон.

Построение СКНФ

1. Построение по ТИ.

Найти строки в ТИ, где $f = 0$.

1) ∀ найденному набору $\sigma_1, ..., \sigma_n$. поставить в соответствие дизъюнкцию $\tilde{x}_1 \vee \tilde{x}_2 \vee \ldots \vee \tilde{x}_n$, где $\tilde{x}_i = \begin{cases} x_i, & \text{если } \sigma_i = 0 \\ \overline{x_i}, & \text{если } \sigma_i = 1 \end{cases}$

2) Составить произведение дизъюнкций из п.2.

2. Получение из КНФ.

Если некоторая элементарная дизъюнкция КНФ не содержит какой-либо переменной, то необходимо дизъюнктивно добавить в нее произведение этой переменной и ее отрицания и применить дистрибутивный закон.

1. 3 Лекция №3 (2 часа).

Тема: «Алгебра Буля. Минимизация булевых функций в классе ДНФ**»**

1.2.1 Вопросы лекции:

1. Алгебра Буля.
2. Минимизация булевых функций в классе ДНФ.

1.3.2 Краткое содержание вопросов:

1. Алгебра Буля.
2. Минимизация булевых функций в классе ДНФ.

Формула АВ. Равносильные формулы. Основные равносильности. Равносильные преобразования формул.

Алфавитом алгебры высказываний называется множество элементы, которого называются буквами.

Алфавит (язык исчисления высказываний) состоит из переменных высказываний *A, B, C,...;* знаков логических связок $\&, \vee, \neg, \rightarrow$ и скобок (,).

Конечные последовательности букв алфавита называются словами в этом алфавите. Некоторые слова в алфавите являются формулами алгебры высказываний.

Формулами называют логические операции, которые получаются комбинированием конечного числа введенных операций. Для всякой формулы можно построить истинностную таблицу, последовательно используя истинностные таблицы основных операций. Строгое определение формулы алгебры высказываний дается по индукции:

Формулы: а) переменное высказывание – есть формула;

б) если $\mathfrak{I}$ и $\mathfrak{R}$ – формулы, то $(\mathfrak{I}\&\mathfrak{R}), (\mathfrak{I}\vee\mathfrak{R}), (\mathfrak{I}\rightarrow\mathfrak{R}), \overline{(\mathfrak{I})}$ тоже формулы;

в) других формул нет.

.

Минимизация булевых функций в классе ДНФ

1. Основные понятия.
2. Минимизация булевых функций в классе ДНФ.

1. Понятие минимальной ДНФ, импликанты формулы, простой импликанты, сокращённой ДНФ, тупиковой ДНФ.

2. Методы отыскания сокращённой ДНФ. Метод Квайна получения минимальной ДНФ из сокращённой, другие методы.

Минимальная ДНФ – это такая ДНФ функции, которая содержит наименьшее количество вхождений переменных по сравнению с остальными.

Элементарная конъюнкция называется *импликантой функции* $f(x_1,\dots,x_n)$, если она равна 0 на тех наборах, на которых *f* обращается в 0. *Простой импликантой* называется импликанта, в которой отбрасывание любой буквы ведет к получению элементарной конъюнкции, которая не является импликантой (т.е. никакая часть простой импликанты сама импликантой не является). Каждая импликанта соответствует покрытию на карте Карно, а простая импликанта – покрытию наибольшей размерности.

1) $f(x_1,x_2,x_3)=x_1x_2\vee\overline{x_1}\,\overline{x_2}\,\overline{x_3}\vee x_1x_2x_3$. $\overline{x_1}\,\overline{x_2}\,\overline{x_3}$ – импликанта, причем простая; $x_1x_2x_3$ – импликанта, но не простая, т.к. удаление x_3 снова дает импликанту x_1x_2 (которая является простой).

2) Найдем импликанты и простые импликанты для функции $f(x_1,x_2)=x_1\to x_2$. Всего имеется 8 элементарных конъюнкций с переменными x_1, x_2. Приведем их таблицы истинности.

x_1	x_2	$x_1\to x_2$	$\overline{x}_1\overline{x}_2$	$\overline{x}_1x_2$	$x_1\overline{x}_2$	x_1x_2	$\overline{x}_1$	$\overline{x}_2$	x_1	x_2
0	0	1	1	0	0	0	1	1	0	0
0	1	1	0	1	0	0	1	0	0	1
1	0	0	0	0	1	0	0	1	1	0
1	1	1	0	0	0	1	0	0	1	1

Из таблицы истинности заключаем, что $\overline{x}_1\overline{x}_2$, $\overline{x}_1x_2$, x_1x_2, $\overline{x}_1$, x_2 являются имплантами функции *f*. Из них простыми являются $\overline{x}_1$ и x_2.

Дизъюнкция всех простых импликантов функции называется *сокращенной ДНФ*. Сокращенная ДНФ функции единственна.

Сокращенная ДНФ может содержать лишние импликанты, удаление которых не меняет значения функции.

Если из сокращенной ДНФ удалить все лишние дизъюнктивные члены, и удаление любого из оставшихся приведет к изменению значения функции, то такая форма называется *тупиковой* ДНФ. Та из всех тупиковых ДНФ, которая имеет наименьшее число вхождений переменных, является *минимальной* ДНФ.

Процесс нахождения минимальной ДНФ из СДНФ можно разбить на следующие этапы:

1) нахождение сокращенной ДНФ (она единственна);
2) нахождение всех тупиковых ДНФ (их м.б. несколько);
3) выбор из всех тупиковых минимальной ДНФ (их тоже м.б. несколько).

Известны аналитические и графические способы построения минимальной ДНФ.

Графический способ использует представление на картах Карно.

1. 4 Лекция № 4 (2 часа).

Тема: «Полиномы Жегалкина. Представление булевых функций полиномами Жегалкин»

1.4.1 Вопросы лекции:

1. Полиномы Жегалкина.
2. Представление булевых функций полиномами Жегалкина.

1.4.2 Краткое содержание вопросов:

1. Полиномы Жегалкина.
2. Представление булевых функций полиномами Жегалкина.

Полиномы Жегалкина, представление булевых функций полиномами Жегалкина.

Алгебра Жегалкина. Булевы функции с операциями умножения и сложения по модулю 2 образуют алгебру Жегалкина.

Аксиомы алгебры Жегалкина:

1. Операции с константами: $A\cdot 1 \equiv A$; $A\cdot 0 \equiv 0$; $A \oplus 0 \equiv A$.
2. Идемпотентность: $A\cdot A \equiv A$; $A \oplus A \equiv 0$.
3. Коммутативность: $A\cdot B \equiv B\cdot A$; $A \oplus B \equiv B \oplus A$.
4. Ассоциативность: $(A \oplus B) \oplus C \equiv A \oplus (B \oplus C)$; $(A\cdot B)\cdot C \equiv A\cdot(B\cdot C)$.
5. Дистрибутивность: $A\cdot(B \oplus C) \equiv A\cdot B \oplus A\cdot C$.

Представление булевых функций полиномами Жегалкина

Можно перейти от алгебры Буля к алгебре Жегалкина, используя следующие соотношения: $A \oplus 1 \equiv \overline{A}$; $A \vee B = A \oplus B \oplus A\cdot B$.

И наоборот, от алгебры Жегалкина к алгебре Буля: $A \oplus B = \overline{A}\cdot B \vee A\cdot \overline{B}$

Перейти к выражению булевой алгебры: $(x \oplus 1)\cdot y \oplus (x \oplus 1) = \overline{x}\cdot y \oplus \overline{x} = \overline{x}y\cdot \overline{x} \vee x\cdot \overline{x}y = (x \vee \overline{y})\cdot \overline{x} \vee 0 = \overline{\overline{x}\ \overline{y}}$.

1. 5 Лекция № 5 (2 часа).

Тема: «Полные системы булевых функций, критерий полноты**»**

1.5.1 Вопросы лекции:

1. Полные системы булевых функций.
2. Критерий полноты.

1.5.2 Краткое содержание вопросов:

1. Полные системы булевых функций.
2. Критерий полноты.

Критерии полноты Поста-Яблонского

Теорема о функциональной полноте была сформулирована в 1921 г. американским ученым *Эмилем Постом* и доказана советским ученым *Яблонским С.В.*

Система булевых функций $\{f_i\}$ является полной тогда и только тогда, когда выполняются 5 условий: в этой системе существует функция f_i не сохраняющая константу нуль; существует функция f_i, не сохраняющая константу 1;существует нелинейная функция; существует не самодвойственная функция; существует функция, не являющаяся монотонной.

Построим все булевы функции от двух переменных (табл.).

X_1	X_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Индекс i функциональной переменной f_i $\overline{i=1,15}$ равен десятичному эквиваленту этой функции, читаемому сверху вниз.

$f_0(X_1, X_2) = 0$ - константа 0

$f_1(X_1, X_2) = X_1 \,\&\, X_2$ - конъюнкция

$f_2(X_1, X_2) = X_1 \overline{X}_2 = \overline{X_1 \Rightarrow X_2} = X_1 \Rightarrow X_2$ - левая коимпликация (читается “неправда, что если X_1 то X_2», префикс ко – от латинского conversus - обратный

$f_3(X_1, X_2) = X_1$

$f_4(X_1, X_2) = \overline{X}_1 X_2 = \overline{X_1 \Leftarrow X_2} = X_1 \Leftarrow X_2$ - правая коимпликация

$f_5(X_1, X_2) = X_2$

$f_6(X_1, X_2) = X_1 \oplus X_2$ - сложение по модулю 2 (неравнозначность)

$f_7(X_1, X_2) = X_1 \vee X_2$ - дизъюнкция

$f_8(X_1, X_2) = \overline{X_1 \vee X_2} = X_1 \Downarrow X_2$ - стрелка Пирса, функция Вебба

$f_9(X_1, X_2) = X_1 \Leftrightarrow X_2$ - эквивалентность

$f_{10}(X_1, X_2) = \overline{X}_2$ - отрицание X_2

$f_{11}(X_1, X_2) = \overline{X}_2 \vee X_1 = X_1 \Leftarrow X_2$ - правая импликация («если X_1 то X_2»)

$f_{12}(X_1, X_2) = \overline{X}_1$ - отрицание X_1 («если X_1 то X_2»)

$f_{13}(X_1, X_2) = X_2 \vee \overline{X}_2 = X_1 \Rightarrow X_2$ - левая импликация

$f_{14}(X_1, X_2) = \overline{X}_1 \vee \overline{X}_2 = X_1 / X_2$ - штрих Шеффера

$f_{15}(X_1, X_2) = 1$

«к- значная логика»

1. Понятие к-значной логики.
2. Функции к-значной логики

Двузначная логика допускает обобщение на k - значный случай. При этом хотя в k - значных логиках сохраняются многие результаты и свойства двузначной логики, ряд фактов принципиально отличаются от соответствующих результатов алгебры логики. Многие решённые задачи двузначной логики не имеют исчерпывающего решения в k - значных логиках, а иные и вовсе не решены.

Функция $f(x_1, x_2, ..., x_n)$ называется функцией k - значной логики, если её аргументы определены на множестве $\{0,1,2,...,k-1\}$, состоящим из k элементов, а сама функция принимает значения из того же множества.

Множество всех функций k - значной логики обозначается через P_k. Функция $f(x_1, x_2, ..., x_n)$ задана, если задана её таблица истинности. При $k = 3$ таблица истинности имеет вид

x_1	x_2	x_3	$f(x_1,x_2,x_3)$	x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	$f(0,0,0)$	1	1	2	$f(1,1,2)$
0	0	1	$f(0,0,1)$	1	2	0	$f(1,2,0)$
0	0	2	$f(0,0,2)$	1	2	1	$f(1,2,1)$
0	1	0	$f(0,1,0)$	1	2	2	$f(1,2,2)$
0	1	1	$f(0,1,1)$	2	0	0	$f(2,0,0)$
0	1	2	$f(0,1,2)$	2	0	1	$f(2,0,1)$
0	2	0	$f(0,2,0)$	2	0	2	$f(2,0,2)$
0	2	1	$f(0,2,1)$	2	1	0	$f(2,1,0)$
0	2	2	$f(0,2,2)$	2	1	1	$f(2,1,1)$
1	0	0	$f(1,0,0)$	2	1	2	$f(2,1,2)$
1	0	1	$f(1,0,1)$	2	2	0	$f(2,2,0)$
1	0	2	$f(1,0,2)$	2	2	1	$f(2,2,1)$
1	1	0	$f(1,1,0)$	2	2	2	$f(2,2,2)$
1	1	1	$f(1,1,1)$				

Так как количество k - значных наборов длины n равно k^n, то число функций от n переменных в k - значной логике равно k^{k^n}. Например, если функций двух переменных в P_2 всего 16, то в P_3 их уже 19683. Таким образом в P_k возрастают трудности по сравнению с P_2 даже с возможностью перебора функций.

1. 6 Лекция №6 (2 часа).

Тема: «Логика предикатов»

1.6.1 Вопросы лекции:

1. Предикаты и их свойства. Логические операции над предикатами.
2.Кванторные операции. Логика предикатов.

1.6.2 Краткое содержание вопросов:

1. Предикаты и их свойства. Логические операции над предикатами.
2.Кванторные операции. Логика предикатов.

Логика высказываний описывает многие важные логические законы и позволяет решать многие проблемы, однако во многих случаях средства логики высказываний оказываются недостаточными.

1. Предикаты и кванторы

Предикатом $P(x_1,...x_n)$ называется функция $P:M^n \to B$, где M – произвольное множество, а B – двоичное множество $\{0,1\}$. M – называется предметной областью предиката, а $x_1,...x_n$ – предметными переменными. 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 $P(a_1,...a_n)=1$, если и только если $(a_1,...a_n) \in R$

б) всякий предикат $P(x_1,...x_n)$ определяет отношение R, такое, что $(a_1,...a_n) \in R$, если и только если $P(a_1,...a_n)=1$.

При этом R задает область истинности предиката P. Константы 0 и 1 называют нульместными предикатами.

Например: "прямая проходит через точки А и В" – трехместный предикат, у которого предметными областями двух переменных (А и В) являются множества точек, а третьей - множество прямых.

"если тетрадь лежит в папке, а папка – в портфеле, то тетрадь лежит в портфеле" – это трехместный тождественно истинный предикат.

Поскольку предикаты принимают два значения и интерпретируются как высказывания, из них можно образовывать выражения алгебры высказываний, т.е. формулы. Элементарные формулы можно связывать операциями алгебры высказываний $\&, \vee, \to, \neg$, сохраняя за операциями те определения, которые давались в алгебре высказываний.

Кванторы

Кроме операций алгебры высказываний употребляют еще две операции, которые относятся уже не к одной фиксированной ситуации, а ко всему множеству ситуаций.

Пусть $P(x)$ – предикат, определенный на M. Высказывание "для всех x из M – $P(x)$ истинно" обозначается $\forall x P(x)$. Знак $\forall$ называется квантором общности. Высказывание "существует такой X из M, что P истинно" обозначается $\exists x P(x)$. Знак $\exists$ называется квантором существования.

Переход от P к $\forall x P(x)$ или $\exists x P(x)$ называется *связыванием* переменной x, или навешиванием квантора на переменную x. Предметную переменную, не связанную никаким квантором, называют *свободной* переменной. Смысл связанных и свободных переменных в предикатных выражениях различен. Свободная переменная – это обычная переменная, которая может принимать значения из M; P – переменное высказывание, зависящее от x. Выражение $\forall x\, P(x)$ не зависит от переменной x и при фиксированных P и M имеет вполне определенное значение. Это, в частности, означает, что переименование связанной переменной не меняет истинности выражения.

Переменные, являющиеся по существу связанными, встречаются не только в логике. В выражениях $\sum_{x=1}^{10} f(x)$ или $\int_a^b f(x)dx$ переменная x связана, при фиксированной f первое выражение становится равно определенному числу, а второе становится функцией a и b.

Навешивать кванторы можно и на многоместные предикаты и вообще на любые логические выражения, которые при этом заключаются в скобки. Навешивание квантора на многоместный предикат уменьшает в нем число свободных переменных.

2. Логика предикатов.

Применение логики предикатов для записи математических определений, утверждений.

Строгое определение формулы логики предикатов дается по индукции:

1. Все отдельно взятые предикаты, в которых все места замещены предметными переменным или предметными постоянными из соответствующих предметных областей являются формулами. При этом все входящие в предикат предметные переменные считаются свободными.

2. Если F – формула логики предикатов, содержащая свободную переменную x, то $\forall xF$ и $\exists xF$ – также формулы, в которых x связанная переменная, а все остальные переменные те же и того же характера, что и в F.

3. Если F – формула, то и $\neg F$ – формула, все переменные которой те же и того же характера что и в F.

Если F и J формулы, причем нет такой переменной, которая в одну из них входит свободно, а в другую связанно, то FJ, $F \vee J$, $F \to J$,$F \sim J$ – формулы, причем в них входят все переменные из формул F и J и все вхождения те же и того же характера, что и в F и J.

4. Каждая формула получается за конечное число шагов из элементарных формул п.1 при помощи операций п.п. 2 и 3.

Предикаты F и J называются *равными*, если их значения совпадают при всех значениях входящих в них переменных.

Множество истинных формул логики предикатов входит в любую теорию. В исследовании этого множества возникает две проблемы: 1 – получение истинных формул; 2 – проверка формулы на истинность. Прямой перебор всех значений невозможен, т.к. предметные и предикатные переменные имеют в большинстве случаев бесконечные области определения.

Часто используют метод интерпретаций: когда в формулу, требующую доказательства подставляют константы. Подстановка констант позволяет интерпретировать формулу, как осмысленное утверждение об элементах конкретного множества $\boldsymbol{M}$. Этот метод удобен для доказательства выполнимости формул или их неэквивалентности.

Свойства кванторов

$$\forall x\big(A(x)\&\, B(x)\big) = \forall x A(x)\&\, \forall y B(y) \qquad (2.21)$$
$$\exists x\big(A(x) \vee B(x)\big) = \exists x A(x) \vee \exists y B(y)$$

$$\forall x\big(A(x) \vee B\big) = \forall x A(x) \vee B$$
$$\forall x\big(A(x)\&\, B\big) = \forall x A(x)\&\, B \qquad (2.22)$$
$$\exists x\big(A(x) \vee B\big) = \exists x A(x) \vee B$$
$$\exists x\big(A(x)\&\, B\big) = \exists x A(x)\&\, B$$

$$\forall x \forall y A(x,y) = \forall y \forall x A(x,y) \qquad (2.23)$$
$$\exists x \exists y A(x,y) = \exists y \exists x A(x,y)$$

По аналогии с двойственностью конъюнкции и дизъюнкции имеет место двойственность между кванторами

$$\overline{\forall x A(x)} = \exists x \overline{A(x)}$$
$$\overline{\exists x A(x)} = \forall x \overline{A(x)} \qquad (2.24)$$

Эти равносильности и закон двойственности позволяют преобразовать любую формулу логики предикатов в равносильную формулу, в которой символ отрицания стоит только над элементарными предикатами. Получающуюся в результате формулу называют *почти нормальной формой исходной формулы***.**

Формулы, содержащие кванторы как по предметным, так и по предикатным переменным, используют для характеристики какого-либо множества предметных областей с фиксированными индивидуальными предикатами на них. Система таких формул называется системой аксиом, а удовлетворяющие этим аксиомам множества с индивидуальными предикатами – интерпретациями системы аксиом.

1. 7 Лекция №7 (2 часа).

Тема: «Основные подходы к формализации понятия алгоритма. Машина Тьюринга. Принцип Тьюринга - Поста**»**

1.7.1 Вопросы лекции:

1. Основные подходы к формализации понятия алгоритма.
2. Машина Тьюринга. Принцип Тьюринга - Поста.

1.7.2 Краткое содержание вопросов:

1. Основные подходы к формализации понятия алгоритма.
2. Машина Тьюринга. Принцип Тьюринга - Поста.

Формализация понятия алгоритма. Универсальные модели алгоритмов.

Интуитивное понятие алгоритма обладает целым рядом недостатков. Очевидно, что такие понятия, использованные при описании общих свойств алгоритмов, как элементарность шагов, сами нуждаются в уточнении. Очевидно, что их словесные определения будут содержать новые понятия, которые снова потребуют уточнения и т.д. Начиная с 30-х годов, было предложено несколько уточнений понятия алгоритма. Считается, что все они достаточно полно отражают основные черты интуитивного понятия алгоритма. Действительно, все формальные определения алгоритма в некотором смысле эквивалентны друг другу. Поэтому в теории алгоритмов применяется другой подход: выбирается конечный набор исходных объектов, которые объявляются элементарными и конечный набор способов построения их них новых объектов. Этот метод был уже использован в теории множеств и получил название *конструктивного подхода***.**

Алгоритмические модели, которые претендуют на право считаться формализацией понятия «алгоритм», должны быть универсальными, т.е. допускать описание любых алгоритмов.

Можно выделить три основных типа универсальных алгоритмических моделей, различающихся исходными эвристическими соображениями относительно того, что такое алгоритм. 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.

Второй тип модели связан с развитием вычислительной техники и основан на представлении об алгоритме как о некотором детерминированном устройстве, способном выполнять в каждый отдельный дискретный момент времени весьма примитивные операции. Такое представление не оставляет сомнений в однозначности алгоритма и элементарности его шагов. Кроме того, эвристика этой модели близка к ЭВМ и, следовательно, к инженерной интуиции. Основной теоретической моделью этого типа является созданная в 30-х годах концепция машины Тьюринга. Именно машина Тьюринга явилась моделью современной ЭВМ и способствовала развитию современной вычислительной техники.

Наконец, третий тип алгоритмических моделей – это преобразование слов в произвольных алфавитах, в которых элементарными операциями являются подстановки, т.е. замены части слова (подслова) другим словом. Преимущества этого типа моделей заключаются в максимальной абстрактности и возможности применить понятие алгоритма к объектам произвольной, не обязательно числовой природы. Примерами моделей этого типа являются канонические системы Поста и нормальные алгоритмы Маркова. При этом общность формализации в конкретной модели не теряется и доказывается сводимость одних моделей к другим, т.е. показывается, что всякий алгоритм, описанный средствами одной модели, может быть описан средствами другой.

Благодаря взаимной сводимости моделей в общей 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.

МАШИНА ТЬЮРИНГА.

Введение. История вопроса.

В 1935 г. возникло такое положение: свойства, обнаруженные у некоторого точно определенного класса вычислимых теоретико-числовых функций, изучавшихся Чёрчем и Клини в 1932—1935 гг. и названных "λ-определимыми функциями", упорно подсказывали мысль, что этот класс, может быть, охватывает все функции, которые в соответствии с нашим интуитивным представлением можно рассматривать как вычислимые. При этих обстоятельствах Чёрч выдвинул тезис (опубликован в 1936 г.), что все функции, которые интуитивно мы можем рассматривать как вычислимые, или, говоря его словами, как «эффективно вычислимые», являются λ-определимыми, или, эквивалентным образом, общерекурсивными.

Несколько позже, но независимо появилась статья Тьюринга (1936), в которой был введен еще один точно определенный класс интуитивно вычислимых функций, которые мы будем называть «функциями, вычислимыми по Тьюрингу», и относительно этого класса было высказано такое же утверждение; это утверждение мы называем *тезисом Тьюринга.* Вскоре Тьюрингом [1937] было показано, что его вычислимые функции — это то же самое, что λ-определимые функции, и, следовательно, то же самое, что и общерекурсивные функции. Поэтому тезисы Тьюринга и Чёрча эквивалентны. Мы будем обычно ссылаться на оба эти тезиса как на *тезис Чёрча,* а в связи с тем его вариантом, в котором идет речь о «машинах Тьюринга»,— как на *тезис Чёрча — Тьюринга.* В 1936 г. Пост независимо от Тьюринга опубликовал в довольно сжатом изложении формулировку, в основе ту же, что у Тьюринга. В 1943 г., основываясь на своей неопубликованной работе 1920—1922 гг., он опубликовал третий эквивалент аналогичного тезиса. Еще одну эквивалентную формулировку дает теория алгоритмов Маркова [1951г].

Область использования машины Тьюринга

Понятие *машины Тьюринга* возникает в результате прямой попытки разложить интуитивно известные нам вычислительные процедуры на элементарные операции: Тьюринг привел ряд доводов в пользу того, что повторения его элементарных операций было бы достаточно для проведения любого возможного вычисления. Поэтому машина Тьюринга (МТ) используется:

1) если требуется доказать *возможность* алгоритмической реализации вычислительной функции;
2) если требуется *оценить вычислительную сложность* или *трудоемкость* решения задачи по данному алгоритму, т.е. время выполнения алгоритма.

Для этого мы моделируем работу произвольного алгоритма в терминах рассматриваемой задачи. Затем *определяется* класс машин-вычислителей, которые могут решить данную задачу – формально описываются правила работы машины, исходные данные, ограничения и т.д. (поскольку в определении задачи ничего не говорится о программах так таковых в привычном для нас понимании, то алгоритмическая *разрешимость* или *неразрешимость*, сводится к проблеме остановки произвольного алгоритма решения задачи). В качестве машины-вычислителя выберем машину Тьюринга, поскольку ранее было показано, что всякая вычислимая функция реализуема на МТ и сведем решение данной задачи к существующим группам задач, для которых известно, что они решаются на МТ.

Принцип работы машины Тьюринга.

Какая именно команда программы будет выполняться в данный момент, определяется двумя параметрами: читаемым головкой символом и состоянием машины.

Результатами выполнения команды являются: новый символ записанный на ленту в ту ячейку, напротив которой находится в данный момент головка; перемещение головки на одну позицию (ячейку) вправо или влево вдоль ленты; переход машины в новое состояние. В частных случаях новый символ может быть равен старому, перемещение может отсутствовать, состояние может остаться прежним.

Формат команды имеет следующий вид:

a q b r D,

где *a* — читаемый символ; *q* — текущее состояние; *b* — символ записываемый в обозреваемую ячейку ленты вместо символа *a; r* — новое состояние; *D* — направление движения головки машины относительно ленты.

Символы выбираются из конечного алфавита $A = \{a_1, \ldots, a_1\}$.

В дальнейшем будем использовать трехсимвольный алфавит {*e* 0, 1}, причем *e* будет означать «пустой (empty)» символ — отсутствие информации в ячейке, а с помощью нуля и единицы будут кодироваться все данные. Иногда используют двухсимвольный алфавит *A* = {*e,* 1}. В этом случае числа кодируются только единицами: нуль кодируется одной единицей, число один кодируется двумя единицами, а число *x* кодируется *x* + 1 единицами Это — единичная система счисления. Однако она плоха с точки зрения сложности задач (см. гл. 5).

Множество состояний обозначим $Q = \{ q_1, \ldots, q_k\}$. Направление движения *D* выбирается из множества {L, *R, S*} где *L* — движение влево, *R* — движение вправо, *S* — отсутствие движения.

Таким образом, команда 1 q_3 0 q_6 *L* означает: если, находясь в состоянии q_3, машина Тьюринга обозревает ячейку ленты в которой записана 1, то машина должна записать в

эту ячейку 0, произвести сдвиг головки относительно ленты влево на одну ячейку и перейти в состояние q_6.

Это описание действия, соответствующего команде говорит о том, что команда может рассматриваться как отображение пар *(a, q)* в тройки *(b, r, D),* т. е. отображение

$$AxQ => AxQx \{L, R, S\}.$$

Данное отображение является частичным, так как не для любой пары-аргумента существует тройка-результат. Но для произвольной пары существует не более одной тройки, т. е. отображение не является многозначным.

Все действия производятся в дискретном времени. Иначе говоря, можно рассматривать целочисленные моменты времени t = = 0, 1, 2, 3, . . . Любое изменение происходит мгновенно в момент *t = i u* ничего не меняется между двумя соседними моментами времени.

Работает машина Тьюринга следующим образом. Стартовая конфигурация: на ленте находятся исходные данные — строка символов в алфавите *A,* состояние внутренней памяти соответствует некоторому оговоренному (всегда одному и тому же) начальному состоянию, например, q_1. При этом головка машины обозревает некоторую ячейку ленты с записанным там символом *a.* Нормальным считается начальное положение головки напротив самого левого непустого символа, т. е. не совпадающего с *e.*

Момент старта рассматривается как нулевой момент времени. В момент старта выполняется первая команда, это единственная команда, начинающаяся с пары *(a, q_1).* В результате выполнения команды машина перейдет в новое состояние, и головка машины прочтет новый символ с ленты. Эта пара (новый символ, новое состояние) станет начальной частью следующей команды и т. д. Машина будет продолжать работать в дискретном времени, шаг за шагом переходя из состояния в состояние, и постепенно изменяя содержимое ленты. Наконец, для некоторой пары *(a, q)* не окажется команды в программе. Такая ситуация считается завершающей. Машина прекращает функционирование. Оставшаяся запись на ленте считается записью результата.

Таким образом, машина Тьюринга реализует вычисление некоторой функции — отображения исходной строки символов в результирующую строку.

Существует несколько способов представления программы машины Тьюринга (множества команд). Два наиболее употребительных:

1) двумерная таблица (рис. 1.2);

2) диаграмма (нагруженный псевдограф).

В двумерной таблице строки помечаются различными символами алфавита, а столбцы — именами различных состояний машины, т. е. таблица имеет размер *Ik.* Каждой команде программы

Состояние Символ	q_1	…	q	…	q_k
a_1					
…					
			brD		
…					
a1					

Рис. 1.2. Табличная форма программы машины Тьюринга

соответствует единственная клетка в таблице. Она определяется для команды *a q b r D* следующим образом: в клетку, находящуюся на пересечении строки, помеченной сим-

волом я, и столбца помеченного состоянием *q.* вписывается тройка b *r D.*
Для некоторых пар *(a, q)* в программе нет команд, следовательно, соответствующие клетки таблицы остаются пустыми. При достижении в процессе работы пустой клетки машина Тьюринга останавливается.

В качестве простого примера приведем программу вычисления функции S(x) = x + 1, т. е. увеличение аргумента на единицу (рис. 1.3). Используем алфавит A = *{e,* 0, 1}, причем x будем кодировать последовательностью нулей и единиц так. как это принято при двоичном кодировании целых неотрицательных чисел.
предположим также, что в момент старта головка машины Тьюринга находится напротив крайней левой ячейки с символом 1**.**

	q_1	q_2	q_3	q_4
0	$0q_1R$	$1q_3L$	$0q_3L$	
1	$1q_1R$	$0q_2L$	$1q_3L$	
e	eq_2L	$1q_4S$	eq_4R	

Р и с. 1.3. Программа машины Тьюринга для вычисления функции S(x)=x+1

Первая выполняемая команда *—1q₁1q₁R* оставляет 1 в ячейке ленты, оставляет неизменным состояние q_1 и производит сдвиг головки вправо по ленте. В новой читаемой ячейке может оказаться любой из трех символов алфавита. Если это 0 или 1, то производится дальнейшее движение вправо до окончания кода числа *x.* Если же встретится символ *e,* то это будет означать, что код числа закончился и головка находится справа от младшей цифры кода числа. После этого, собственно, и начинается процесс прибавления единицы. Если младшая цифра — 0 то достаточно заменить се на 1 (команда 0 q_2 1 q_3 *L)* и начать обратное движение к исходной позиции. Если младшая цифра — 1 то результатом в данной ячейке будет 0 (сложение по mod 2), и единица перейдет в следующий по старшинству разряд (влево). Процесс распространения переноса может закончиться где-то внутри кода числа и тогда необходимо осуществить «прокрутку» ленты так, чтобы машина остановилась на крайней левой единице кода результата (x+1)

Второй пример - программа вычисления функции *Z(x)* = 0(x) = 0, превращающей запись любого аргумента. x в запись нуля (рис. 1.4). Эта программа стирает с ленты код x, т. е. запол

	q_1	q_2
0	eq_1R	
1	eq_1R	
e	$0q_2R$	

Р 11 с. 1.4. Программа машины Тьюринга для вычисления функции 0(x) = 0

няет клетки символом *e* и перед остановкой записывает в текущую клетку 0.

Более длинная программа получается для вычисления функции I^n_m (x_1,x_2, … , x_n) выбирающей m-й аргумент из последовательности *n* аргументов, $1 \le m \le n$, I^n_m (x_1, x_2, ... ,x_n)=x_m (рис. 1.5).

Представление последовательности *n* аргументов зададим на ленте в виде записанных один за другим через разделитель — пустую клетку *e* — двоичных кодов x_i. Программа, написанная для конкретного значения *m,* действует следующим образом. Сначала стирается (заменяется на *e ...e)* первый аргумент, затем стирается второй, ..., стирается *(m* - 1)-й; затем

подтверждается *m-й* аргумент; затем стираются оставшиеся аргументы.

	q_1	q_2	…	q_{m-1}	q_m	q_{m+1}	…	q_n	q_{n+1}
0	eq_1R	eq_2R	…	$eq_{m-1}R$	$0q_mR$	$eq_{m+1}R$		eq_nR	
1	eq_1R	eq_2R	…	$eq_{m-1}R$	$1q_mR$	$eq_{m+1}R$		eq_nR	
e	eq_2R	eq_3R	…	eq_mR	$eq_{m+1}R$	$eq_{m+2}R$		$eq_{n+1}R$	

Рис. 1.5. Программа машины Тьюринга для вычисления функции

$$I^n_m(x_1, x_2, \dots, x_n)$$

Другой способ представления программы машины Тьюринга — диаграмма (граф).

Диаграмма представляет собой геометрический объект, состоящий из вершин (обозначаемых точками или окружностями) и дуг (рисуемых в виде направленных отрезков прямой со стрелкой на одном из концов или в виде отрезков не самопересекающихся кривых). Каждой вершине приписывается состояние машины Тьюринга: таким образом вершин в диаграмме ровно столько, сколько имеется состояний. Дуге, соединяющей две вершины q_i и q_j, приписывается некоторый символ *a* алфавита *A* и двойка *b D* так, что запись *a* q_i *b* q_j *D* образует команду программы машины Тьюринга.

Дуга (стрелка) символизирует переход из состояния q_i в состояние q_j при условии, что головка читает символ *a.* Одновременно с этим символ *a* заменяется на символ *b* и совершается движение *D.*

Программа вычисления *Z(x)* = O может быть изображена диаграммой, изображенной на рис. 1.7.

Алан Тьюринг сформулировал тезис, связывающий понятие алгоритма и машины: *«Для всякого (неформального) алгоритма может быть построен Тьюрингов алгоритм (программа машины Тьюринга), дающий при одинаковых исходных данных тот же результат».*

Это недоказуемое математическими методами утверждение играет важную роль при проектировании программного обеспечения, особенно на начальных этапах проектирования. Первоначальная постановка задачи зачастую является словесной, неформальной. Если ее решение удается описать в виде конечной последовательности шагов, каждый из которых достаточно прост, то в соответствии с Тезисом Тьюринга это означает, что может быть написана программа на каком-либо алгоритмическом языке, решающая поставленную задачу.

1. 8 Лекция №8 (2 часа).

Тема: «Рекурсивный алгоритм, нормальные алгоритмы Маркова. Понятие эффективности и сложности алгоритмов**»**

1.8.1 Вопросы лекции:

1. Рекурсивные функции.
2. Рекурсивный алгоритм.
3. нормальные алгоритмы Маркова.
4. Понятие эффективности и сложности алгоритмов

1.8.2 Краткое содержание вопросов:

1. Рекурсивные функции.
2. Рекурсивный алгоритм.

Рекурсивные функции.

Всякий алгоритм однозначно ставит в соответствие исходным данным (в случае если определен на них) определенный результат. Поэтому с каждым алгоритмом однозначно связана функция, которую он вычисляет. Исследование этих вопросов привело к созданию

в 30-х годах прошлого 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 вычислимость которых достаточна очевидна. Операции над функциями будем называть *операторами*.

Будем рассматривать только числовые функции, т.е. функции, аргументы и значения которых принадлежат множеству натуральных чисел N (в теории рекурсивных функций полагают N=0, 1, 2, …). Иначе говоря, числовой n-местной функцией $f(x_1, x_2, \ldots, x_n)$ называется функция, определенная на некотором подмножестве $N \subseteq N^n$ с натуральными значениями. Если область определения совпадает с множеством N^n, т.е. $f: N^n \to N$, то говорят, что функция f всюду определенная, в противном случае – частично определенная.

Например: $f(x, y) = x + y$ – всюду определенная двуместная функция.

$f(x, y) = x - y$ – частично определенная функция (она определена при $x \geq y$).

Рекурсивным определением функции принято называть такое определение, при котором значения функции для данных аргументов определяются значениями функции для более простых аргументов (уже вычисленных) или значениями более простых функций.

Простейшим примером рекурсивного определения являются числа Фибоначчи, представляющие собой последовательность чисел $f(n)$, удовлетворяющих условиям

$$f(0) = 1, \quad f(1) = 1, \quad f(n+2) = f(n) + f(n+1),$$

1) $0(x)=0$ – нуль-функция.
2) $S(x)=x+1$ – функция следования.
3) $I_m(x_1, x_2, \ldots, x_n) = x_m$, где $m = 1, \ldots, n$ – проектирующая функция.

Оператор суперпозиции. Суперпозиция является мощным средством получения новых функций из уже имеющихся. Напомним, что суперпозицией называется любая подстановка функций в функции. Оператором суперпозиции P_m^n называется подстановка в функцию от m переменных m функций от n одних и тех же переменных. Например, для функций

$$h(x_1, x_2, \ldots, x_m), g_1(x_1, x_2, \ldots, x_n), \ldots, g_m(x_1, x_2, \ldots, x_n)$$

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

1. Функции $0(x)$, $S(x)$ и $I_m^n(x)$ для всех натуральных n, m, где $m \leq n$, являются примитивно рекурсивными.
2. Если $g_1(x_1, x_2, \ldots, x_n)$, …, $g_m(x_1, x_2, \ldots, x_n)$, $h(x_1, x_2, \ldots, x_n)$ примитивно рекурсивные, то $P_m^n(h, g_1, \ldots, g_m)$ – примитивно-рекурсивные функции для любых натуральных n, m.

3. Если $g_1(x_1,\ldots,x_n)$ и $h(x_1,\ldots,x_n,y,z)$ – примитивно рекурсивные функции, то $R_n(g,h)$ – примитивно-рекурсивная функция.
4. Других примитивно-рекурсивных функций нет.

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

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

Нормальные алгоритмы Маркова.

Третий тип алгоритмических моделей – это преобразование слов в произвольных алфавитах, в которых элементарными операциями являются подстановки, т.е. замены части слова (подслова) другим словом. Преимущества этого типа моделей заключаются в максимальной абстрактности и возможности применить понятие алгоритма к объектам произвольной, не обязательно числовой природы. Примерами моделей этого типа являются канонические системы Поста и нормальные алгоритмы Маркова. При этом общность формализации в конкретной модели не теряется и доказывается сводимость одних моделей к другим, т.е. показывается, что всякий алгоритм, описанный средствами одной модели, может быть описан средствами другой.

Тезисы об «универсальности» алгоритмов: тезис Чёрча, тезис Тьюринга, принцип нормализации Маркова. Эквивалентность различных теорий алгоритмов. Алгоритмические проблемы.

Тьюрингом [1937] было показано, что его вычислимые функции — это то же самое, что λ-определимые функции, и, следовательно, то же самое, что и общерекурсивные функции. Поэтому тезисы Тьюринга и Чёрча эквивалентны. Мы будем обычно ссылаться на оба эти тезиса как на *тезис Чёрча,* а в связи с тем его вариантом, в котором идет речь о «машинах Тьюринга»,— как на *тезис Чёрча — Тьюринга.* В 1936 г. Пост независимо от Тьюринга опубликовал в довольно сжатом изложении формулировку, в основе ту же, что у Тьюринга. В 1943 г., основываясь на своей неопубликованной работе 1920— 1922 гг., он опубликовал третий эквивалент аналогичного тезиса. Еще одну эквивалентную формулировку дает теория алгоритмов Маркова [1951г].

Благодаря взаимной сводимости моделей в общей 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.

Меры сложности алгоритмов. Классы задач P и NP.

Основы анализа алгоритмов

Одну и ту же задачу могут решать много алгоритмов. Эффективность работы каждого из них описывается разнообразными характеристиками.

При анализе алгоритма определяется количество "времени", необходимое для его выполнения. Это не реальное число секунд или других промежутков времени, а приблизительное число операций, выполняемых алгоритмом. Число операций и измеряет относительное время выполнения алгоритма. Таким образом, иногда мы будем называть "време-

нем" вычислительную сложность алгоритма. Фактически количество секунд, требуемое для выполнения алгоритма на компьютере непригодно для анализа, поскольку нас интересует только относительная эффективность алгоритма, решающего конкретную задачу. Действительно алгоритм не становится лучше, если его перенести на более быстрый компьютер, или хуже, если его исполнять на более медленном компьютере.

Во-вторых, фактическое количество операций алгоритма на тех или иных вводимых данных не предоставляет большого интереса и мало его характеризует. Вместо этого важной характеристикой является зависимость числа операций конкретного алгоритма от размера входных данных. Мы можем сравнить два алгоритма по скорости роста числа операций от роста входных данных. Именно скорость роста играет ключевую роль.

При анализе алгоритмов учитывается сложность алгоритмов по времени, однако нужно учитывать и то, сколько памяти нужно тому или иному алгоритму. На ранних этапах развития компьютеров этот анализ носил принципиальный характер. Нередко приходилось выбирать более медленный алгоритм, если он требовал меньше памяти. Разработчики современных программ не ощущают потребность в экономии памяти, в результате чего компьютер морально устаревает задолго до их физической негодности.

Скоростью роста алгоритма называется скорость роста числа операций при возрастании объёма входных данных. Нас интересует только общий характер поведения алгоритма, а не подробности этого поведения. Подводя итоги, при анализе алгоритмов нас будет интересовать скорее класс скорости роста, к которому относится алгоритм, нежели точное количество выполняемых им операций аддитивного и мультипликативного типа.

Некоторые часто встречающиеся классы функций приведены в таблице. В этой таблице приведены значения функций из данного класса на широком диапазоне значений аргумента. Видно, что при небольших размерах входных данных значения функций отличаются незначительно, однако при росте этих размеров разница существенно возрастает. Во-вторых, быстродействующие функции доминируют над функциями с более медленным ростом. Поэтому если мы обнаружим, что сложность алгоритма представляет собой сумму двух или нескольких таких функций, то будем часто отбрасывать все функции кроме тех, которые растут быстрее всего. Если, например, установлено, что алгоритму нужно x^3-30x операций, то будем считать, что сложность алгоритма растёт как x^3. Причина этого в том, что уже при 100 входных данных разница между x^3 и x^3 -30x составляет лишь 0,3%.

Таблица классов роста функций

n	$\log_2 n$	n^2	n^3	2^n	n!
1	0	1	1	2	1
2	1	4	8	4	2
5	2.3	25	125	32	120
10	3.3	100	1000	1024	362880
15	3.9	225	3375	32768	--------
20	4.3	400	8000	1048576	--------
30	4.9	900	27000	1073741824	--------

Скорость роста сложности алгоритма играет важную роль, скорость роста определяется старшим, доминирующим членом формулы. Отбросив все младшие члены, мы получаем то, что называется порядком функции или алгоритма, скоростью роста сложности которого она является. Алгоритмы можно сгруппировать по скорости роста их сложностей. Мы вводим три категории: алгоритмы, сложность которых растёт по крайней мере так же быстро, как данная функция (класс $\Omega(f)$ - читается Омега большое), алгоритмы, сложность которых растёт с той же скоростью(класс $O(f)$ - читается О большое) и алгоритмы, сложность которых растёт медленнее, чем эта функция (класс $\theta(f)$ - читается Тета большое).

Мы занимаемся эффективностью алгоритмов, поэтому класс $\Omega(f)$ не будет представлять для нас большого интереса: например в $\Omega(n^2)$ входят все функции, растущие быстрее, чем n^2.

Класс O(*f*) состоит из функций, растущих не быстрее *f*. Функция *f* образует верхнюю границу для класса O(*f*). Проверить принадлежит ли данная функция классу O(*f*) можно двумя способами:

1. С формальной точки зрения функция *g* принадлежит классу O(*f*), если $g(n) \Leftarrow cf(n)$ для всех n, больших некоторого n_0, и для некоторой положительной константы c.

2. g принадлежит O(*f*), если $\lim(g(n)/f(n)) = c$ (n-> ∞) для некоторой константы c.

По правилу Лопиталя можно заменить предел самих функций пределом их производных.

Через θ (*f*) мы обозначаем класс функций, растущих с той же скоростью, что и *f*. С формальной точки зрения этот класс представляет пересечение двух предыдущих классов $\theta(f) = \Omega(f) \cap O(f)$. При сравнении алгоритмов нас будут интересовать такие, которые решают задачу быстрее, поэтому класс $\theta(f)$ нам не очень интересен.

Алгоритмы полиномиальной сложности (класс P), Алгоритмы недетерминированной полиномиальной сложности (класс NP задач),

1. 9 Лекция № 9 (2 часа).

Тема: «Исчисление высказываний и предикатов. Математические (формальные аксиоматические) теории первого порядка**»**

1.9.1 Вопросы лекции:

1. Формальные системы.
2. Исчисление высказываний.

1.9.2 Краткое содержание вопросов:

1. Формальные системы.
2. Исчисление высказываний.

Исчисление высказываний

Формальная теория или исчисление строится следующим образом:

- Определяется множество формул, или правильно построенных выражений, образующее *язык теории*. Это множество задается конструктивными средствами (как правило, индуктивным определением) и, следовательно, оно перечислимо, обычно оно и разрешимо.
- Выделяется подмножество формул, называемых *аксиомами теории,*
- Задаются правила вывода теории. Правило вывода $R(F_1,...,F_n,G)$ – это вычислимое отношение на множестве формул. Формулы $F_1,\ldots,F_n$ называются *посылками правила R*, а *G* – его *следствием* или заключением.

Выводом формулы *B* из формул $A_1,\ldots,A_n$ называется последовательность формул $F_1,...,F_m$ такая, что $F_m = B$, а любая F_i есть либо аксиома, либо одна из исходных формул $A_1,\ldots,A_n$, либо непосредственно выводима из $F_1,\ldots,F_{i-1}$ по одному из правил вывода. *B* выводима из $A_1,\ldots,A_n$, если существует вывод *B* из $A_1,\ldots,A_n$. Этот факт обозначается $A_1,\ldots,A_n \mapsto B$. $A_1,\ldots,A_n$ называются *гипотезами* или посылками вывода.

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 формулой, доказуемой в теории *T* или теоремой теории *T*. Факт доказуемости формулы *B* обозначается $\mapsto B$.

Очевидно, что присоединение формул к гипотезам не нарушает выводимости. Поэтому, если $\vdash B$ (B – доказуема), то $A \vdash B$ (то есть B – доказуема и с некоторой формулой A). Особое внимание уделяется тождественно-истинным высказываниям, поскольку они должны включаться в любую теорию в качестве общелогических законов. Их порождение и является задачей исчисления высказываний.

Тавтологии

Интерпретацией формулы А алгебры высказываний называется всякий набор истинностных значений атомов, входящих в формулу А. Таблица, содержащая всевозможные интерпретации формулы и соответствующие этим интерпретациям значения формулы, называется истинностной таблицей.

Пусть F_1 и F_2 − две формулы алгебры высказываний, а $A_1, A_2, \ldots A_n$ − набор простых высказываний, входящих, по крайней мере, в одну из формул F_1, F_2. Формулы называются *равносильными*, если при всех значениях истинности $A_1, A_2, \ldots A_n$, значения истинности F_1 и F_2 совпадают. Очевидно, что равносильные формулы имеют одинаковые истинностные таблицы, и, наоборот, если истинностные таблицы формул совпадают, то они равносильны. Отношение равносильности формул является отношением эквивалентности:

1. $A \equiv A$ для любой формулы A.
2. Если $A \equiv B$, то $B \equiv A$ для любых формул A и B.
3. Если $A \equiv B$ и $B \equiv C$, то $A \equiv C$ для любых формул A, B, C.

Поэтому множество всех формул разбивается на классы эквивалентности – классы равносильных формул. Все формулы из одного класса характеризуются одной истинностной таблицей.

В каждой своей интерпретации формула принимает одно из двух истинностных значений: И или Л. Иначе говоря, она задает функцию вида $B^n \to B$. Функция вида $B^n \to B$ называется n–местной истинностной функцией или функцией алгебры высказываний. Две равносильные формулы определяют одну и ту же истинностную функцию.

Исходя из данного набора n атомов, можно составить счетное множество формул. Однако все эти формулы описывают лишь конечное множество истинностных функций.

Число n –местных истинностных функций равно 2^{2^n}.

Для некоторых классов формул применяют специальные названия:

Формула A называется *общезначимой* (тождественно истинной, ***тавтологией***), если во всех своих интерпретациях она принимает значение И.

Формула A называется *невыполнимой* (тождественно ложной, ***противоречием***), если во всех своих интерпретациях она принимает значение Л.

Формула A называется *нейтральной*, если она не является ни общезначимой, ни невыполнимой.

Формула A называется *выполнимой*, если она общезначимая или нейтральная.

Формула A называется *необщезначимой*, если она невыполнимая или нейтральная.

Если $A \to B$ является тавтологией, то говорят, что A логически влечет B, или, что B является логическим следствием A. Если $A \equiv B$ есть тавтология, то говорят, что A и B логически эквивалентны. Истинностные таблицы дают эффективную процедуру для решения вопроса о том, является ли данная формула тавтологией.

Имеют место следующие свойства общезначимых формул:

1. Если E общезначимая формула, содержащая атомы $A_1, \ldots, A_n$, то формула E , получающаяся из E одновременной подстановкой формул $F_1, \ldots, F_n$ вместо атомов $A_1, \ldots, A_n$ соответственно, также общезначимая.

2. Если $\Rightarrow A$ и $\Rightarrow A \to B$, то $\Rightarrow B$.

3. $\Rightarrow E$ тогда и только тогда, когда $\neg E$ – противоречие.

Правила следования

Правила следования являются логической основой содержательных дедуктивных рассуждений. Они позволяют судить о правомерности некоторых следований, исходя из правомерности других следований:

1. Введение импликации – если $\Gamma, A \Rightarrow B$, то $\Gamma \Rightarrow A \to B$. В практике доказательства обычно к тем предложениям, которые уже доказаны (множество Г), добавляется предложение А и, исходя из Г и А, выводится В. После чего говорят "теорема доказана", т.е. доказана $A \to B$. За этим оборотом скрывается неявное прменение данного правила – переход от следования $\Gamma, A \Rightarrow B$ к следованию $\Gamma \Rightarrow A \to B$.
2. Удаление дизъюнкции – если $\Gamma, A \Rightarrow C$ и $\Gamma, B \Rightarrow C$, то $\Gamma, A \vee B \Rightarrow C$.
3. Введение отрицания – если $\Gamma, A \Rightarrow B$ и $\Gamma, A \Rightarrow \neg B$, то $\Gamma \Rightarrow \neg A$. Это правило является основой косвенного доказательства – доказательства методом от противного.
4. Удаление импликации – если $\Gamma \Rightarrow A$ и $\Gamma \Rightarrow A \to B$, то $\Gamma \Rightarrow B$ – modus ponens.
5. Введение конъюнкции – если $\Gamma \Rightarrow A$ и $\Gamma \Rightarrow B$, то $\Gamma \Rightarrow A \wedge B$.
6. Первое удаление конъюнкции – если $\Gamma \Rightarrow A \wedge B$, то $\Gamma \Rightarrow A$.
7. Второе удаление конъюнкции – если $\Gamma \Rightarrow A \wedge B$, то $\Gamma \Rightarrow B$.
8. Первое введение дизъюнкции – если $\Gamma \Rightarrow A$, то $\Gamma \Rightarrow A \vee B$.
9. Второе введение дизъюнкции – если $\Gamma \Rightarrow B$, то $\Gamma \Rightarrow A \vee B$.
10. Удаление двойного отрицания – если $\Gamma \Rightarrow \neg\neg A$, то $\Gamma \Rightarrow A$.
11. Слабое удаление отрицания – если $\Gamma \Rightarrow A$ и $\Gamma \Rightarrow \neg A$, то $\Gamma \Rightarrow B$.
12. Введение эквивалентности – если $\Gamma \Rightarrow A \to B$ и $\Gamma \Rightarrow B \to A$, то $\Gamma \Rightarrow A \equiv B$.
13. Первое удаление эквивалентности – если $\Gamma \Rightarrow A \equiv B$, то $\Gamma \Rightarrow A \to B$.
14. Второе удаление эквивалентности – если $\Gamma \Rightarrow A \equiv B$, то $\Gamma \Rightarrow B \to A$.
15. Удаление отрицания конъюнкции – если $\Gamma \Rightarrow \neg(A \wedge B)$, то $\Gamma \Rightarrow \neg A \vee \neg B$.
16. Удаление отрицания дизъюнкции – если $\Gamma \Rightarrow \neg(A \vee B)$, то $\Gamma \Rightarrow \neg A \wedge \neg B$.
17. Удаление отрицания импликации – если $\Gamma \Rightarrow \neg(A \to B)$, то $\Gamma \Rightarrow A \wedge \neg B$.
18. Удаление отрицания эквивалентности – если $\Gamma \Rightarrow \neg(A \equiv B)$, то $\Gamma \Rightarrow (A \vee B) \wedge \neg(A \wedge B)$.
19. Силлогизм – если $\Gamma \Rightarrow A \to B$ и $\Gamma \Rightarrow B \to C$, то $\Gamma \Rightarrow A \to C$.
20. Контрапозиция – если $\Gamma \Rightarrow A \to B$, то $\Gamma \Rightarrow \neg B \to \neg A$.
21. Modus tollens – если $\Gamma \Rightarrow \neg B$ и $\Gamma \Rightarrow A \to B$,то $\Gamma \Rightarrow \neg A$.
22. Дизъюнктивный силлогизм – если $\Gamma \Rightarrow A \vee B$ и $\Gamma \Rightarrow \neg A$, то $\Gamma \Rightarrow B$.
23. Соединение посылок – если $\Gamma \Rightarrow A \to (B \to C)$, то $\Gamma \Rightarrow A \wedge B \to C$.
24. Разъединение посылок – если $\Gamma \Rightarrow A \wedge B \to C$, то $\Gamma \Rightarrow A \to (B \to C)$.
25. Перемена посылок – если $\Gamma \Rightarrow A \to (B \to C)$, то $\Gamma \Rightarrow B \to (A \to C)$.
26. Конструктивная дилемма – если $\Gamma \Rightarrow A \to B$, $\Gamma \Rightarrow C \to D$, $\Gamma \Rightarrow A \vee C$, то $\Gamma \Rightarrow B \vee D$.
27. Деструктивная дилемма – если $\Gamma \Rightarrow A \to B$, $\Gamma \Rightarrow C \to D, \Gamma \Rightarrow \neg B \vee \neg D$, то $\Gamma \Rightarrow \neg A \vee \neg C$.

Система аксиом для исчисления высказываний

Аксиомы: используют две системы аксиом, одна использует все логические связки

I1. $A \to (B \to A)$

I2. $(A\to B)\to(A\to(B\to C))\to(A\to C)$

I3. $(A\&B)\to A$

I4. $(A\&B)\to B$

I5. $A\to(B\to(A\&B))$

I6. $A\to(A\vee B)$

I7. $B\to(A\vee B)$

I8. $(A\to C)\to((B\to C)\to((A\vee B)\to C))$

I9. $(A\to B)\to((A\to\neg B)\to\neg A)$

I10. $\neg\neg A\to A$

Другая система аксиом использует только две связки $\to$ и $\neg$. При этом сокращается алфавит исчисления и соответственно определение формулы. В результате система аксиом становится компактнее:

II1. $A\to(B\to A)$

II2. $(A\to(B\to C))\to((A\to B)\to(A\to C))$

II3. $(\neg A\to\neg B)\to((\neg A\to B)\to A)$

Эти две системы равносильны в том смысле, что порождают одно и то же множество формул. Это утверждение нуждается в доказательстве, которое заключается в том, что показывается выводимость всех аксиом II из аксиом I, и наоборот, с учетом того, что $\vee$ и & рассматриваются в II не как связки, а как сокращения для некоторых его формул: $A\vee B$ заменяет $\neg A\to B$, $A\&B$ заменяет $\neg(A\to\neg B)$. Возможны и другие системы аксиом, равносильные приведенным.

Правила вывода: правило подстановки Если $\mathfrak{I}$ – выводимая формула, содержащая A, то выводима формула $\mathfrak{I}(B)$, получающаяся из заменой всех вхождений A на произвольную формулу B $\dfrac{\mathfrak{I}(A)}{\mathfrak{I}(B)}$.

Правило заключения Если $\mathfrak{I}$ и $\mathfrak{I}\to\mathfrak{R}$ выводимые формулы, то выводима $\mathfrak{R}$ $\dfrac{\mathfrak{I},\mathfrak{I}\to\mathfrak{R}}{\mathfrak{R}}$.

В этом описании исчисления высказываний аксиомы являются формулами исчисления. Формулы использующиеся в правилах вывода это схемы формул или метаформулы.

Например: 1) Формула A→A выводима из II .

Подставим в II2 A→A вместо B и A вместо C (A→((A→A)→A))→((A→(A→A))→(A→A).

Подставим в II1 A→A вместо B: A→((A→A)→A).

По правилу заключения из шагов 1 и 2 следует (A→(A→A))→(A→A).

Подставим в II1 A вместо B: A→(A→A).

По правилу заключения из шагов 3 и 4 следует A→A.

То есть $\mapsto A \to A$.

2) $A \mapsto B \to A$.

Пусть A - выводима, тогда из A и II1 по правилу заключения получаем $\dfrac{A, A \to (B \to A)}{B \to A}$, что и доказывает искомую выводимость.

Всякую доказанную выводимость вида $\Gamma \mapsto \mathfrak{I}$, где Г - список формул, $\mathfrak{I}$- формула, можно рассматривать как правило вывода $\dfrac{\Gamma}{\mathfrak{I}}$, которое можно присоединить к уже имеющимся.

Полученную выводимость $A \mapsto B \to A$ вместе с правилом подстановки можно рассматривать как правило $\dfrac{\mathfrak{I}}{\mathfrak{R} \to \mathfrak{I}}$, если $\mathfrak{I}$ выводима, то выводима и $\mathfrak{R} \to \mathfrak{I}$, где $\mathfrak{R}$ любая формула.

3) $A \to B, B \to C \mapsto A \to C$. $B \to C \mapsto A(B \to C)$ по новому правилу $\dfrac{\mathfrak{I}}{\mathfrak{R} \to \mathfrak{I}}$.

Из A→(B→C) и II2 по правилу заключения следует (A→B)→(A→C), следовательно $B \to C \mapsto (A \to B) \to (A \to C)$.

Из (A→B) и (A→B)→(A→C) по правилу заключения следует A→C, учитывая 2., получим искомую выводимость.

При переходе от первого шага ко второму неявно использовалось следующее свойство выводимости: если $\Gamma \mapsto \mathfrak{I}$ (Г – список формул), а $\mathfrak{I} \to \mathfrak{R}$, то $\Gamma \to \mathfrak{R}$. Это свойство следует из определения выводимости.

Непротиворечивость теории исчисления высказываний

Выше были введены понятия "доказательство" и "вывод", и на их основе – свойство доказуемости формул и отношение выводимости между формулами. применение этих понятий непосредственно на основе их определений связано со значительными трудностями.

Рассмотрим свойства доказуемости и выводимости.

МТ1. а) $A_1, \ldots, A_n \mapsto A_i, i = \overline{1,n}$. Другими словами, из данного множества посылок выводима каждая из посылок этого множества.

б) Если $A_1, \ldots, A_n \mapsto B_1, \ldots, A_1, \ldots, A_n \mapsto B_k$ и $B_1, \ldots, B_k \mapsto C$, то $A_1, \ldots, A_n \mapsto C$. Другими словами, если из данного множества выводима каждая из формул некоторого другого множества, а из этого другого множества посылок выводима некоторая формула *C*, то она выводима и из первоначального множества посылок.

В алгебре высказываний были введены два важнейших понятия – общезначимости и логического следования. В исчислении высказываний (в форме теории L) подобную роль играют понятия доказуемости и выводимости:

(1) $\Rightarrow A \to B$ означает, что формула $A \to B$ общезначима, т.е., что при всех наборах значений атомов, входящих хотя бы в одну из формул *A* или *B*, формула $A \to B$ принимает только значение И.

(2) $A \Rightarrow B$ означает, что из формулы *A* следует формула *B*, т.е., что при всех наборах значений атомов, входящих хотя бы в одну из формул *A* или *B*, при которых формула *A* имеет значение И, формула *B* также имеет значение И.

(3) $\vdash A \to B$ означает, что формула $A \to B$ доказуема, т.е., что существует конечная последовательность формул, заканчивающаяся формулой $A \to B$, причем каждая формула этой последовательности либо аксиома, либо получена из некоторых двух предшествующих формул последовательности по правилу MP.

(4) $A \vdash B$ означает, что из формулы *A* выводима формула *B*, т.е. существует конечная последовательность формул, заканчивающаяся формулой *B*, причем каждая из формул этой последовательности либо формула *A*, либо аксиома, либо получена из некоторых двух предшествующих формул по правилу MP.

Эти понятия связаны между собой. Так предложением 1а) б) установлена связь между общезначимостью и следованием: $\Rightarrow A \to B$ тогда и только тогда, когда $A \Rightarrow B$.

Следующие утверждения устанавливают связь между доказуемостью и выводимостью, между общезначимостью и доказуемостью.

MT2. Пусть Г – любое множество формул. Тогда: а) если $\Gamma \vdash A \to B$, то $\Gamma, A \vdash B$. В частности, б) если $\vdash A \to B$, то $A \vdash B$.

MT3. Теорема дедукции. Пусть Г – любое множество формул. Тогда: а) если $\Gamma, \mathfrak{I} \vdash \mathfrak{R}$, то $\Gamma \vdash \mathfrak{I} \to \mathfrak{R}$. В частности, б) если $\mathfrak{I} \vdash \mathfrak{R}$, то $\vdash \mathfrak{I} \to \mathfrak{R}$.

Например: В качестве первого применения теоремы дедукции покажем, что II3 выводима из I.

Подставим в I9 $\neg$ A вместо A $(\neg A \to B) \to ((\neg A \to \neg B) \to \neg\neg A)$.

Двойное применение правила заключения дает $\neg A \to B, \neg A \to \neg B \vdash \neg\neg A$.

Из I10 по правилу заключения следует, что $\neg\neg A \vdash A$, то по транзитивности выводимости $\neg A \to B, \neg A \to \neg B \vdash A$.

Переставим гипотезы (из определения выводимости следует, что их порядок не имеет значения) $\neg A \to \neg B, \neg A \to B \vdash A$.

Применяя дважды к шагу 4 теорему дедукции получим II3 $(\neg A \to \neg B) \to ((\neg A \to B) \to A)$.

Распространенным методом математических доказательств является метод доказательства от противного: “Если $\Gamma, A \vdash B$ и $\Gamma, A \vdash \neg B$, то $\Gamma \vdash \neg A$”.

Действительно, по теореме дедукции, если $\Gamma, A \vdash B$ и $\Gamma, A \vdash \neg B$, то $\Gamma \vdash A \to B$ и $\Gamma \vdash A \to \neg B$.Из этих импликаций и аксиомы I9 двойным применением правила заключения получаем $\Gamma \vdash \neg A$.

Обобщая результаты MT2 и MT3 можно утверждать, что $\vdash A \to B$ тогда и только тогда, когда $A \vdash B$. Тем самым установлена связь между доказуемостью и выводимостью.

Формальная аксиоматическая теория называется *непротиворечивой*, если ни для какой формулы *A* формулы *A* и $\neg A$ не являются обе доказуемыми в ней. Формальная аксиоматическая теория называется *противоречивой*, если существует формула *A,* для которой одновременно *A* и $\neg A$ доказуемы в этой теории.

MT4. Если $\vdash E$, то $\Rightarrow E$ для любой формулы *E*.

Следствие. Теория исчисления высказываний непротиворечива.

Полнота теории исчисления высказываний

Логически непротиворечивое исчисление называется *полным* относительно общезначимости, если в нем доказуема всякая общезначимая формула.

МТ6. В исчислении высказываний, если $\Rightarrow E$, то $\vdash E$ для любой формулы *E*.

Следствие. Теория исчисления высказывания полна относительно общезначимости.

Разрешимость теории исчисления высказываний

Известны такие классы вопросов (общие вопросы), что на любой вопрос из данного класса можно найти ответ с помощью единого метода, причем метод применим к любому вопросу рассматриваемого класса и в результате его использования будет непременно получен определенный ответ: "да" или "нет". Метод позволяющий ответить "да" или "нет" на любой частный вопрос общего вопроса, называется *разрешающей процедурой* или *разрешающим алгоритмом* для этого класса вопросов, а проблема отыскания алгоритма называется *проблемой разрешения* для общего вопроса. Не для любого класса вопросов имеются разрешающие алгоритмы. Например, не существует разрешающего алгоритма для класса вопросов: "Имеет ли произвольное диофантово уравнение $F(x_1,\ldots,x_n)=0$, где F – многочлен с целыми коэффициентами, решения в целых числах?" Проблема существования такого алгоритма известна как десятая проблема Гильберта, ее алгоритмическая неразрешимость установлена в 1970 г. советским математиком Ю. В. Матиясевичем.

Проблемы разрешения можно ставить и в формальных аксиоматических теориях:

1. Является ли данное слово алфавита теории формулой?
2. Является ли данная конечная последовательность формул доказательством?
3. Доказуема ли данная формула?

Проблемы разрешения для первых двух классов в любой формальной аксиоматической теории решаются положительно – разрешающие алгоритмы вытекают непосредственно из определений формулы и доказательства. В соответствии с этими алгоритмами можно всегда за конечное число шагов установить, является ли данное слово формулой и является ли данная конечная последовательность доказательством.

Проблема разрешения для третьего класса вопросов если и существует, то не вытекает непосредственно из определения. Хотя именно она представляет наибольший интерес, так как доказуемые формулы логической теории выражают законы логики.

Формальная аксиоматическая теория называется *разрешимой*, если проблема разрешения этой теории решается положительно, т.е. если существует алгоритм, позволяющий за конечное число шагов относительно любой формулы языка теории установить, доказуема ли в данной теории эта формула или нет.

Следствие из МТ4 и МТ6. Теория исчисления высказываний разрешима.

Принципы построения формальных теорий. Аксиоматические системы, формальный вывод.

Формальные системы - это системы операций над объектами, понимаемыми как последовательность символов (т.е. как слова в фиксированных алфавитах), сами операции также являются операциями над символами. Термин "формальный" подчёркивает, что объекты и операции над ними рассматриваются чисто формально, без каких бы то ни было содержательных интерпретаций символов. Предполагается, что между символами не существует никаких связей и отношений, кроме тех, которые явно описаны средствами самой формальной системы.

Исторически теория формальных систем, так же как и теория алгоритмов, возникла в рамках оснований математики при исследовании строения аксиоматических теорий и методов доказательства в таких теориях. Всякая точная теория определяется, во-первых, языком, т.е. некоторым множеством высказываний, имеющих смысл с точки зрения этой теории, и, во-вторых, совокупностью теорем - подмножеством языка, состоящим из высказываний, истинных в данной теории.

В математике с античных времён существовал образец систематического построения теории - геометрия Евклида, в которой все исходные предпосылки сформированы явно, в виде аксиом, а теоремы выводятся из этих аксиом с помощью цепочек логических рассуждений, называемых доказательствами. Однако, до середины 19 века математические теории, как правило, не считали нужным явно выделять все исходные принципы, критерии же строгости доказательств и очевидности утверждений в разные времена были различными и явно не формулировались. Время от времени это приводило к необходимости пересмотра основ той или иной теорий. Известно, например, что основания дифференциального и интегрального счисления, разработанных в 18 век Ньютоном и Лейбницем, в 19 века подверглись серьёзному пересмотру. Математический анализ в его современном виде опирается на работы Коши, Больцано и Вейерштрасса по теории пределов.

В конце 19 века такой пересмотр затронул общие принципы доказательств в математических теориях. Это привело к созданию новой отрасли математики - оснований математики, предметом которой и стало построение теорий, чтобы в них не возникало противоречий. Одной из фундаментальных идей, на которые опираются исследования по основанию математики, является идея формализации теорий, т.е. последовательного проведения аксиоматического метода построения теорий.

При этом не допускается пользоваться какими-либо предположениями об объектах теории, кроме тех, которые выражены явно в виде аксиом; аксиомы рассматриваются как формальные последовательности символов (выражения), а методы доказательств— как методы получения одних выражении из других с помощью операций над символами. Такой подход гарантирует четкость исходных утверждений и однозначность выводов, однако может создаться впечатление, что осмысленность и истинность в формализованной теории не играют никакой роли. Внешне это так, однако, в действительности и аксиомы и правила вывода стремятся выбирать таким образом, чтобы построенной с их помощью формальной теории можно было придать содержательный смысл.

Более конкретно *формальная система* (или *исчисление)* строится следующим образом.

1. Определяется некоторое счетное множество символов, т.е. множество, элементы которого могут быть взаимно однозначно сопоставлены элементам натурального ряда 1,2,...N, которые называется термами. Имеется другое конечное множество символов, элементы которого называются связками или операциями. Наконец, существует конечное множество вспомогательных символов. Конечные последовательности символов называются выражениями данной системы.

2. Определяется *множество формул,* или правильно построенных выражений, образующее язык теории. Это множество задается конструктивными средствами (как правило, индуктивным определением) и, следовательно, перечислимо. Обычно оно и разрешимо. Для правильно построенных формул (ППФ) задаются правила их конструирования, т.е. определяется эффективная процедура, с помощью которой по данному выражению выясняется, является ли формула правильно построенной в данной формальной системе (ФС) или нет. Формула, для которой существует такая процедура, называется разрешимой в данной ФС, в противном случае неразрешимой. Иначе говоря, для неразрешимых формул нельзя построить алгоритм выяснения свойства формулы быть теоремой, для этого требуются все новые и новые озарения (изобретательства), не поддающиеся формализации.

3. Выделяется подмножество формул, называемых *аксиомами* ФС. Так же как и для ППФ для аксиом должна иметься процедура, позволяющая определить, является ли ППФ аксиомой или нет. Подмножество может быть и бесконечным, во всяком случае, оно должно быть разрешимо.

4. Задается конечное множество R_1, $R_{,2,..,}R_k$ отношений между ППФ, называемых правилами вывода. Должна иметься эффективная процедура, позволяющая для произвольной конечной последовательности ППФ решить, может ли каждый член этой последовательности быть выведен с помощью конечного числа правил вывода. Правило вывода

$R(F_1, ..., F_n, G)$ —это вычислимое отношение на множестве формул. Если формулы $F_1, ..., F_n, G$ находятся в отношении R, то формула G называется *непосредственно выводимой* из $F_1, ..., F_n$ по правилу R. Часто правило $R(F_1, ..., F_n, G)$ записывается в виде $(F_1, ..., F_n)/G$. Формулы $F_1, ..., F_n$ называются *посылками* правила R, а G—его следствием или *заключением.* Примеры аксиом и правил вывода будут приведены несколько позднее.

Выводом формулы B из формул $A_1, ..., A_n$ называется последовательность формул $F_1, ..., F_m$, такая, что $F_m = B$, а любая $F_i (i = 1,...,m)$ есть либо аксиома, либо одна из исходных формул $A_1, ..., A_n$, либо непосредственно выводима из формул $F_1, ..., F_{i-1}$ (или какого-то их подмножества) по одному из правил вывода. Если существует вывод B из $A_1, ..., A_n$, то говорят, что B *выводима* из $A_1, ..., A_n$. Этот факт обозначается так: $A_1,...,An \vdash B$. Формулы $A_1, ..., A_n$ называются гипотезами или посылками вывода. Переход в выводе от F_{i-1} к F_i называется i-м *шагом вывода.*

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 формулой, *доказуемой* в теории Т, или *теоремой* теории T; факт доказуемости B обозначается $\vdash B$.

Очевидно, что присоединение формул к гипотезам не нарушает выводимости. Поэтому *если* $\vdash B$, то $A \vdash B$, и если $A_1, ..., A_n \vdash B$, то $A_1, ..., A_n, A_{n+1} \vdash B$ для любых A и A_{n+1}. Порядок гипотез в списке несуществен.

Например, если удалось построить вывод B из $A_1, ..., A_n$, то элементы последовательности ППФ $A_1, ..., A_n$ называются посылками вывода (или гипотезами). Сокращенно вывод B из $A_1, ..., A_n$ записывается в виде $A_1, ..., A_n \vdash B$, или если $\Gamma = \{A_1,.., A_n\}$ то $\Gamma \vdash B$. Напомним, что вывод ППФ без использования посылок есть доказательство ППФ В, а сама В – теорема, и это записывается $\vdash B$.

4.3. Формальные теории. Основные понятия и определения

Исторически понятие формальной теории было разработано в период интенсивных исследований в области оснований математики для формализации собственно логики и теории доказательства. Сейчас этот аппарат широко используется при создании специальных исчислений для решения конкретных прикладных задач.

Выводимость

Пусть $F_1, ..., F_n, G$ - формулы теории Т, то есть $F_1, ..., F_n, G$ являются ППФ. Если существует такое правило вывода R , что $(F_1, ..., F_n, G) \in R$, то говорят, что формула G *непосредственно выводима* из формул $F_1, ..., F_n$ по правилу вывода R. Обычно этот факт записывают следующим образом:

$\frac{F_1,...,F_n}{G} R$, где формулы $F_1, ..., F_n$ называются *посылками*, а формула G – *заключением*.

Замечание. Обозначение правила вывода справа от черты, разделяющей посылки и заключение, часто опускают, если оно ясно из контекста.

Если в теории Т существует вывод формулы G из формул $F_1, ..., F_n$, то это записывают следующим образом:

$F_1, ..., F_n \vdash_T G$, где формулы $F_1, ..., F_n$ называются *гипотезами* вывода. Если теория Т подразумевается, то ее значение обычно опускают.

Если $\vdash_T G$, то формула G называется *теоремой* теории Т (то есть теорема – это формула, выводимая только из аксиом, без гипотез).

Если $\Gamma \vdash_T G$, то $\Gamma, \Delta \vdash_T G$, где Γ и Δ- любые множества формул (то есть при добавлении лишних гипотез выводимость сохраняется).

Правила вывода делятся на прямые и непрямые. Прямые правила вывода – это правила непосредственного перехода от одних формул к другим, т.е. переход от посылки к

заключению. Им сопоставляются определенные шаги формального вывода. Непрямые правила вывода суть правила перехода от одних формальным выводам к другим. Таким правилам соответствуют мета утверждения о преобразованиях одних формальных выводов в другие.

Еще одним интересным способом рассуждения, который может быть оформлен в виде непрямого производного правила, является метод доказательства от противного. Суть его сводится к следующему. Пусть нам надо доказать вывод формулы А из посылок Г. Тогда применяют следующий формальный прием: отрицание формулы А добавляют к множеству формул Г и пытаются получить из посылок ¬А, Г противоречие. Если такое противоречие получено, то это означает, что можно построить вывод А из Г

Синтаксис: Синтаксисом называется набор правил конструирования ППФ.

Семантика: Семантикой называется набор правил интерпретации формул.

Интерпретация: Интерпретацией называется приписывание формуле одного из двух значений истинности: 1 (истинно) или 0 (ложно).

Композиционность семантики заключается в том, что приписываемое значение истинности некоторой формулы зависит от значений истинности составляющих высказываний и структуры формулы.

Общезначимость и непротиворечивость

Формула называется *общезначимой* (или *тавтологией*), если она истинна в любой интерпретации. Формула называется *противоречивой*, если она ложна в любой интерпретации. Выполнимой называется формула, для которой существует хотя бы одна интерпретация, для которой она истинна.

Формула *G* называется *логическим следствием* множества формул Г, если *G* выполняется в любой модели Г.

Фундаментальная проблема логики, называемая проблемой дедукции, состоит в том, чтобы определить, является ли формула G логическим следствием множества формул Г. Само слово дедукция (лат. deductio – выведение) определяется как логическое умозаключение от общих суждений к частным или другим общим суждениям. Если логическим следствием из множества формул Г является формула А, имеющая значение истинности Л (ложь или 0), то говорят, что формула А невыполнима. Именно в этом и состоит принцип дедукции: формула А является логическим следствием множества формул Г тогда и только тогда, когда $\Gamma \cup \{\neg A\}$ невыполнимо.

Полнота, независимость и разрешимость

Пусть множество *M* является моделью формальной теории Т. Формальная теория Т называется *полной* (или *адекватной*), если каждому истинному высказыванию *M* соответствует теорема теории Т.

Если для множества (алгебраической системы) *M* существует формальная полная непротиворечивая теория Т, то *M* называется *аксиоматизируемым* (или *формализуемым*) множеством.

Система аксиом (или аксиоматизация) формально непротиворечивой теории Т называется *независимой*, если никакая из аксиом не выводима из остальных по правилам вывода теории Т.

Еще одна важная характеристика формальной теории – это ее разрешимость. Формальная теория Т называется разрешимой, если существует алгоритм, который для любой формулы языка определяет, является она теоремой в Т или нет.

Например, исчисление высказываний разрешимо, а исчисление предикатов неразрешимо. 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.

Исчисление предикатов неразрешимо. Несмотря на полноту исчисления предикатов, разрешающий алгоритм, связанный с вычислением значений истинности предикатных

формул, построить не удается из-за бесконечности предметной области, которая приводит в общем случае к бесконечным таблицам истинности.

Метатеория формальных систем.

При изучении формальных теорий мы имеем дело с двумя типами высказываний. Во-первых, с высказываниями самой теории (теоремами), которые рассматриваются как чисто формальные объекты, определенные ранее, а во-вторых, с высказываниями о теории (о свойствах ее теорем, доказательств и т.д.), которые формулируются на языке, внешнем по отношению к теории, - метаязыке и называются *метатеоремами.* Различие между теоремами и метатеоремами не всегда будет проводиться явно, но его обязательно надо иметь в виду.

Интерпретацией формальной теории Т в *область интерпретации М* называется функция $I : \mathfrak{I} \to M$, которая каждой формуле формальной теории Т однозначно сопоставляет некоторое содержательное высказывание относительно объектов множества (алгебраической системы) *М*. Это высказывание может быть истинным или ложным (или не иметь истинностного значения). Если соответствующее высказывание является истинным, то говорят, что формула *выполняется* в М.

Интерпретация *I* называется *моделью* множества формул Г, если все формулы этого множества выполняются в интерпретации *I*. Интерпретация *I* называется *моделью* формальной теории Т, если все теоремы этой теории выполняются в интерпретации *I* (то есть все выводимые формулы оказываются истинными в данной интерпретации).

Непротиворечивость. Напомним, что формула называется противоречивой, если она ложна в любой интерпретации. Такое определение противоречивой формулы является семантическим, т.е. связывающим непротиворечивость с истинностью. Исходя из него, можно сформулировать понятие семантически непротиворечивой теории:

Формальная теория Т называется *семантически непротиворечивой*, если ни одна ее теорема не является противоречием. Таким образом, формальная теория пригодна для описания тех множеств (алгебраических систем), которые являются ее моделями. Модель для формальной теории Т существует тогда и только тогда, когда Т семантически непротиворечива.

Формальная теория Т называется *формально непротиворечивой*, если в ней не являются выводимыми одновременно формулы *F* и $\neg F$. Теория Т формально непротиворечива тогда и только тогда, когда она семантически непротиворечива.

С помощью введенных понятий можно сформулировать следующий тезис, что теория Т пригодна для описания тех множеств, которые являются ее моделями. Модель для теории Т существует тогда и только тогда, когда Т семантически непротиворечива. Чисто логические теории – исчисление высказываний и исчисление предикатов пригодны для описания любых множеств, что соответствует общенаучному принципу универсальности законов логики. Лейбниц формулировал его как выполнимость логических законов во всех «мыслимых мирах». Аналогом этого критерия, сформулированным в терминах самих формальных теорий без привлечения семантических понятий, является формальная или дедуктивная непротиворечивость.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Лабораторные работы не предусмотрены рабочим учебным планом

3. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРОВЕДЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

3.1 Практическое занятие №ПЗ-1 (2 часа).

Тема: «Основные операции алгебры высказываний**».**

1. Основные понятия алгебры высказываний.
2. Основные операции алгебры высказываний**».**

3.1.1 Задание для работы:

1. Основные понятия алгебры высказываний.
2. Основные операции алгебры высказываний

3.1.2 Краткое описание проводимого занятия:

1. Основные понятия алгебры высказываний.
2. Основные операции алгебры высказываний

1. Составим таблицу истинности для формулы $\overline{B} \to \overline{A}$:

B	A	$\overline{B}$	$\overline{A}$	$\overline{B} \to \overline{A}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	1
1	1	0	0	1

2. Проверим эквивалентность формул $A \vee B$ и $\overline{A \wedge \overline{B}}$, составив для них таблицы истинности.

A	B	$A \vee B$	$\overline{B}$	$A \wedge \overline{B}$	$\overline{A \wedge \overline{B}}$
0	0	0	1	0	1
0	1	1	0	0	1
1	0	1	1	1	0
1	1	1	0	0	1

Формулы не эквивалентны, так как 3-й и 6-й столбцы таблицы не совпадают.

3. $1 \vee 1 \vee 1$равно-...

ОТВЕТ:1

4. $1 \wedge 0 \vee 1 \to 0$ равно

+а) 0
б) 1
в) -1
г) i
д) e

5.Значение y, при котором выполняется равенство $(1 \to 1) \to y = 0$, равно-…

ОТВЕТ:0

6. $f(x, y) = (1 \to x) \to y$. Тогда $f(1,1)$ равно

+а) 1
б) 0
в) 11
г) 100
д) 10

3.1.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили основных понятия и операции алгебры высказываний;
- приобрели умения и навыки выполнения операций алгебры высказываний.

3.2 Практическое занятие №ПЗ-2 (2 часа).

Тема: «Формулы алгебры высказываний. Основные равносильности. Равносильные преобразования формул**».**

3.2.1 Задание для работы:

1.Формулы алгебры высказываний.

2.Основные равносильности.

3.Равносильные преобразования формул.

3.2.2 Краткое описание проводимого занятия:

1.Формулы алгебры высказываний.

2.Основные равносильности.

3.Равносильные преобразования формул

3.2.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия формулы алгебры высказываний; равносильностей и равносильных преобразований;
- приобрели умения и навыки равносильных преобразований.

3.3 Практическое занятие №ПЗ-3 (2 часа).

Тема: «Булевы функции. Элементарные булевы функции. Представление булевых функций**»**

3.3.1 Задание для работы:

1. Булевы функции. Элементарные булевы функции.
2. Представление булевых функций формулами

3.3.2 Краткое описание проводимого занятия:

1. Булевы функции. Элементарные булевы функции.
2. Представление булевых функций формулами

1. Булевы функции. Элементарные булевы функции.

1.Составим таблицу истинности для формулы $\overline{B} \to \overline{A}$:

B	A	$\overline{B}$	$\overline{A}$	$\overline{B} \to \overline{A}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	1
1	1	0	0	1

2. Проверим эквивалентность формул $A \vee B$ и $\overline{A \wedge \overline{B}}$, составив для них таблицы истинности.

A	B	$A \vee B$	$\overline{B}$	$A \wedge \overline{B}$	$\overline{A \wedge \overline{B}}$
0	0	0	1	0	1
0	1	1	0	0	1
1	0	1	1	1	0
1	1	1	0	0	1

Формулы не эквивалентны, так как 3-й и 6-й столбцы таблицы не совпадают.

2. Представление булевых функций формулами.

1. Задание. Записать ДНФ и КНФ формулы.

Решение. Элементарные дизъюнкции: $x \vee \bar{y}$, z. Элемент. конъюнкции: $x \cdot \bar{y} \cdot z$, x. $f(x,y,z) = xyz \vee \bar{x}y$ – ДНФ ; $f(x,y,z) = (x \vee \bar{y}) \cdot z$ – КНФ.

2. Для упрощения формулы используем правило исключения импликации:

$$A_1 \to A_2 = \bar{A}_1 \vee A_2 .$$

$$\neg(A_1 \to A_2) \vee (A_2 \to \bar{A}_1) = \overline{(\bar{A}_1 \vee A_2)} \vee \bar{A}_2 \vee \bar{A}_1 = (\bar{\bar{A}}_1 \wedge \bar{A}_2) \vee \bar{A}_2 \vee \bar{A}_1 =$$
$$= (A_1 \wedge \bar{A}_2) \vee \bar{A}_2 \vee \bar{A}_1 = \bar{A}_2 \wedge (A_1 \vee 1) \vee \bar{A}_1 = \bar{A}_2 \vee \bar{A}_1 .$$

3. Используя законы логики приведем формулу $\overline{(A \wedge B) \vee C}$ к виду, содержащему только дизъюнкции элементарных конъюнкций. Полученная формула и будет искомой ДНФ: $\overline{(A \wedge B) \vee C} = \overline{(A \wedge B)} \wedge \bar{C} = (\bar{A} \vee \bar{B}) \wedge \bar{C} = (\bar{A} \wedge \bar{C}) \vee (\bar{B} \wedge \bar{C})$

Для построения СДНФ составим таблицу истинности для данной формулы:

A	B	C	$A \wedge B$	$(A \wedge B) \vee C$	$\overline{(A \wedge B) \vee C}$
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	1	0

Помечаем те строки таблицы, в которых формула (последний столбец) принимает значение “1”. Для каждой такой строки выпишем формулу, истинную на наборе переменных *A,B,C* данной строки: строка 1 – $\bar{A} \wedge \bar{B} \wedge \bar{C}$; строка 3 – $\bar{A} \wedge B \wedge \bar{C}$;строка 5 – $A \wedge \bar{B} \wedge \bar{C}$. Дизъюнкция этих трех формул будет принимать значение “1” только на наборах переменных в строках 1, 3, 5, а следовательно и будет искомой совершенной дизъюнктивной нормальной формой (СДНФ):

$$(\bar{A} \wedge \bar{B} \wedge \bar{C}) \vee (\bar{A} \wedge B \wedge \bar{C}) \vee (A \wedge \bar{B} \wedge \bar{C})$$

4. Задание. Найти СДНФ и СКНФ двумя способами.

Решение.

x	y	z	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

1) Получим СДНФ и СКНФ по ТИ:

$$\boldsymbol{f(x,y,z) = \bar{x}\bar{y}\bar{z} \vee \bar{x}\bar{y}z \vee x\bar{y}z \vee xy\bar{z} \vee xyz - СДНФ,}$$

$$\boldsymbol{f(x,y,z) = (x \vee y \vee \bar{z})(x \vee \bar{y} \vee z)(\bar{x} \vee \bar{y} \vee z) - СКНФ}$$

2) Получим СДНФ и СКНФ из ДНФ и КНФ:

$$g(x,y,z) = xy \vee \bar{x}z = xy(z \vee \bar{z}) \vee \bar{x}z(y \vee \bar{y}) = xyz \vee xy\bar{z} \vee \bar{x}yz \vee \bar{x}\bar{y}z$$

$$g(x,y,z) = (\bar{x} \vee y)(x \vee z)(y \vee z) = (\bar{x} \vee y \vee z\bar{z})(x \vee y\bar{y} \vee z)(x\bar{x} \vee y \vee z) =$$

$$(\bar{x} \vee y \vee z)(\bar{x} \vee y \vee \bar{z})(x \vee y \vee z)(x \vee \bar{y} \vee z)(x \vee y \vee z)(\bar{x} \vee y \vee z) =$$

$$(\bar{x} \vee y \vee z)(\bar{x} \vee y \vee \bar{z})(x \vee y \vee z)(x \vee \bar{y} \vee z)$$

5. СДНФ функции f(1, 1, 0)=f(0,1,1)=f(0,0,1)=1 равна

+а) $xy\bar{z} \vee \bar{x}yz \vee \bar{x}\,\bar{y}z$

б) $xyz \vee x\bar{y}z \vee x\bar{y}\,\bar{z} \vee \bar{x}yz \vee \bar{x}\,\bar{y}z$

в) $\bar{x}\,\bar{y}z \vee x\bar{y}\,\bar{z} \vee xy\bar{z}$

г) $xyz \vee \bar{x}yz \vee \bar{x}\,\bar{y}z$

д) $x\bar{x}\,\bar{z} \vee \bar{x}yz \vee \bar{x}\,\bar{y}z$

6. СКНФ функции f(1, 1, 0)=f(0,1,1)=f(0,0,1)=0 равна

+а) $(\bar{x} \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (x \vee y \vee \bar{z})$

б) $(x \vee y \vee z) \vee (x \vee \bar{y} \vee z) \vee (x \vee \bar{y} \vee \bar{z}) \vee (\bar{x} \vee y \vee z)$

в) $\bar{x}\,\bar{y}z \wedge x\bar{y}\,\bar{z} \wedge xy\bar{z}$

г) $(x \vee y \vee z) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee z)$

д) $x\bar{x}\,\bar{z} \vee \bar{x}yz \vee \bar{x}\,\bar{y}z$

3.3.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия о булевых функциях, представлении булевых функций формулами;
- приобрели умения и навыки решать задачи, связанные с булевыми функциями, представлением булевых функций формулами.

3.4 Практическое занятие №ПЗ-4 (2 часа).

Тема: «Алгебра Буля. Модели алгебры Буля»

3.4.1 Задание для работы:

1. Алгебра Буля.
2. Модели алгебры Буля.

3.4.2 Краткое описание проводимого занятия:

1. Формулы алгебры высказываний.
2. Модели алгебры Буля.

1. Упростить $\neg(A_1 \to A_2) \vee (A_2 \to \bar{A}_1)$

Для упрощения формулы используем правило исключения импликации:

$A_1 \to A_2 = \bar{A}_1 \vee A_2$.

$$\neg(A_1 \to A_2) \vee (A_2 \to \bar{A}_1) = \overline{(\bar{A}_1 \vee A_2)} \vee \bar{A}_2 \vee \bar{A}_1 = (\bar{\bar{A}}_1 \wedge \bar{A}_2) \vee \bar{A}_2 \vee \bar{A}_1 =$$
$$= (A_1 \wedge \bar{A}_2) \vee \bar{A}_2 \vee \bar{A}_1 = \bar{A}_2 \wedge (A_1 \vee 1) \vee \bar{A}_1 = \bar{A}_2 \vee \bar{A}_1.$$

2. Используя законы логики приведем формулу $\overline{(A \wedge B) \vee C}$ к виду, содержащему только дизъюнкции элементарных конъюнкций. Полученная формула и будет искомой ДНФ:

$$\overline{(A \wedge B) \vee C} = \overline{(A \wedge B)} \wedge \bar{C} = (\bar{A} \vee \bar{B}) \wedge \bar{C} = (\bar{A} \wedge \bar{C}) \vee (\bar{B} \wedge \bar{C})$$

3. Формула $x \wedge (x \vee \bar{y})$ равносильна

+а) x

б) y

в) $\overline{y}$

г) 1

д) $x \cdot \overline{y}$

4. Формула $x \vee x \cdot \overline{y}$ равносильна

+а) x

б) y

в) $\overline{y}$

г) 1

д) $x \cdot \overline{y}$

5. Формула $x \to 1$ равносильна

+а) 1

б) 0

в) $\overline{x}$

г) x

д) $x \cdot \overline{y}$

6. Формула $x \cdot \overline{y} \to 1$ равносильна

+а) 1

б) 0

в) $\overline{x}$

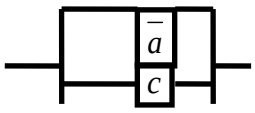
г) $\overline{x} \cdot y$

д) $x \cdot \overline{y}$

7. Составим функцию проводимости для схемы:

$$f(a,b,c) = (\overline{a} \vee c) \vee [(a \vee b) \wedge c] = (\overline{a} \vee c) \vee [ac \vee bc] = \overline{a} \vee c \vee ac \vee bc = \overline{a} \vee c.$$

Полученной формуле соответствует схема:



3.4.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия формулы алгебры высказываний, модели алгебры Буля;

3.5 Практическое занятие №ПЗ-5 (2 часа).

Тема: «Техническая интерпретация алгебры Буля. Булевы функции и математические модели дискретных устройств для переработки информации**»**

3.5.1 Задание для работы:

1. Техническая интерпретация алгебры Буля.
2. Булевы функции и математические модели дискретных устройств для переработки информации.

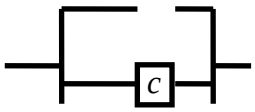
3.5.2 Краткое описание проводимого занятия:

1. Техническая интерпретация алгебры Буля.
2. Булевы функции и математические модели дискретных устройств для переработки информации.

1. Составим функцию проводимости для схемы:

$$f(a,b,c) = (\overline{a} \vee c) \vee [(a \vee b) \wedge c] = (\overline{a} \vee c) \vee [ac \vee bc] = \overline{a} \vee c \vee ac \vee bc = \overline{a} \vee c.$$

Полученной формуле соответствует схема:



2. Раздел дискретной математики, изучающий модели преобразователей дискретной информации, называется теорией
+а) автоматов
б) вероятностей
в) множеств
г) функций
д) поля

3. Конечный автомат это математическая модель дискретного устройства по переработке
+а) информации
б) вероятности
в) алгебры
г) формул
д) поля

3.5.3 Результаты и выводы: в результате проведенного занятия студенты:
- освоили понятия о технической интерпретации алгебры Буля и роли булевых функций при математическом моделировании дискретных устройств для переработки информации.

3.6 Практическое занятие №ПЗ-6 (2 часа).

Тема: «Двойственность. Проблема разрешимости»

3.6.1 Задание для работы:
1. Двойственность.
2. Проблема разрешимости

3.6.2 Краткое описание проводимого занятия:
1. Двойственность.
2. Проблема разрешимости

Принцип двойственности

Пусть $f(x_1, x_2, \ldots, x_n)$ – булева функция. *Двойственной* к ней называется функция $f^*(x_1, x_2, \ldots, x_n) \equiv f(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$. Из определения видно, что двойственность инволютивна: $f^{**}=f$.

Если двойственная функция f^* совпадает с исходной функцией f, то такая функция f называется *самодвойственной*.

$(0)^* \equiv \bar{0} \equiv 1$; $(x)^* = \neg(\bar{x}) \equiv x \Rightarrow$ Функция, тождественно равная своему аргументу, является самодвойственной.

Принцип двойственности для булевых функций

Двойственная к булевой функции может быть получена заменой констант 0 на 1, 1 на 0, дизъюнкции на конъюнкцию, конъюнкции на дизъюнкцию и сохранением структуры формулы (т.е. соответствующего исходному порядка действий).

x	y	$f=x \vee y$	x	y	f^*
0	0	0	1	1	1
0	1	1	1	0	0
1	0	1	0	1	0
1	1	1	0	0	0

$(x\ \bar{y} \vee z)^* \equiv (x \vee \bar{y}) \& z., (x \vee y)^* \equiv x \& y \Rightarrow$ в таблице истинности значения функции и переменных меняются на противоположные

1. В алгебре логики свойство «Если формулы равносильны, то двойственные формулы тоже равносильны» называется законом
+а) двойственности
б) противоречия
в) отрицания
г) де Моргана
д) идемпотентности

2. Функция $f^*(x,y) = \overline{f(\bar{x},\bar{y})}$ по отношению к $f(x,y)$ называется
+а) двойственной
б) булевой
в) дискретной
г) ограниченной
д) целевой

3. Если $f^*(x,y) = f(x,y)$, то функция $f(x,y)$ называется
+а) самодвойственной
б) целочисленной
в) дискретной
г) ограниченной
д) целевой

4. Формула, принимающая значение 1 при каком-то наборе входящих в неё переменных, называется
+а) выполнимой
б) тождественно истинной
в) тождественно ложной
г) опровержимой
д) характеристической

5. Формула, принимающая значение 0 при каком-то наборе входящих в неё переменных, называется
+а) опровержимой
б) тождественно истинной
в) тождественно ложной
г) выполнимой
д) характеристической

6. Формулу, принимающую значение 1 при всех значениях входящих в неё переменных, называют
+а) тавтологией
б) тождественно ложной
в) выполнимой
г) опровержимой
д) логической

7. Тождественно ложная формула называется
+а) противоречивой
б) тавтологией
в) выполнимой
г) опровержимой
д) логической

8. Формула $x \vee y \vee z$ является
+а) выполнимой

б) тавтологией
в) противоречивой
г) нечёткой
д) интегральной

9. Формула $x \vee y \vee z$ является

+а) опровержимой
б) тавтологией
в) противоречивой
г) нечёткой
д) интегральной

3.6.3 Результаты и выводы: в результате проведенного занятия студенты:
- освоили понятия двойственности, проблемы разрешимости;

3.7 Практическое занятие №ПЗ -7 (2 часа).

Тема: «Полиномы Жегалкина. Представление булевых функций полиномами Жегалкина**»**

3.7.1 Задание для работы:

1. Полиномы Жегалкина.
2. Представление булевых функций полиномами Жегалкина.

3.7.2 Краткое описание проводимого занятия:

1. Полиномы Жегалкина.
2. Представление булевых функций полиномами Жегалкина.

1. Полиномы Жегалкина. Представление булевых функций полиномами Жегалкина.

1. Представить полиномом Жегалкина функцию $x_1 \leftrightarrow x_2$

Способ 1. (Метод неопределенных коэффициентов).

Составляем таблицу истинности для функции $x_1 \leftrightarrow x_2$

x_1	x_2	$x_1 \leftrightarrow x_2$
0	0	1
0	1	0
1	0	0
1	1	1

Записываем полином Жегалкина с неизвестными коэффициентами a_0, a_1, a_2, a_{12} для функции от двух переменных: $x_1 \leftrightarrow x_2 = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_{12} x_1 x_2$.

Подставляя в это разложение значения x_1 и x_2 из таблицы, определяем неизвестные коэффициенты:

Подставляя $x_1=0$, $x_2=0$, получаем: $1= a_0$;

$x_1=0$, $x_2=1$ — $0=1 \oplus a_2 \Rightarrow a_2=1$;

$x_1=1$, $x_2=0$ — $0=1 \oplus a_1 \Rightarrow a_1=1$;

$x_1=1$, $x_2=1$— $1=1 \oplus a_{12} \Rightarrow a_{12}=0$.

Полином Жегалкина имеет вид: $x_1 \sim x_2 = 1 \oplus x_1 \oplus x_2$.

Способ 2. (Эквивалентные преобразования).

Сначала запишем СДНФ $\bigvee\limits_{(\sigma_1,\sigma_2)\,|\,f(\sigma_1,\sigma_2)=1} x_1^{\sigma_1} x_2^{\sigma_2}$ эквивалентности:

$$x_1 \leftrightarrow x_2 = \bar{x}_1\bar{x}_2 \vee x_1x_2 = \{\text{т.к. } x \vee y = x \oplus y \oplus xy\} = = \bar{x}_1\bar{x}_2 \oplus x_1x_2 \oplus \bar{x}_1\bar{x}_2x_1x_2 =$$

$$\{\text{поскольку } \bar{x}_1\bar{x}_2x_1x_2 = 0\} = \bar{x}_1\bar{x}_2 \oplus x_1x_2 = \{\text{далее, } \bar{x} = 1 \oplus x, \text{ поэтому}\}$$

$$= (1 \oplus x_1)(1 \oplus x_2) \oplus x_1x_2 = 1 \oplus x_1 \oplus x_2 \oplus x_1x_2 \oplus x_1x_2 = 1 \oplus x_1 \oplus x_2$$

6. Формула $xy \oplus x \oplus y$ равносильна

+а) $x \vee y$

б) y

в) x

г) $x \wedge y$

д) 0

3.7.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия о полиномах Жегалкина и представлении булевых функций полиномами Жегалкина;
- приобрели умения и навыки решать задачи, связанные с полиномами Жегалкина, представлением булевых функций полиномами Жегалкина.

3.8 Практическое занятие №ПЗ-8 (2 часа).

Тема: «Минимизация булевых функций в классе ДНФ»

3.8.1 Задание для работы:

1. Основные понятия.
2. Минимизация булевых функций в классе ДНФ.

3.8.2 Краткое описание проводимого занятия:

1. Основные понятия.
2. Минимизация булевых функций в классе ДНФ.

1) $f(x_1, x_2, x_3) = x_1x_2 \vee \overline{x_1x_2x_3} \vee x_1x_2x_3$. $\overline{x_1x_2x_3}$ – импликанта,

причем простая; $x_1x_2x_3$ – импликанта, но не простая, т.к. удаление x_3 снова дает импликанту x_1x_2 (которая является простой).

2) Найдем импликанты и простые импликанты для функции $f(x_1, x_2) = x_1 \to x_2$. Всего имеется 8 элементарных конъюнкций с переменными x_1, x_2. Приведем их таблицы истинности.

x_1	x_2	$x_1 \to x_2$	$\bar{x}_1\bar{x}_2$	$\bar{x}_1x_2$	$x_1\bar{x}_2$	x_1x_2	$\bar{x}_1$	$\bar{x}_2$	x_1	x_2
0	0	1	1	0	0	0	1	1	0	0
0	1	1	0	1	0	0	1	0	0	1
1	0	0	0	0	1	0	0	1	1	0
1	1	1	0	0	0	1	0	0	1	1

Из таблицы истинности заключаем, что $\bar{x}_1\bar{x}_2$, $\bar{x}_1x_2$, x_1x_2, $\bar{x}_1$, x_2 являются импликантами функции *f*. Из них простыми являются $\bar{x}_1$ и x_2.

3.8.3 Результаты и выводы: В результате проведенного занятия студенты:

- освоили понятия о булевых функциях, минимизации булевых функций;
- приобрели умения и навыки решать задачи, связанные с булевыми функциями, минимизацией булевых функций.

3.9 Практическое занятие №ПЗ-9 (2 часа).

Тема: «Полнота и замкнутость систем булевых функций. Классы Поста**»**

3..1 Задание для работы:

1. Полнота и замкнутость систем булевых функций.
2. Классы Поста.

3.9.2 Краткое описание проводимого занятия:

1. Полнота и замкнутость систем булевых функций.
2. Классы Поста.

Полнота системы логических функций. Базис. При использовании аналитических форм представления логических функции широко используется принцип суперпозиции, заключающийся в замене одних аргументов данной функции другими. Например, если аргументы функции Z = Z(X, Y) являются в свою очередь функциями других аргументов X = X(a, b) и Y = Y(c, d), то можно записать Z = Z(a, b, c, d).

Система S логических функций f_0, f_1, f_2, … , f_k называется *функционально полной*, если любую функцию алгебры логики можно представить в аналитической форме через эти функции. Как известно, любое сложное высказывание можно представить в виде выражения, в которое входят простые высказывания (переменные x_i), операции дизъюнкции, конъюнкции, отрицания и, быть может, скобки (,). Рассмотрим, каким свойствам должны удовлетворять операции, с помощью которых можно выражать любое сложное высказывание.

Система S называется полной в P_k, если любая функция f, $f \in P_k$ представима в виде суперпозиции этой системы, и минимальным базисом, если теряется полнота S при удалении хотя бы одной функции, где P_k – k-значная логика.

В общем случае для установления полноты системы S булевых функций используется критерий полноты Поста-Яблонского.

Дадим предварительно классификацию булевых функций.

Все булевы функции подразделяются на следующие типы:

1. Функция, сохраняющая константу нуль. (Если функция на "0" наборе аргументов равна 0, то она называется сохраняющей константу нуль. $f(0,0,...,0) = 0$).
2. Функция, сохраняющая константу 1. (Если функция на "1" наборе аргументов равна 1, то она называется сохраняющей константу "1". $f(1,1,...,1) = 1$).
3. Самодвойственные функции.

Два набора аргументов называются противоположными, если значения всех аргументов у них противоположны.

Функция называется самодвойственной, если на каждой паре противоположных наборов аргументов она принимает противоположные значения.

4. Монотонная функция

Набор аргументов является возрастающим, если он является старшим, хотя бы в одном из разрядов. Функция называется монотонной, если. она возрастает при, любом возрастании значений аргументов.

Пример: 01 – старший 11 - старший 00 - младший 10 - младший;

1011 – старший, 0011 – младший; 01 и 10- несравнимый набор.

5. Линейная функция

Функция называется линейной, если она может быть представлена многочленом первой степени, т.о.

$f = d \oplus ax \oplus by \oplus cz$;a,b,c,d- могут принимать только два значения 0 и 1. $\oplus$ mod 2,

таблица сложения по модулю 2.

$0 \oplus 0$	0
$0 \oplus 1$	1
$1 \oplus 1$	0
$1 \oplus 0$	1

⊕ - знак сложения по модулю 2. ax≡a/\x.

Определим теперь пять классов булевых функций:

1. Классом P_0 булевых функций $f_i(X_1, X_2, ..., X_n)$ сохраняющих константу 0, называется множество функций вида $\{f_i(X_1, X_2, ..., X_n) / (f_i(0,0,...,0) = 0\}$

2. Классом P_1 булевых функций $f_i(X_1, X_2, ..., X_n)$ сохраняющих константу 1, называется множество функций вида $\{f_i(X_1, X_2, ..., X_n) / f_i(1,1,...,1) = 1\}$.

3. Классом L линейных булевых функций $f_i(X_1, X_2, ..., X_n)$ называется множество функций вида $\left\{f_i(X_1, X_2, ..., X_n) / f_i(X_1, X_2, ..., X_n) = C_0 \oplus \sum C_i X_i\right\}$, где $C_0, C_i = (0,1)$; ⊕ - знак операции «сложение по модулю 2»; 1 ⊕ 0 = 1, 0 ⊕ 1 = 1, 1 ⊕ 1 = 0.

$f_0, f_3, f_5, f_6, f_9, f_{10}, f_{12}, f_{15}$ - линейные функции.

4. Классом S самодвойственных булевых функций $f_i(X_1, X_2, ..., X_n)$ называется множество булевых функций вида $\left\{f_i(X_1, X_2, ..., X_n) / f_i(X_1, X_2, ..., X_n) = \bar{f}(\overline{X}_1, \overline{X}_2, ..., \overline{X}_n)\right\}$.
Функция самодвойственная, если на любой паре противоположных наборов функция принимает противоположные значения.

f_3, f_5, f_{10}, f_{12} - самодвойственные функции

5. Классом M монотонных булевых функций $f_i(X_1, X_2, ..., X_n)$ называется множество булевых функций вида:

$\{f_1(X_1, X_2, ..., X_n) / (\delta_1^*, \delta_2^*, ..., \delta_n^*) \geq (\delta_1, \delta_2, ..., \delta_n) \Leftrightarrow (\delta_i^* \geq \delta_i, \overline{i = 1, n}) \Rightarrow$

$f(\delta_1^*, \delta_2^*, ..., \delta_n^*) \geq f(\delta_1, \delta_2, ..., \delta_n)\}$

1. Проверить самодвойственность функции.

Сначала преобразуем исходную формулу: $(x_1 \to x_2) \to x_1 x_3 = \overline{(\bar{x}_1 \vee x_2)} \vee x_1 x_3 = x_1 \bar{x}_2 \vee x_1 x_3$ $= x_1(\bar{x}_2 \vee x_3)$; $f(x_1, x_2, x_3) = x_1(\bar{x}_2 \vee x_3)$. $\bar{f}(\bar{x}_1, \bar{x}_2, \bar{x}_3) = \overline{\bar{x}_1(x_2 \vee \bar{x}_3)} = x_1 \vee \overline{(x_2 \vee \bar{x}_3)} =$ $x_1 \vee \overline{x_2} x_3$. Пусть $x_1 = 0 \quad x_2 = 0 \quad x_3 = 1$, тогда $f(x_1, x_2, x_3) = 0, \quad \bar{f}(\bar{x}_1, \bar{x}_2, \bar{x}_3) = 1,$ поэтому $f(x_1, x_2, x_3) \neq \overline{f}(\overline{x_1}, \overline{x_2}, \overline{x_3})$, следовательно функция f несамодвойственна.

2. Проверить монотонность.

Функция $f = (1011)$ немонотонная, т.к. $(00) < (01)$, но $f(0,0) > f(0,1)$.

3. Система всех булевых функций обозначается

+а) P_2

б) P_0

в) P_1

г) S

д) L

4. Собственным функционально замкнутым классом Поста из P_2 является класс функций

+а) сохраняющих нуль

б) частично-рекурсивных

в) эффективно вычислимых

г) элементарных булевых

д) проводимости

5. Булевы функции со свойством $f(0,0,...,0)=0$ составляют собственный функционально замкнутый класс Поста из P_2, называемый классом функций

+а) сохраняющих нуль
б) сохраняющих единицу
в) самодвойственных
г) рекурсивных
д) монотонных

6. Собственный функционально замкнутый класс Поста функций из P_2, сохраняющих нуль, обозначается

+а) P_0
б) M
в) P_1
г) S
д) L

7. Булева функция $f(x_1,x_2,...,x_n)=a_1x_1+a_2x_2+...+a_nx_n+a_0$ называется

+а) линейной
б) вычислимой
в) самодвойственной
г) рекурсивной
д) монотонной

8. Булевы функции со свойством $f(x_1,x_2,...,x_n)=a_1x_1+a_2x_2+...+a_nx_n+a_0$ составляют собственный функционально замкнутый класс Поста из P_2, называемый классом функций

+а) линейных
б) булевых
в) элементарных
г) рекурсивных
д) монотонных

9. Собственный функционально замкнутый класс Поста линейных функций из P_2 обозначается

+а) L
б) M
в) P_1
г) S
д) P_0

3.9.3 Результаты и выводы: в результате проведенного занятия студенты:
- освоили понятия полноты и замкнутости систем булевых функции, классов Поста;
приобрели умения и навыки выявлять принадлежность функций классам Поста.

3.10 Практическое занятие № ПЗ-10 (2 часа).

Тема: «Полные системы булевых функций, критерий полноты. К – значные логики**»**

3.10.1 Задание для работы:

1. Полные системы булевых функций.

2. Критерий полноты.

3.10.2 Краткое описание проводимого занятия:

1. Полные системы булевых функций.
2. Критерий полноты.

Теорема «Для того, чтобы система булевых функций $\{f_1, f_2, ..., f_m\}$ была полной, необходимо и достаточно, чтобы она целиком не содержалась ни в одном из пяти замкнутых классов P_0, P_1, S, L, M » называется теоремой

+а) Поста
б) Чёрча
в) Тьюринга
г) Маркова
д) Буля

1. Проверить полноту системы $\{x_1 \leftrightarrow x_2, \overline{x_1}, \overline{x_1} \to \overline{x_2}\}$.

Для доказательства полноты системы $\{x_1 \leftrightarrow x_2, \overline{x_1}, \overline{x_1} \to \overline{x_2}\}$ необходимо проверить, что система содержит функцию не сохраняющую 0, функцию не сохраняющую 1, немонотонную функцию, несамодвойственную функцию и нелинейную функцию. Докажем полноту системы $\sum = \{x_1 \sim x_2, \bar{x}_1, \bar{x}_1 \to \bar{x}_2\}$. Обозначим $f_1(x_1, x_2) = x_1 \sim x_2$ и выпишем ее таблицу истинности

x_1	x_2	$x_1 \sim x_2$
0	0	1
0	1	0
1	0	0
1	1	1

Функция f_1 не сохраняет 0. Выясним, является ли f_1 самодвойственной.

$\bar{x}_1$	$\bar{x}_2$	$\bar{x}_1 \sim \bar{x}_2$	$\bar{f}_1(\bar{x}_1, \bar{x}_2)$
1	1	1	0
1	0	0	1
0	1	0	1
0	0	1	0

Т.к. $f_1(x_1, x_2) \neq \bar{f}_1(\bar{x}_1, \bar{x}_2)$, то f_1 несамодвойственна.

Функция $f_2(x) = \bar{x}$ немонотонная, и не сохраняет 1. Найдем полином Жегалкина для $f_3(x_1, x_2) = \bar{x}_1 \to \bar{x}_2 = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_{12} x_1 x_2$

x_1	x_2	$\bar{x}_1$	$\bar{x}_2$	$\bar{x}_1 \to \bar{x}_2$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	1
1	1	0	0	1

$a_0 = 1; \quad 0 = 1 \oplus a_2 \Rightarrow a_2 = 1; \quad 1 = 1 \oplus a_1 \Rightarrow a_1 = 0; \quad 1 = 1 \oplus 1 \oplus a_{12} \Rightarrow a_{12} = 1;$

Функция $f_3(x_1, x_2) = \bar{x}_1 \to \bar{x}_2 = 1 \oplus x_2 \oplus x_1 x_2$ нелинейная. Согласно теореме о полноте $\sum$ – полная система.

1. Понятие к-значной логики.

2. Функции к-значной логики.

Двузначная логика допускает обобщение на k - значный случай. При этом хотя в k - значных логиках сохраняются многие результаты и свойства двузначной логики, ряд фактов принципиально отличаются от соответствующих результатов алгебры логики. Многие решённые задачи двузначной логики не имеют исчерпывающего решения в k - значных логиках, а иные и вовсе не решены.

Функция $f(x_1, x_2, ..., x_n)$ называется функцией k - значной логики, если её аргументы определены на множестве $\{0,1,2,...,k-1\}$, состоящим из k элементов, а сама функция принимает значения из того же множества.

Множество всех функций k - значной логики обозначается через P_k. Функция $f(x_1, x_2, ..., x_n)$ задана, если задана её таблица истинности. При $k=3$ таблица истинности имеет вид

x_1	x_2	x_3	$f(x_1,x_2,x_3)$	x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	$f(0,0,0)$	1	1	2	$f(1,1,2)$
0	0	1	$f(0,0,1)$	1	2	0	$f(1,2,0)$
0	0	2	$f(0,0,2)$	1	2	1	$f(1,2,1)$
0	1	0	$f(0,1,0)$	1	2	2	$f(1,2,2)$
0	1	1	$f(0,1,1)$	2	0	0	$f(2,0,0)$
0	1	2	$f(0,1,2)$	2	0	1	$f(2,0,1)$
0	2	0	$f(0,2,0)$	2	0	2	$f(2,0,2)$
0	2	1	$f(0,2,1)$	2	1	0	$f(2,1,0)$
0	2	2	$f(0,2,2)$	2	1	1	$f(2,1,1)$
1	0	0	$f(1,0,0)$	2	1	2	$f(2,1,2)$
1	0	1	$f(1,0,1)$	2	2	0	$f(2,2,0)$
1	0	2	$f(1,0,2)$	2	2	1	$f(2,2,1)$
1	1	0	$f(1,1,0)$	2	2	2	$f(2,2,2)$
1	1	1	$f(1,1,1)$				

Так как количество k - значных наборов длины n равно k^n, то число функций от n переменных в k - значной логике равно k^{k^n}. Например, если функций двух переменных в P_2 всего 16, то в P_3 их уже 19683. Таким образом в P_k возрастают трудности по сравнению с P_2 даже с возможностью перебора функций.

Задание. Рассмотреть примеры функций из P_k, которые можно считать элементарными. Построить таблицы истинности при $k=3$.

Задание 1. Функция, называемая отрицанием Поста $\overline{x} = (x+1) \bmod k$. При $k=3$ таблица истинности имеет вид

x	$\bar{x} = (x+1) \bmod 3$
0	1
1	2
2	0

x	$Nx = k-1-x$
0	2
1	1
2	0

Задание 2. Функция, называемая отрицанием Лукасевичаа $\tilde{x} = Nx = k-1-x$. При $k=3$ таблица истинности имеет вид

Задание 3. Функции, называемые а) первым обобщением конъюнкции $\min(x_1, x_2)$;

б) вторым обобщением конъюнкции $(x_1 \cdot x_2) \bmod k$.

При $k=3$ таблицы истинности имеют вид

x_1	x_2	$\min(x_1, x_2)$
0	0	0
0	1	0
0	2	0
1	0	0
1	1	1
1	2	1
2	0	0
2	1	1
2	2	2

x_1	x_2	$(x_1 \cdot x_2) \bmod 3$
0	0	0
0	1	0
0	2	0
1	0	0
1	1	1
1	2	2
2	0	0
2	1	2
2	2	1

Задание 3. Функции, называемые обобщением дизъюнкции

$$\max(x_1, x_2), \qquad (x_1 + x_2) \bmod k.$$

При $k=3$ таблицы истинности имеют вид

x_1	x_2	$\max(x_1, x_2)$
0	0	0
0	1	1
0	2	2
1	0	1
1	1	1
1	2	2
2	0	2
2	1	2
2	2	2

x_1	x_2	$(x_1 + x_2) \bmod 3$
0	0	0
0	1	1
0	2	2
1	0	1
1	1	2
1	2	0
2	0	2
2	1	0
2	2	1

3.10.3 Результаты и выводы: в результате проведенного занятия студенты:

- познакомились с понятием к-значной логики и функции к-значной логики.

3.10.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия о полных системах булевых функций, критерии полноты.;
- приобрели умения и навыки решать задачи, связанные с полными системами булевых функций, критерием полноты.

3.11 Практическое занятие №ПЗ-11 (2 часа).

Тема: «Компьютерные технологии решения задач алгебры высказываний»

3.11.1 Задание для работы:

1 Компьютерные технологии решения задач алгебры высказываний

3.11.2 Краткое описание проводимого занятия:

3.12 Практическое занятие №ПЗ-12 (2 часа).

Тема: «Логика предикатов»

3.12.1 Задание для работы:

1. Предикаты и их свойства. Логические операции над предикатами.
2.Кванторные операции. Логика предикатов.

3.12.2 Краткое описание проводимого занятия:

1. Предикаты и их свойства. Логические операции над предикатами.
2.Кванторные операции. Логика предикатов.

Логика высказываний описывает многие важные логические законы и позволяет решать многие проблемы, однако во многих случаях средства логики высказываний оказываются недостаточными.

1. Предикаты и кванторы

Предикатом $P(x_1,...x_n)$ называется функция $P:M^n \to B$, где M – произвольное множество, а B – двоичное множество $\{0,1\}$. M – называется предметной областью предиката, а $x_1,...x_n$ – предметными переменными. 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 $P(a_1,...a_n)=1$, если и только если $(a_1,...a_n) \in R$

б) всякий предикат $P(x_1,...x_n)$ определяет отношение R, такое, что $(a_1,...a_n) \in R$, если и только если $P(a_1,...a_n)=1$.

При этом R задает область истинности предиката P. Константы 0 и 1 называют нульместными предикатами.

Поскольку предикаты принимают два значения и интерпретируются как высказывания, из них можно образовывать выражения алгебры высказываний, т.е. формулы. Элементарные формулы можно связывать операциями алгебры высказываний $\&, \vee, \to, \neg$, сохраняя за операциями те определения, которые давались в алгебре высказываний.

Кванторы

Кроме операций алгебры высказываний употребляют еще две операции, которые относятся уже не к одной фиксированной ситуации, а ко всему множеству ситуаций.

Пусть *P(x)* – предикат, определенный на *M*. Высказывание "для всех *x* из *M* – *P(x)* истинно" обозначается $\forall xP(x)$. Знак $\forall$ называется квантором общности. Высказывание "существует такой *X* из *M*, что *P* истинно" обозначается $\exists xP(x)$. Знак $\exists$ называется квантором существования.

Переход от *P* к $\forall xP(x)$ или $\exists xP(x)$ называется *связыванием* переменной *x*, или навешиванием квантора на переменную x. Предметную переменную, не связанную никаким квантором, называют *свободной* переменной. Смысл связанных и свободных переменных в предикатных выражениях различен. Свободная переменная – это обычная переменная, которая может принимать значения из *M; P* – переменное высказывание, зависящее от *x*. Выражение $\forall x\, P(x)$ не зависит от переменной *x* и при фиксированных *P* и *M* имеет вполне определенное значение. Это, в частности, означает, что переименование связанной переменной не меняет истинности выражения.

Переменные, являющиеся по существу связанными, встречаются не только в логике. В выражениях $\sum_{x=1}^{10} f(x)$ или $\int_a^b f(x)dx$ переменная $\boldsymbol{x}$ связана, при фиксированной *f* первое выражение становится равно определенному числу, а второе становится функцией *a* и *b*.

Навешивать кванторы можно и на многоместные предикаты и вообще на любые логические выражения, которые при этом заключаются в скобки. Навешивание квантора на многоместный предикат уменьшает в нем число свободных переменных.

Предикаты *F* и *J* называются *равными*, если их значения совпадают при всех значениях входящих в них переменных.

Множество истинных формул логики предикатов входит в любую теорию. В исследовании этого множества возникает две проблемы: 1 – получение истинных формул; 2 – проверка формулы на истинность. Прямой перебор всех значений невозможен, т.к. предметные и предикатные переменные имеют в большинстве случаев бесконечные области определения.

Часто используют метод интерпретаций: когда в формулу, требующую доказательства подставляют константы. Подстановка констант позволяет интерпретировать формулу, как осмысленное утверждение об элементах конкретного множества $\boldsymbol{M}$. Этот метод удобен для доказательства выполнимости формул или их неэквивалентности.

Свойства кванторов

$$\forall x\big(A(x)\&\, B(x)\big) = \forall xA(x)\&\, \forall yB(y)$$
$$\exists x\big(A(x)\vee B(x)\big) = \exists xA(x)\vee \exists yB(y) \qquad (2.21)$$

$$\forall x\big(A(x)\vee B\big) = \forall xA(x)\vee B$$
$$\forall x\big(A(x)\&\, B\big) = \forall xA(x)\&\, B \qquad (2.22)$$
$$\exists x\big(A(x)\vee B\big) = \exists xA(x)\vee B$$
$$\exists x\big(A(x)\&\, B\big) = \exists xA(x)\&\, B$$

$$\forall x\forall yA(x,y) = \forall y\forall xA(x,y)$$
$$\exists x\exists yA(x,y) = \exists y\exists xA(x,y) \qquad (2.23)$$

По аналогии с двойственностью конъюнкции и дизъюнкции имеет место двойственность между кванторами

$$\overline{\forall x A(x)} = \exists x \overline{A(x)}$$
$$\overline{\exists x A(x)} = \forall x \overline{A(x)} \qquad (2.24)$$

Эти равносильности и закон двойственности позволяют преобразовать любую формулу логики предикатов в равносильную формулу, в которой символ отрицания стоит только над элементарными предикатами. Получающуюся в результате формулу называют *почти нормальной формой исходной формулы***.**

1. На множестве $D = \{1,2,3,4,5,6,7\}$ задан предикат $P(x): x -$простое число. Тогда мощность области истинности предиката равна-…
ОТВЕТ:4

2. На множестве $D = \{1,2,3,4,5,6,7\}$ задан предикат $P(x): x -$простое число. Тогда $P(1)$ равно-…
ОТВЕТ:0

3. На множестве $D = \{1,2,3,4,5,6,7\}$ задан предикат $P(x): x -$простое число. Тогда $P(2)$ равно-…
ОТВЕТ:1

4. На множестве $D = \{1,2,3,4,5,6,7\}$ задан предикат $P(x): x -$простое число. Тогда $P(4)$ равно-…
ОТВЕТ:0

5. На множестве $D = \{1,2,3,4,5,6,7\}$ задан предикат $P(x): x -$простое число; $q = \forall_{x \in D} P(x) -$ высказывание. Тогда q равно-…
ОТВЕТ:0

6. На множестве $D = \{1,2,3,4,5,6,7\}$ задан предикат $P(x): x -$простое число; $p = \exists_{x \in D} P(x) -$ высказывание. Тогда p равно-…
ОТВЕТ:1

7. Задан предикат $P(x):$ множество x таких, что $\sqrt{3x-2} \geq x$. Тогда область истинности предиката I_p равна

+а) [1; 2]
б) (1; 2)
в) [1; 2)
г) (1;2]
д) $[2;+\infty)$

3.12.3 Результаты и выводы: в результате проведенного занятия студенты:
- освоили понятия предиката, логические операции над предикатами, кванторные операции, элементы логики предикатов;
- приобрели умения и навыки решать задачи с элементами логики предикатов.

3.13 Практическое занятие №ПЗ-13 (2 часа).

Тема: «Основные подходы к формализации понятия алгоритма. Машина Тьюринга**»**

3.131 Задание для работы:

1. Основные подходы к формализации понятия алгоритма.
2. Машина Тьюринга. Принцип Тьюринга - Поста

3.13.2 Краткое описание проводимого занятия:

1. Основные подходы к формализации понятия алгоритма.
2. Машина Тьюринга. Принцип Тьюринга - Поста.

1.

	a_0	1
q_1	$1Hq_0$	$1Пq_1$

Из любой начальной конфигурации(УУ обозревает не пустой символ) эта машина Тьюринга переводит слово 11 в слово-...(Отв.: 111)

2. В команде $a_3q_2 \rightarrow a_0Лq_0$ следующее состояние машины Тьюринга

+а) q_0

б) q_2

в) q_1

г) a_0

д) q_2

3. Одной из моделей (формализаций) алгоритма является

+а) машина Тьюринга

б) задача линейного программирования

в) эйлеровы графы

г) алгебра множеств

д) алгебра логики

4.По команде $a_3q_2 \rightarrow a_0Лq_0$ состояние машины меняется

+а) с q_2 на q_0

б) с q_0 на q_2

в) с a_3 на q_2

г) с a_3 на a_0

д) с a_0 на a_3

5.По команде $a_3q_2 \rightarrow a_0Лq_0$ машина меняет в ячейке символ внешнего алфавита

+а) с a_3 на a_0

б) с a_0 на a_3

в) с a_0 на q_2

г) с q_2 на q_1

д) с q_1 на q_2

6.Состояние машины перед исполнением команды $a_3q_2 \rightarrow a_0Лq_0$ это

+а) q_2

б) q_1

в) q_0

г) a_1

д) a_2

7.В конфигурации $\begin{matrix} a_0 & 3 & 1 & 5 & a_0 \\ & & & q_1 & \end{matrix}$ обозревается символ-...

ОТВЕТ:5

3.13.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия об основных подходах к формализации понятия алгоритма, понятие машины Тьюринга;
- приобрели умения и навыки алгоритмизации простейших задач.

3.14 Практическое занятие №ПЗ-14 (2 часа).

Тема: «Рекурсивные функции (Рекурсивный алгоритм)»

3.14.1 Задание для работы:

1. Рекурсивные функции.
2. Рекурсивный алгоритм

3.14.2 Краткое описание проводимого занятия:

1. Рекурсивные функции.
2. Рекурсивный алгоритм

1. Пусть заданы число a и функция $\psi(x,y)$. Функцию $f(y)$, определённую системой равенств $\begin{cases} f(0)=a \\ f(y+1)=\psi\left(y,f(y)\right) \end{cases}$, называют полученной по схеме …

(Отв. примитивной рекурсии)

2. 10. Пусть заданы число $a=2$ и функция $\psi(x,y)=y+3$. Функция $f(y)$ получена по схеме примитивной рекурсии: $\begin{cases} f(0)=a \\ f(y+1)=\psi\left(y,f(y)\right) \end{cases}$. Тогда $f(1)$ равно-…

ОТВЕТ:5

3. Одна из моделей (формализаций) алгоритма это
+а) рекурсивный алгоритм
б) логика предикатов
в) алгебра множеств
г) алгоритм Краскала
д) булевы функции

4.Теория рекурсивных функций это модель (формализация)
+а) алгоритма
б) алгебры логики
в) алгебры множеств
г) теории групп
д) линейной алгебры

5.Гипотеза «Числовая функция тогда и только тогда алгоритмически вычислима, когда она частично рекурсивна» называется тезисом(принципом)
+а) Чёрча
б) Маркова
в) Тьюринга
г) Миля
д) Мура

6.Исходная простейшая функция $\lambda(x)=x+1$ в классе рекурсивных функций называется оператором
+а) следования
б) аннулирования
в) Чёрча
г) проектирования
д) суперпозиции

7.Исходная простейшая функция $O(x)=0$ в классе рекурсивных функций называется оператором

+а) аннулирования
б) следования
в) Лапласа
г) проектирования
д) суперпозиции

8. Пусть заданы число $a=2$ и функция $\psi(x,y)=y+3$. Функция $f(y)$ получена по схеме примитивной рекурсии: $\begin{cases} f(0)=a \\ f(y+1)=\psi\left(y,f(y)\right) \end{cases}$. Тогда $f(3)$ равно

+а) 11
б) 8
в) 5
г) 3
д) 7

9. $a=1$, $\psi(x,y)=2y$, $y=0,1,2,3,\ldots$, функция $f(y)$ получена по схеме примитивной рекурсии $\begin{cases} f(0)=a \\ f(y+1)=\psi\left(y,f(y)\right) \end{cases}$. Алгоритм вычисляет функцию $f(y)=$

+а) 2^y
б) $(-3)^y$
в) -2y
г) 1-2y
д) $(-2)^y$

10. $a=1$, $\psi(x,y)=(x+1)y$, $y=0,1,2,3,\ldots$, функция $f(y)$ получена по схеме примитивной рекурсии $\begin{cases} f(0)=a \\ f(y+1)=\psi\left(y,f(y)\right) \end{cases}$. Алгоритм вычисляет функцию $f(y)=$

+а) y!
б) (x + 1)y
в) $(x+1)^y$
г) y^{x+1}
д) 2^y

11. Алгоритм $f(0,x)=x$, $f(y+1,x)=f(y,x)+1$, $y=0,1,2,3,\ldots$, вычисляет функцию $f(y,x)=$

+а) y + x
б) y+1
в) y-1
г) 0
д) y

3.14.3 Результаты и выводы: в результате проведенного занятия студенты:
- освоили понятие о рекурсивном алгоритме;
- приобрели умения и навыки алгоритмизации простейших задач.

3.1516 Практическое занятие № ПЗ-15-16 (2 часа).

Тема: «Нормальные алгоритмы Маркова». «Понятие эффективности и сложности алгоритмов»

3.15-16.1 Задание для работы:

Нормальные алгоритмы Маркова. Понятие эффективности и сложности алгоритмов.
1. Нормальные алгоритмы Маркова.
2. Понятие эффективности и сложности алгоритмов.

3.15-16.2 Краткое описание проводимого занятия:
1. Нормальные алгоритмы Маркова.
2. Понятие эффективности и сложности алгоритмов.

Третий тип алгоритмических моделей – это преобразование слов в произвольных алфавитах, в которых элементарными операциями являются подстановки, т.е. замены части слова (подслова) другим словом. Преимущества этого типа моделей заключаются в максимальной абстрактности и возможности применить понятие алгоритма к объектам произвольной, не обязательно числовой природы. Примерами моделей этого типа являются канонические системы Поста и нормальные алгоритмы Маркова. При этом общность формализации в конкретной модели не теряется и доказывается сводимость одних моделей к другим, т.е. показывается, что всякий алгоритм, описанный средствами одной модели, может быть описан средствами другой.

Тезисы об «универсальности» алгоритмов: тезис Чёрча, тезис Тьюринга, принцип нормализации Маркова. Эквивалентность различных теорий алгоритмов. Алгоритмические проблемы.

Тьюрингом [1937] было показано, что его вычислимые функции — это то же самое, что λ-определимые функции, и, следовательно, то же самое, что и общерекурсивные функции. Поэтому тезисы Тьюринга и Чёрча эквивалентны. Мы будем обычно ссылаться на оба эти тезиса как на *тезис Чёрча,* а в связи с тем его вариантом, в котором идет речь о «машинах Тьюринга»,— как на *тезис Чёрча — Тьюринга.* В 1936 г. Пост независимо от Тьюринга опубликовал в довольно сжатом изложении формулировку, в основе ту же, что у Тьюринга. В 1943 г., основываясь на своей неопубликованной работе 1920— 1922 гг., он опубликовал третий эквивалент аналогичного тезиса. Еще одну эквивалентную формулировку дает теория алгоритмов Маркова [1951г].

Благодаря взаимной сводимости моделей в общей 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.

1. Одна из моделей(формализаций) алгоритма это
+а) нормальный алгоритм
б) логика предикатов
в) линейная алгебра
г) алгебра множеств
д) булевы функции
2. Гипотеза «Для нахождения значений функции, заданной в некотором алфавите, тогда и только тогда существует некоторый алгоритм, когда функция нормально вычислима» называется тезисом (принципом)
+а) Маркова
б) Чёрча
в) Тьюринга
г) Краскала
д) Мура

3. Задан нормальный алгоритм Маркова: алфавит $A=\{1,+\}$ и схема подстановок 1) $1+\to+1$, 2) $+1\to 1$, 3) $1\to\bullet\ 1$. Алгоритм перерабатывает слово 1111+11+111 в слово
+а) 111111111
б) 111111
в) 11111
г) Λ
д) 111

4. Задан нормальный алгоритм Маркова: алфавит $A=\{1,+\}$ и схема подстановок 1) $+\to\Lambda$, 2) $1\to\bullet\ 1$. Алгоритм перерабатывает слово 11+111+1+11 в слово
+а) 11111111
б) Λ
в) 11111
г) 111111
д) 111

5. Задан нормальный алгоритм Маркова: алфавит $A=\{1\}$ и схема подстановок 1) $\Lambda\to\bullet\ 1$. Алгоритм перерабатывает слово 11 в слово
+а) 111
б) Λ
в) 1
г) 1+1
д) +

6. Задан нормальный алгоритм Маркова: алфавит $A=\{1\}$ и схема подстановок 1) $\Lambda\to\bullet\ 1$. Алгоритм перерабатывает слово 111 в слово
+а) 1111
б) Λ
в) 11
г) 11+1
д) +1

7. Задан нормальный алгоритм Маркова: алфавит $A=\{1\}$ и схема подстановок 1) $\Lambda\to\bullet\ 1$. Алгоритм вычисляет функцию $f(x)=$
+а) x+1
б) 0
в) 1
г) x
д) 2^x

8. Марковская подстановка (P,Q) обозначается через
+а) $P\to Q$
б) $f(P)=Q$
в) $P=Q$
г) $P\Lambda Q$
д) $P\div Q$

9. Символом $P\to\bullet\ Q$ обозначается марковская подстановка
+а) заключительная
б) начальная
в) вторая в списке
г) третья в списке

д) предпоследняя

10. Нормальный алгоритм Маркова с алфавитом $A=\{a,b\}$ и схемой подстановок 1) $bb\to ba$, 2) $ba\to a$, 3) $a\to\Lambda$, 4) $b\to_{\bullet}\Lambda$ преобразует слово aba в слово

+а) Λ

б) ba

в) aa

г) b

д) $abab$

2. Понятие эффективности и сложности алгоритмов.

Алгоритмы полиномиальной сложности (класс P).

Большинство алгоритмов имеют полиномиальный порядок сложности. Иногда время работы оказывается линейным, как при последовательном поиске: при удлинении списка данных вдвое алгоритм работает вдвое дольше. В алгоритмах последовательного поиска нас интересует процесс просмотра списка в поисках некоторого элемента, называемого целевым. При последовательном поиске предполагается, что список не отсортирован. Например, ключевое значение может быть номером сотрудника, фамилией, или любым другим уникальным идентификатором. Алгоритм последовательного поиска последовательно просматривает по одному элементу списка, начиная с первого, до тех пор пока не найдет нужный элемент. Очевидно, что чем дальше в списке находится конкретное значение ключа, тем больше времени уйдет на его поиск.

Очень часто встречаются алгоритмы сложности $O(N^2)$ – такую сложность имеют некоторые алгоритмы сортировки: если длину входного списка удвоить, то время работы алгоритма возрастет в 4 раза. Все восемь существующих алгоритмов сортировки демонстрируют широкий спектр возможных вариантов поведения. Первая из них, сортировка вставками, сортирует список, вставляя очередной элемент в нужное место уже отсортированного списка. Пузырьковая сортировка сравнивает элементы попарно, переставляя между собой элементы тех пар, порядок в которых нарушен. Сортировка Шелла представляет собой многопроходную сортировку, при которой список разбивается на подсписки, каждый из которых сортируется отдельно, причем на каждом проходе число подсписков уменьшается, а их длина растет.

Рассмотрим наиболее типичный вариант сортировки – пузырьковую сортировку. Алгоритм пузырьковой сортировки совершает несколько проходов по списку. При каждом проходе происходит сравнение соседних элементов. Если порядок соседних элементов неправильный, они меняются местами. Каждый проход начинается с начала списка. Сперва сравниваются 1 и 2 элементы, затем 2 и 3, потом 3 и 4 и т.д. Элементы с неправильным порядком в паре переставляются. При обнаружении на первом проходе наибольшего элемента списка он будет переставляться со всеми последующими пока не дойдет до конца списка. Поэтому при втором проходе нет необходимости производить сравнение с последним элементом. При втором проходе второй по величине элемент списка опустится во вторую позицию с конца и т.д. Стоит заметить, что при каждом проходе ближе к своему месту продвигается сразу несколько элементов, хотя гарантировано занимает окончательное положение лишь один.

Сколько сравнений выполняется в наихудшем случае. На первом проходе будет выполнено $N-1$ сравнений соседних значений, на втором $N-2$ сравнений. Дальнейшее исследование показывает, что при каждом очередном проходе число сравнений уменьшается на 1. Поэтому сложность в наихудшем случае дается формулой

$$W(N)=\sum_{i=N-1}^{1} i=\sum_{i=1}^{N-1} i=\frac{(N-1)N}{2}=\frac{N^2-N}{2}\approx\frac{1}{2}N^2=O(N^2)$$

Сложность стандартного алгоритма матричного умножения равна $O(N^3)$ и при увеличении размеров матриц вдвое такой алгоритм работает в 8 раз дольше.

Матрица – математический объект, эквивалентный двумерному массиву. Если число столбцов в первой матрице совпадает с числом строк во второй, то эти две матрицы можно перемножить:

Для вычисления произведения двух матриц каждая строка первой почленно умножается на каждый столбец второй. Затем подсчитывается сумма таких произведений и записывается в соответствующую клетку результата. Стандартный алгоритм умножения матрицы размером $a \times b$ на матрицу размером $b \times c$ выполняет abc умножений и $a(b-1)c$ сложений. Однако исследователям удалось обнаружить другие алгоритмы, умножающие матрицы более эффективно, в частности алгоритм Виноградова и алгоритм Штрассена. Алгоритм Штрассена работает с квадратными матрицами. На самом деле он настолько эффективен, что иногда разумно расширить матрицы до квадратных, и при этом он все равно дает выигрыш. Анализ общего случая показывает, что число умножений при перемножении двух $N \times N$ матриц приблизительно равно $N^{2,81}$, а число сложений $6N^{2,81} - 6N^2$.

Сводя три результата воедино, имеем следующую таблицу:

	Умножение	Сложение
Стандартный алгоритм	N^3	$N^3 - N^2$
Алгоритм Виноградова	$\frac{N^3 + 2N^2}{2}$	$\frac{3N^3 + 4N^2 - 4N}{2}$
Алгоритм Штрассена	$N^{2,81}$	$6N^{2,81} - 6N^2$

Все рассмотренные алгоритмы имеют полиномиальную сложность. Самым времяемким был алгоритм умножения матриц, его сложность $O(N^3)$. Главное, однако то, что мы могли найти такое решение задач за разумный промежуток времени. Все эти задачи относятся к классу P – классу задач полиномиальной сложности. Такие задачи называются также практически разрешимыми.

Алгоритмы недетерминированной полиномиальной сложности (класс NP задач).

Кроме практически разрешимых задач, относящихся к классу p – классу задач полиномиальной сложности, существует и другой класс задач: они практически неразрушимы и мы не знаем алгоритмов, способных решить их за разумное время. Эти задачи образуют класс NP – недетерминированной полиномиальной сложности.

Отметим только, что сложность всех известных детерминированных алгоритмов, решающих эти задачи, либо экспоненциально, либо факториальна. Сложность некоторых из них равна 2^N, где N – количество входных данных. В этом случае при добавлении к списку входных данных одного элемента время работы алгоритма удваивается. Если для решения такой задачи на входе из 10 элементов алгоритму требовалось 1024 операций, то на входе из 11 элементов число операций составит уже 2048. Это значительное возрастание времени при небольшом удлинении входа.

Термин «недетерминированные полиномиальные» характеризующие задачи из класса NP, объясняется следующим двухшаговым подходом к их решению. На первом шаге имеется недетерминированный алгоритм, генерирующий возможное решение такой задачи – что-то вроде попытки указать решение; иногда такая попытка оказывается успешной, и мы получаем оптимальный или близкий к оптимальному ответу, чаще нет (ответ далек от оптимального). На втором шаге проверяется, действительно ли ответ, полученный на 1 шаге, является решением исходной задачи. Каждый из этих шагов по отдельности требует

полиномиального времени. Проблема, однако, в том, что мы не знаем, сколько раз нам придется повторить оба эти шага, чтобы получить искомое решение. Хотя оба шага и полиномиальны, число обращений к ним может быть экспоненциальным или факториальным.

К классу NP относится задача о коммивояжере. Нам задан набор городов и «стоимость» путешествия между любыми двумя из них. Нужно определить такой порядок, в котором следует посетить все города (по одному разу) и вернуться в исходный город, чтобы общая стоимость путешествия оказалась минимальной. Эту задачу можно применить, например, для определения порядка эффективного сбора мусора из баков на улицах города или выбора кратчайшего пути распространения информации по всем узлам компьютерной сети. Восемь городов можно упорядочить 40 320 возможными способами, а для десяти городов это число возрастает уже до 3 628 800. Поиск кратчайшего пути требует перебора всех этих возможностей. Предположим, что у нас есть алгоритм, способный подсчитать стоимость путешествия через 15 городов в указанном порядке. Если за секунду такой алгоритм способен пропустить через себя 100 вариантов, то ему потребуется больше четырех веков, чтобы исследовать все возможности и найти кратчайший путь. Даже если в нашем распоряжении имеется 400 компьютеров, все равно у них уйдет на это год, а ведь мы имеем дело лишь с 15 городами. Для 20 городов миллиард компьютеров должен будет работать параллельно в течение девяти месяцев, чтобы найти кратчайший путь. Ясно, что быстрее и дешевле путешествовать хоть как-нибудь, чем ждать, пока компьютеры выдадут оптимальное решение.

Можно ли найти кратчайший путь, не просматривая их все? До сих пор никому не удалось придумать алгоритм, который не занимается, по существу, просмотром всех путей. Когда число городов невелико, задача решается быстро, однако это не означает, что так будет всегда, а нас как раз интересует решение общей задачи.

Задача о коммивояжере, конечно, очень похожа на задачи про графы. Каждый город можно представить вершиной графа, наличие пути между двумя городами - ребром, стоимость путешествия между ними — весом этого ребра. Отсюда можно сделать вывод, что алгоритм поиска кратчайшего пути решает и задачу коммивояжера, однако это не так. Какие два условия задачи о коммивояжере отличают ее от задачи о кратчайшем пути? Во-первых, мы должны посетить все города, а алгоритм поиска кратчайшего пути дает лишь путь между двумя заданными городами. Если выбрать путь из кратчайших кусков, выдаваемых алгоритмом поиска кратчайших путей, то он будет проходить через некоторые города по нескольку раз. Второе отличие состоит в требовании возвращения в исходную точку, которое отсутствует в поиске кратчайшего пути.

Наше краткое обсуждение того, насколько велико число возможных упорядочиваний вершин, должно было убедить Вас в том, что детерминированный алгоритм, сравнивающий все возможные способы упорядочивания. работает чересчур долго. Чтобы показать, что эта задача относится к классу NP, нам необходимо понять, как ее можно решить посредством описанной выше двухшаговой процедуры. В задаче о коммивояжере на первом шаге случайным образом генерируется некоторое упорядочивание городов. Поскольку это недетерминированный процесс, каждый раз будет получаться новый порядок. Очевидно, что процесс генерации можно реализовать за полиномиальное время: мы можем хранить список городов, генерировать случайный номер, выбирать из списка город с этим именем и удалять его из списка, чтобы он не появился второй раз. Такая процедура выполняется за 0(N) операций, где N — число городов. На втором шаге происходит подсчет стоимости путешествия по городам в указанном порядке. Для этого нам нужно просто просуммировать стоимости путешествия между последовательными парами городов в списке, что также требует 0(N) операций. Оба шага полиномиальны, поэтому задача о коммивояжере лежит в классе NP. Времяемкой делает ее именно необходимое число итераций этой процедуры.

Здесь следует отметить, что такую двухшаговую процедуру можно было применить к любой из рассматривавшихся нами ранее задач. Например, сортировку списка можно выполнять, генерируя произвольный порядок элементов исходного списка и проверяя, не является ли этот порядок возрастающим. Не относит ли это рассуждение задачу сортировки к классу NP? Конечно, относит. Разница между классом P и классом NP в том, что в первом случае у нас имеется детерминированный алгоритм, решающий задачу за полиномиальное время, а во втором мы такого алгоритма незнаем.

Сведение задачи к другой задаче

Один из способов решения задач состоит в том, чтобы свести, или редуцировать, одну задачу к другой. Тогда алгоритм решения второй задачи можно преобразовать таким образом, чтобы он решал первую. Если преобразование выполняется за полиномиальное время и вторая задача решается за полиномиальное время, то и наша новая задача также решается за полиномиальное время.

Поясним наше рассуждение примером. Пусть первая задача состоит и том, чтобы вернуть значение «да» в случае, если одна из данных булевских поименных имеет значение «истина», и вернуть «нет» в противоположном случае. Вторая задача заключается в том, чтобы найти максимальное значение в списке целых чисел. Каждая из них допускает простое ясное решение, но предположим на минуту, что мы знаем решение задачи о списке максимума, а задачу про булевские переменные решать не умеем. Мы хотим свести задачу о булевских переменных к задаче о максимуме целых чисел. Напишем алгоритм преобразования набора значений булевских переменных в список целых чисел, который значению «ложь» сопоставляет число 0, а значению «истина»— число 1. Затем воспользуемся алгоритмом поиска максимального элемента в списке. По тому, как составлялся список, заключаем, что этот максимальный элемент может быть либо нулем, либо единицей. Такой ответ можно преобразовать в ответ в задаче о булевских переменных, возвращая «да», если максимальное значение равно 1, и «нет», если оно равно 0.

Мы видели в главе 1, что поиск максимального значения выполняется за линейное время, а редукция первой задачи ко второй тоже требует линейного времени, поэтому задачу о булевских переменных тоже можно решить за линейное время.

В следующем разделе мы воспользуемся техникой сведения, чтобы кое-что узнать о NP задачах. Однако редукция NP задач может оказаться гораздо более сложной.

Понятие сложности вычислений. NP- полные задачи.

NP-полные задачи

При обсуждении класса NP следует иметь в виду, что наше мнение, согласно которому их решение требует большого времени, основано на том, что мы просто не нашли эффективных алгоритмов их решения. Может быть, посмотрев на задачу коммивояжера с другой точки зрения, мы смогли бы разработать полиномиальный алгоритм ее решения. То же самое можно сказать и про другие задачи, которые мы будем рассматривать в следующем параграфе.

Термин NP-полная относится к самым сложным задачам в классе NP. Эти задачи выделены тем, что если нам все-таки удастся найти полиномиальный алгоритм решения какой-либо из них, то это будет означать, что все задачи класса NP допускают полиномиальные алгоритмы решения.

Мы показываем, что задача является NP-полной, указывая способ выведения к ней всех остальных задач класса NP. На практике эта деятельность выглядит не столь уж устрашающе - нет необходимости осуществлять редукцию для каждой NP задачи. Вместо этого для того, чтобы доказать NP-полноту некоторой NP задачи A, достаточно свести к ней какую-нибудь NP-полную задачу B. Редуцировав задачу B к задаче A, мы показываем,

что и любая NP задача может быть сведена к A за два шага, первый из которых ее редукция к B.

В предыдущем разделе мы выполняли редукцию полиномиального алгоритма. Посмотрим теперь на редукцию алгоритма, решающего NP задачу. Нам понадобится процедура, которая преобразует все составные части задачи в эквивалентные составные части другой задачи. Такое преобразование должно сохранять информацию: всякий раз, когда решение первой задачи дает положительный ответ, такой же ответ должен быть и во второй задаче, и наоборот.

Гамильтоновым путем в графе называется путь, проходящий через каждую вершину в точности один раз. Если при этом путь возвращается в исходную вершину, то он называется гамильтоновым циклом. Граф, в котором есть гамильтонов путь или цикл, не обязательно является полным. Задача о поиске гамильтонова цикла следующим образом сводится к задаче о коммивояжере. Каждая вершина графа — это город. Стоимость пути вдоль каждого ребра графа положим равной 1. Стоимость пути между двумя городами, не соединенными ребром, положим равной 2. А теперь решим соответствующую задачу о коммивояжере. Если в графе есть гамильтонов цикл, то алгоритм решения задачи о коммивояжере найдет циклический путь, состоящий из ребер веса 1. Если же гамильтонова цикла нет, то в найденном пути будет по крайней мере одно ребро веса 2. Если в графе N вершин, то в нем есть гамильтонов цикл, если длина найденного пути равна N, и такого цикла нет, если длина найденного пути больше N.

В 1971 году Кук доказал NP-полноту обсуждаемой в следующем параграфе задачи о конъюнктивной нормальной форме. NP-полнота большого числа задач была доказана путем редукции к ним задачи о конъюнктивной нормальной форме. В книге Гэри и Джонсона, опубликованной в 1979 году, приведены сотни задач, NP-полнота которых доказана.

Редукция — настолько мощная вещь, что если любую из NP-полных задач удастся свести к задаче класса P, то и все NP задачи получат полиномиальное решение. До сих пор ни одна из попыток построить такое сведение не удалась.

Типичные NP задачи

Каждая из задач, которые мы будем обсуждать, является либо оптимизационной, либо задачей о принятии решения. Целью оптимизационной задачи обычно является конкретный результат, представляющий собой минимальное или максимальное значение. В задаче о принятии решения обычно задается некоторое пограничное значение, и нас интересует, существует ли решение, большее (и задачах максимизации) или меньшее (в задачах минимизации) указанной границы. Ответом в задачах оптимизации служит полученный конкретный результат, а в задачах о принятии решений — «да» или «нет».

Ранее мы занимались оптимизационным вариантом задачи о коммивояжере. Это задача минимизации, и нас интересовал путь минимальной стоимости. В варианте принятия решения мы могли бы спросить, существует ли путь коммивояжера со стоимостью, меньшей заданной константы C. Ясно, что ответ в задаче о принятии решения зависит от выбранной границы. Если эта граница очень велика (например, она превышает суммарную стоимость всех дорог), то ответ «да» получить несложно. Если эта граница чересчур мала (например, она меньше стоимости дороги между любыми двумя городами), то ответ «нет» также дается легко. В остальных промежуточных случаях время поиска ответа очень велико и сравнимо со временем решения оптимизационной задачи. Поэтому мы будем говорить вперемешку о задачах оптимизации и принятия решений, используя ту из них, которая точнее отвечает нашим текущим целям.

В следующих нескольких разделах мы опишем еще шесть NP задач — как в оптимизационном варианте, так и в варианте принятия решения.

Раскраска графа

Как мы уже говорили, граф G = (V, E) представляет собой набор вершин, или узлов, V и набор ребер E соединяющих вершины попарно. Здесь мы будем заниматься только неориентированными графами. Вершины графа можно раскрасить в разные цвета, которые обычно обозначаются целыми числами. Нас интересуют такие раскраски, в которых концы каждого ребра окрашены разными цветами. Очевидно, что в графе с N вершинами можно покрасить вершины в N различных цветов, но можно ли обойтись меньшим количеством цветов? В задаче оптимизации нас интересует минимальное число цветов, необходимых для раскраски вершин графа. В задаче принятия решения нас интересует, можно ли раскрасить вершины в C или менее цветов.

У задачи о раскраске графа есть практические приложения. Если каждая вершина графа обозначает читаемый в колледже курс, и вершины соединяются ребром, если есть студент, слушающий оба курса, то получается весьма сложный граф. Если предположить, что каждый студент слушает 5 курсов, то на студента приходится 10 ребер. Предположим, что на 3500 студентов приходится 500 курсов. Тогда у получившегося графа будет 500 вершин и 35 000 ребер. Если на экзамены отведено 20 дней, то это означает, что вершины графа нужно раскрасить в 20 цветов, чтобы ни у одного студента не приходилось по два экзамена в день.

Разработка бесконфликтного расписания экзаменов эквивалентна раскраске графов. Однако задача раскраски графов принадлежит к классу NP, поэтому разработка бесконфликтного расписания за разумное время невозможна. Кроме того при планировании экзаменов обычно требуется, чтобы у студента было не больше двух экзаменов в день, а экзамены по различным частям курсам назначаются в один день. Очевидно, что разработка «совершенного» плана экзаменов невозможна, и поэтому необходима другая техника для получения по крайней мере неплохих планов.

Раскладка по ящикам

Пусть у нас есть несколько ящиков единичной емкости и набор объектов различных размеров $s_1, s_2, ..., s_N$. В задаче оптимизации нас интересует наименьшее количество ящиков, необходимое для раскладки всех объектов, а в задаче принятия решения — можно ли упаковать все объекты в B или менее ящиков.

Эта задача возникает при записи информации на диске или во фрагментированной памяти компьютера, при эффективном распределении груза на кораблях, при вырезании кусков из стандартных порций материала по заказам клиентов. Если, например, у нас есть большие металлические листы и список заказов на меньшие листы, то естественно мы хотим распределить заказы как можно плотнее, уменьшив тем самым потери и увеличив доход.

Упаковка рюкзака

У нас имеется набор объектов объемом $s_1, ..., s_N$ стоимости $w_1, ..., w_N$. В задаче оптимизации мы хотим упаковать рюкзак объемом K так, чтобы его стоимость была максимальной. В задаче принятия решения нас интересует, можно ли добиться, чтобы суммарная стоимость упакованных объектов была по меньшей мере W.

Эта задача возникает при выборе стратегии вложения денег: объемом здесь является объем различных вложений стоимостью - предполагаемая величина дохода, а объем рюкзака определяется размером планируемых капиталовложений.

Задача о суммах элементов подмножеств

Пусть у нас есть множество объектов различных размеров $s_1, ..., s_N$ и некоторая положительная верхняя граница L. В задаче оптимизации нам необходимо найти набор объектов, сумма размеров которых наиболее близка к L и не превышает этой верхней границы.

В задаче принятия решения нужно установить, существует ли набор объектов с суммой размеров L. Это упрощенная версия задачи об упаковке рюкзака.

Задача об истинности КНФ-выражения

Конъюнктивная нормальная форма (КНФ) представляет собой последовательность булевских выражений, связанных между собой операторами AND (обозначаемыми $\wedge$), причем каждое выражение является мономом от булевских переменных или их отрицаний, связанных операторами OR (которые обозначаются через $\vee$). Вот пример булевского выражения в конъюнктивной нормальной форме (отрицание обозначается чертой над именем переменной):

$$(a \vee b) \wedge (\bar{a} \vee c) \wedge (a \vee b \vee \bar{c} \vee d) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (a \vee \bar{b} \vee \bar{c} \vee \bar{d} \vee e).$$

Задача об истинности булевского выражения в конъюнктивной нормальной форме ставится только в варианте принятия решения: существуют ли у переменных, входящих в выражение, такие значения истинности, подстановка которых делает все выражение истинным. Как число переменных, так и сложность выражения не ограничены, поэтому число комбинаций значений истинности может быть очень велико.

Задача планирования работ

Пусть у нас есть набор работ, и мы знаем время, необходимое для завершения каждой из них, $t_1, t_2, ..., t_N$, сроки $d_1, d_2, ..., d_N$, к которым эти работы должны быть обязательно завершены, а также штрафы $p_1, p_2, ..., p_N$, которые будут наложены при незавершении каждой работы в установленные сроки. Задача оптимизации требует установить порядок работ, минимизирующий накладываемые штрафы. В задаче принятия решений мы спрашиваем, есть ли порядок работ, при котором величина штрафа будет не больше P.

Вопросы для самопроверки.

1. Что такое алгоритм?
2. Перечислите основные свойства алгоритмов.
3. Назовите универсальные алгоритмические модели.
4. Дайте определение примитивно-рекурсивных функций.
5. Дайте определение частично рекурсивных и общерекурсивных функций.
6. Дайте определение машины Тьюринга.
7. Дайте определение и приведите примеры полиномиальных алгоритмов.
8. В чем выражается вычислительная сложность алгоритмов?
9. Какая задача считается труднорешаемой?
10. Что означает термин NP- полная задача?

1. Оценить применимость алгоритма.

Агенство недвижимости, база данных. Запись – пара (предложение, спрос). Найти варианты обмена (т.е. такие пары, где первая компонента одной совпадает со второй компонентой другой). Оценить простейший вариант поиска – «лобовой».

Решение. Трудоемкость $n \times (n-1)/2$. Если на одну проверку нужна 1 миллисекунда, то при n = 100 потребуется около 5 секунд, при n=100 000 – 5×10^6 сек, т.е. около 1389 часов. Алгоритм непригодный.

3.15-16.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия об основных подходах к формализации понятия алгоритма (машина Тьюринга; рекурсивный алгоритм, нормальные алгоритмы Маркова); понятие эффективности и сложности алгоритмов;
- приобрели умения и навыки алгоритмизации простейших задач.

3.17 Практическое занятие №ПЗ-17 (2 часа).
Тема: « Конечные автоматы»

3.17.1 Задание для работы:
1. Понятие конечного автомата. Историческая справка. Способы задания конечного автомата. Примеры конечных автоматов. Виды автоматов. Общие задачи теории автоматов.

3.17.2 Краткое описание проводимого занятия:
1. Понятие конечного автомата. Историческая справка. Способы задания конечного автомата. Примеры конечных автоматов. Виды автоматов. Общие задачи теории автоматов.

1.Раздел дискретной математики, изучающий модели преобразователей дискретной информации, называется теорией
+а) автоматов
б) вероятностей
в) множеств
г) функций
д) поля

2. Конечный автомат это математическая модель дискретного устройства по переработке
+а) информации
б) вероятности
в) алгебры
г) формул
д) поля

3. В конечном автомате $A=\{X;Q;Y;\lambda(x,q);\delta(x,q)\}$ множество X называется
+а) входным алфавитом
б) множеством состояний
в) выходным алфавитом
г) функцией переходов
д) функцией выходов

4. В конечном автомате $A=\{X;Q;Y;\lambda(x,q);\delta(x,q)\}$ множество Q называется
+а) множеством состояний
б) входным алфавитом
в) выходным алфавитом
г) функцией переходов
д) функцией выходов

5. В конечном автомате $A=\{X;Q;Y;\lambda(x,q);\delta(x,q)\}$ множество Y называется
+а) выходным алфавитом
б) входным алфавитом
в) множеством состояний
г) функцией переходов
д) функцией выходов

6. В конечном автомате $A=\{X;Q;Y;\lambda(x,q);\delta(x,q)\}$ объект $\lambda(x,q)$ называется
+а) функцией переходов
б) входным алфавитом
в) выходным алфавитом
г) множеством состояний

д) функцией выходов

7. В конечном автомате $A=\{X;Q;Y;\lambda(x,q);\delta(x,q)\}$ объект $\delta(x,q)$ называется

+а) функцией выходов
б) входным алфавитом
в) выходным алфавитом
г) множеством состояний
д) функцией переходов

8. Задан конечный автомат $A=\{X;Q;Y;\lambda(x,q);\delta(x,q)\}$ - элемент задержки (элемент памяти): $X=\{0,1\}$, $Q=\{0,1\}$, $Y=\{0,1\}$, функция переходов $\lambda(0,0)=0$, $\lambda(0,1)=0$, $\lambda(1,0)=1$, $\lambda(1,1)=1$, функция выходов $\delta(0,0)=0$, $\delta(0,1)=1$, $\delta(1,0)=0$, $\delta(1,1)=1$. При входном сигнале $x_1=0$ в состоянии $q_2=1$ автомат выдаёт выходной сигнал-…

ОТВЕТ:1

9. Задан автомат $A=\{X;Q;Y;\lambda(x,q);\delta(x,q)\}$ - элемент задержки (элемент памяти): $X=\{0,1\}$, $Q=\{0,1\}$, $Y=\{0,1\}$, функция переходов $\lambda(0,0)=0$, $\lambda(0,1)=0$, $\lambda(1,0)=1$, $\lambda(1,1)=1$, функция выходов $\delta(0,0)=0$, $\delta(0,1)=1$, $\delta(1,0)=0$, $\delta(1,1)=1$. При входном сигнале $x_2=1$ в состоянии $q_1=0$ автомат выдаёт выходной сигнал-…

ОТВЕТ:0

10. Автомат $A=\{X;Q;Y;\lambda(x,q);\delta(x,q)\}$ - элемент задержки (элемент памяти): $X=\{0,1\}$, $Q=\{0,1\}$, $Y=\{0,1\}$, функция переходов $\lambda(0,0)=0$, $\lambda(0,1)=0$, $\lambda(1,0)=1$, $\lambda(1,1)=1$, функция выходов $\delta(0,0)=0$, $\delta(0,1)=1$, $\delta(1,0)=0$, $\delta(1,1)=1$. При входном сигнале $x_1=0$ из состояния $q_2=1$ автомат переходит в состояние-…

ОТВЕТ:0

11. Задан конечный автомат $A=\{X;Q;Y;\lambda(x,q);\delta(x,q)\}$ - (элемент памяти): $X=\{0,1\}$, $Q=\{0,1\}$, $Y=\{0,1\}$, функция переходов $\lambda(0,0)=0$, $\lambda(0,1)=0$, $\lambda(1,0)=1$, $\lambda(1,1)=1$, функция выходов $\delta(0,0)=0$, $\delta(0,1)=1$, $\delta(1,0)=0$, $\delta(1,1)=1$. При входном сигнале $x_2=1$ из состояния $q_1=0$ автомат переходит в состояние-…

ОТВЕТ:1

3.17.3 Результаты и выводы: в результате проведенного занятия студенты:

- освоили понятия конечного автомата, способы задания конечного автомата, примеры конечных автоматов, виды автоматов;
- приобрели умения и навыки решения простейших задач по теме «Автоматы».

3.18 Практическое занятие №18 (2 часа).

Тема: «Исчисление высказываний и предикатов. Математические (формальные аксиоматические) теории первого порядка**»**

3.18.1 Задание для работы:

1. Формальные системы.
2. Исчисление высказываний.

3.18.2 Краткое описание проводимого занятия:

1. Формальные системы.
2. Исчисление высказываний.

Формальные системы - это системы операций над объектами, понимаемыми как последовательность символов (т.е. как слова в фиксированных алфавитах), сами операции также являются операциями над символами. Термин "формальный" подчёркивает, что объекты и операции над ними рассматриваются чисто формально, без каких бы то ни было содержательных интерпретаций символов. Предполагается, что между символами не существует никаких связей и отношений, кроме тех, которые явно описаны средствами самой формальной системы.

Исторически теория формальных систем, так же как и теория алгоритмов, возникла в рамкам оснований математики при исследовании строения аксиоматических теорий и методов доказательства в таких теориях. Всякая точная теория определяется, во-первых, языком, т.е. некоторым множеством высказываний, имеющих смысл с точки зрения этой теории, и, во-вторых, совокупностью теорем - подмножеством языка, состоящим из высказываний, истинных в данной теории.

В математике с античных времён существовал образец систематического построения теории - геометрия Евклида, в которой все исходные предпосылки сформированы явно, в виде аксиом, а теоремы выводятся из этих аксиом с помощью цепочек логических рассуждений, называемых доказательствами. Однако, до середины 19 века математические теории, как правило, не считали нужным явно выделять все исходные принципы, критерии же строгости доказательств и очевидности утверждений в разные времена были различными и явно не формулировались. Время от времени это приводило к необходимости пересмотра основ той или иной теорий. Известно, например, что основания дифференциального и интегрального счисления, разработанных в 18 век Ньютоном и Лейбницем, в 19 века подверглись серьёзному пересмотру. Математический анализ в его современном виде опирается на работы Коши, Больцано и Вейерштрасса по теории пределов.

В конце 19 века такой пересмотр затронул общие принципы доказательств в математических теориях. Это привело к созданию новой отрасли математики - оснований математики, предметом которой и стало построение теорий, чтобы в них не возникало противоречий. Одной из фундаментальных идей, на которые опираются исследования по основанию математики, является идея формализации теорий, т.е. последовательного проведения аксиоматического метода построения теорий.

При этом не допускается пользоваться какими-либо предположениями об объектах теории, кроме тех, которые выражены явно в виде аксиом; аксиомы рассматриваются как формальные последовательности символов (выражения), а методы доказательств— как методы получения одних выражении из других с помощью операций над символами. Такой подход гарантирует четкость исходных утверждений и однозначность выводов, однако может создаться впечатление, что осмысленность и истинность в формализованной теории не играют никакой роли. Внешне это так, однако, в действительности и аксиомы и правила вывода стремятся выбирать таким образом, чтобы построенной с их помощью формальной теории можно было придать содержательный смысл.

Более конкретно *формальная система* (или *исчисление)* строится следующим образом.

1. Определяется некоторое счетное множество символов, т.е. множество, элементы которого могут быть взаимно однозначно сопоставлены элементам натурального ряда 1,2,...N, которые называется термами. Имеется другое конечное множество символов, элементы которого называются связками или операциями. Наконец, существует конечное множество вспомогательных символов. Конечные последовательности символов называются выражениями данной системы.

2. Определяется *множество формул,* или правильно построенных выражений, образующее язык теории. Это множество задается конструктивными средствами (как правило, индуктивным определением) и, следовательно, перечислимо. Обычно оно и разрешимо. Для правильно построенных формул (ППФ) задаются правила их конструирования, т.е. определяется эффективная процедура, с помощью которой по данному выражению выяс-

няется, является ли формула правильно построенной в данной формальной системе (ФС) или нет. Формула, для которой существует такая процедура, называется разрешимой в данной ФС, в противном случае неразрешимой. Иначе говоря, для неразрешимых формул нельзя построить алгоритм выяснения свойства формулы быть теоремой, для этого требуются все новые и новые озарения (изобретательства), не поддающиеся формализации.

3. Выделяется подмножество формул, называемых *аксиомами* ФС. Так же как и для ППФ для аксиом должна иметься процедура, позволяющая определить, является ли ППФ аксиомой или нет. Подмножество может быть и бесконечным, во всяком случае, оно должно быть разрешимо.

4. Задается конечное множество R_1, $R_{,2,..,}R_k$ отношений между ППФ, называемых правилами вывода. Должна иметься эффективная процедура, позволяющая для произвольной конечной последовательности ППФ решить, может ли каждый член этой последовательности быть выведен с помощью конечного числа правил вывода. Правило вывода $R(F_1, ..., F_n, G)$ —это вычислимое отношение на множестве формул. Если формулы $F_1, ..., F_n, G$ находятся в отношении R, то формула G называется *непосредственно выводимой* из $F_1, ..., F_n$ по правилу R. Часто правило $R(F_1, ..., F_n, G)$ записывается в виде $(F_1, ..., F_n)/G$. Формулы $F_1, ..., F_n$ называются *посылками* правила R, а G—его следствием или *заключением.* Примеры аксиом и правил вывода будут приведены несколько позднее.

Выводом формулы B из формул $A_1, ..., A_n$ называется последовательность формул $F_1, ..., F_m$, такая, что $F_m = B$, а любая $F_i(i = 1,...,m)$ есть либо аксиома, либо одна из исходных формул $A_1, ..., A_n$, либо непосредственно выводима из формул $F_1, ..., F_{i-1}$ (или какого-то их подмножества) по одному из правил вывода. Если существует вывод B из $A_1, ..., A_n$, то говорят, что B *выводима* из $A_1, ..., A_n$. Этот факт обозначается так: $A_1,...,An \vdash B$. Формулы $A_1, ..., A_n$ называются гипотезами или посылками вывода. Переход в выводе от F_{i-1} к F_i называется i-м *шагом вывода.*

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 формулой, *доказуемой* в теории Т, или *теоремой* теории T; факт доказуемости B обозначается $\vdash B$.

Очевидно, что присоединение формул к гипотезам не нарушает выводимости. Поэтому *если* $\vdash B$, то $A \vdash B$, и если $A_1, ..., A_n \vdash B$, то $A_1, ..., A_n, A_{n+1} \vdash B$ для любых A и A_{n+1}. Порядок гипотез в списке несуществен.

Например, если удалось построить вывод B из $A_1, ..., A_n$, то элементы последовательности ППФ $A_1, ..., A_n$ называются посылками вывода (или гипотезами). Сокращенно вывод B из $A_1, ..., A_n$ записывается в виде $A_1, ..., A_n \vdash B$, или если $\Gamma = \{A_1,.., A_n\}$ то $\Gamma \vdash B$. Напомним, что вывод ППФ без использования посылок есть доказательство ППФ В, а сама В – теорема, и это записывается $\vdash B$.

3.18.3 Результаты и выводы: в результате проведенного занятия студенты:
- познакомились с понятием формальной системы, историей возникновения понятия формальной системы, принципами построения.

4. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРОВЕДЕНИЮ СЕМИНАРСКИХ ЗАНЯТИЙ

Семинарские занятия не предусмотрены рабочим учебным планом.