Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВО «МГТУ «СТАНКИН»)

Ю.В. Елисеева, Е.М. Красикова,
В.Г. Костылев, Л.А. Уварова

ЗАДАЧИ ПО МАТЕМАТИЧЕСКОЙ ЛОГИКЕ И ТЕОРИИ АЛГОРИТМОВ

Учебное пособие

Москва 2017

УДК 519.6
ББК 22.12
Е51

Е51 Елисеева Ю.В., Красикова Е.М., Костылев В.Г., УвароваЛ.А. *Задачи по математической логике и теории алгоритмов*: учебное пособие / М.: Янус-К, 2017. –80 с.

ISBN 978-5-8037-0725-7

Учебное пособие содержит задачи по курсу «Математическая логика и теория алгоритмов». Включает разделы математической логики и теории алгоритмов. В каждом разделе приведены необходимые краткие теоретические сведения, а также приводятся примеры решений типовых задач.

Данное пособие предназначено для студентов различных институтов МГТУ «СТАНКИН».

УДК 519.6
ББК 22.12

ISBN 978-5-8037-0725-7

Оглавление

Глава 1. МАТЕМАТИЧЕСКАЯ ЛОГИКА

§ 1. Алгебра высказываний

1. Высказывания. Под высказыванием принято понимать *повествовательное* предложение, о котором можно однозначно заключить, *истинно (И)*оно или *ложно(Л)*. Значения истина (И) или ложь (Л), приписываемые высказыванию, называют *истинностными значениями* высказывания. Они формируют *множество истинностных значений* $M = \{И, Л\}$. Высказывания обозначаются прописными латинскими буквами A, B, C, P, Q и т.д. Высказывания, как и повествовательные предложения, могут быть *простыми* или *сложными.* Высказывание, не содержащее логических связок, называется *простым*. Высказывание, содержащее логические связки, называется *сложным*.

Логические связки в алгебре высказываний: *отрицание, дизъюнкция, конъюнкция, импликация и эквиваленция.* Истинностные значения сложных высказываний однозначно определены истинностными значениями входящих в них простых высказываний. В таблицах 1, 2 представлены значения сложных высказываний, содержащие основные логические связки.

Таблица 1

A	$\overline{A}$
Л	И
И	Л

Таблица 2

A	B	$A \vee B$	$A \wedge B$	$A \to B$	$A \sim B$
Л	Л	Л	Л	И	И
Л	И	И	Л	И	Л
И	Л	И	Л	Л	Л
И	И	И	И	И	И

2.Формула логики высказываний. *Формула* в логике высказываний строится индуктивно. Вводится *алфавит* символов:

– высказывательных переменных (обозначаются, как и высказывания, прописными латинскими буквами A, B, C, P, Q …)

– логических связок;

– открывающихся и закрывающихся скобок ( ).

Словом в данном алфавите называется любая конечная последовательность символов. Формула над данным алфавитом – это слово, составленное по следующим правилам.

1. Любая высказывательная переменная – атомарная формула;
2. Слова ($\overline{A}$), (A$\wedge$ B), (A$\vee$B), (A$\to$B), (A~B) являются формулами, если известно, что A и B – формулы ;

3. Других формул, кроме построенных согласно требованиям 1,2, нет.

Формулы будем обозначать буквами Φ, Φ_i. Если всем высказывательным переменным формулы Φ приписывать значения из множества $M=\{И, Л\}$, то формула задаст *истинностную функцию*, принимающую значения из множества $M=\{И, Л\}$. Значения данной функции часто называют *значениями формулы*. Две формулы Φ_1 и Φ_2, зависящие от одинаковых переменных, называются *равносильными (эквивалентными),* если при любых значениях этих переменных они принимают одинаковые значения.

Равносильность формул будем записывать через знак равенства, т.е. $\Phi_1 = \Phi_2$.

Список основных равносильных формул, которые являются *основными законами логики высказываний* приведён в Приложении I.

Тавтологией или *тождественно истинной формулой* называется формула, принимающая значение И при *всех* значениях входящих в нее высказывательных переменных. *Противоречием* или *тождественно ложной формулой* называется формула, принимающая значение Л при *всех значениях* входящих в неё высказывательныхпеременных. Основные законы логики высказываний для тавтологий (И) и противоречий (Л) (*законы логических констант*) приведены в Приложении I.

Пример 1. Доказать, что формула $\Phi_1 = (\overline{A \sim B})$ равносильна формуле

$$\Phi_2 = (A \wedge \overline{B}) \vee (\overline{A} \wedge B).$$

◀Используем основные законы логики (см. Приложение I).Применив к формуле Φ_1 закон 20, а затем второй закон де Моргана, получим:

$$\Phi_1 = \overline{(A \wedge B) \vee (\overline{\overline{A}} \wedge \overline{\overline{B}})} = \overline{(A \wedge B)} \wedge (\overline{\overline{A} \wedge \overline{B}}).$$

Применяя к полученной формуле первый закон де Моргана, а затем закон 11, получим:

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

К последней формуле применим законы 1,8, 14, 16, получим:

$$((A \vee B) \wedge \overline{A}) \vee ((A \vee B) \wedge \overline{B}) = (\overline{A} \wedge A) \vee (\overline{A} \wedge B) \vee (\overline{B} \wedge A) \vee (\overline{B} \wedge B) =$$
$$= Л \vee (\overline{A} \wedge B) \vee (\overline{B} \wedge A) \vee Л = (\overline{A} \wedge B) \vee (\overline{B} \wedge A).$$

Применив к полученной формуле закон 1, получим формулу Φ_2. Следовательно, эти формулы равносильны.

В математической логике важным является понятие *двойственной формулы*. Напомним, что *булева формула* может содержать только три логических связки: отрицание, конъюнкцию и дизъюнкцию. Пусть Ф – булева формула, тогда формула Φ^* – двойственная по отношению к Ф, если все вхождения связки $\vee$ в Ф заменить на $\wedge$ в Φ^*, а все вхождения связки $\vee$ в Ф заменить на $\vee$ в Φ^*.

Пример 2. Построить двойственную к формуле$\Phi=(A\vee C)\wedge(B\vee\overline{C})$.
◄$\Phi^*=(A\wedge C)\vee(B\wedge\overline{C})$

С помощью таблиц истинности проверить равносильность формул Φ_1 *и* Φ_2.

1.1 $\Phi_1 = A\vee B \sim A\vee(\overline{A}\wedge B)$, $\Phi_1 = A\wedge(B\vee C)\sim(A\wedge B)\vee(A\wedge C)$

1.2 $\Phi_1 = \overline{A\wedge B\vee\overline{C}}$, $\Phi_2 = (\overline{A}\wedge C)\vee(\overline{B}\wedge C)$

1.3 $\Phi_2 = (A\wedge B)\vee(A\wedge\overline{B})\sim A$, $\Phi_2 = (A\vee B)\wedge(A\vee\overline{B})\sim A$

1.4 $\Phi_1 = ((A\to B)\to A)\to A$, $\Phi_2 = (A\to B)\to(\overline{B}\to\overline{A})$

С помощью основных законов проверить равносильность Φ_1 *и* Φ_2*:*

1.5 $\Phi_1=(A\vee B)\wedge(A\vee C)\wedge(B\vee D)\wedge(C\vee D)$, $\Phi_2 = (A\wedge D)\vee(B\wedge C)$

1.6 $\Phi_1= (A\wedge B)\vee((A\vee B)\wedge(\overline{A}\vee\overline{B}))$, $\Phi_2 = A\vee B$

По таблице истинности проверить, что Ф *– тавтология*

1.7 $\Phi = (A\to B)\to((B\to C)\to(A\to C))$

1.8 $\Phi =(A\to(B\to C))\to((A\to B)\to(A\to C))$

3. Проблема разрешимости в логике высказываний. *Проблемой разрешимости* для логики высказываний называют следующую: существует ли такая процедура, которая позволяет для *произвольной* формулы за конечное число шагов определить, является ли она тавтологией.

Данная проблема разрешима с помощью процедуры приведения к *конъюнктивной нормальной форме (КНФ).* Отметим, что процедура проверки, является ли формула противоречием, решается с помощью *дизъюнктивной нормальной формы* формулы, или ДНФ**.**

Формула называется *элементарной конъюнкцией (дизъюнкцией)*, если она является конъюнкцией (дизъюнкцией) переменных и отрицаний переменных.

Дизъюнктивной нормальной формой (ДНФ) формулы называется дизъюнкция конечного числа элементарных конъюнкций. Конъюнктивной нормальной формой (КНФ) называется конъюнкция конечного числа элементарных дизъюнкций.

После приведения формулы к ДНФ или к КНФ можно легко решить проблему разрешимости, используя следующие утверждения.

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

Формула является тождественно ложной (противоречием) в том и только в том случае, если в ее ДНФ *каждая* элементарная конъюнкция содержит в качестве конъюнктивных членов какую-нибудь переменную и ее отрицание.

Пример 3. Привести формулу Ф=$(\mathrm{A}\to(\mathrm{B}\wedge\overline{C}))\to(A\sim C)$ к виду ДНФ и проверить на противоречие.

◄ 1) Приведём формулу к булевой форме:

$$\overline{A}\vee(\mathrm{B}\wedge\overline{C})\to(A\wedge C)\vee(\overline{A}\wedge\overline{C})=\overline{\overline{A}\vee(\mathrm{B}\wedge\overline{C})}\vee(A\wedge C)\vee(\overline{A}\wedge\overline{C});$$

2) Спустим отрицания до переменных по законам 6:

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

3) Раскроем скобки, используя законы 3 и упростим формулу согласно законам 4: $(A\wedge\overline{B})\vee(A\wedge C)\vee(A\wedge C)\vee(\overline{A}\wedge\overline{C})=(A\wedge\overline{B})\vee(A\wedge C)\vee(\overline{A}\wedge\overline{C})$.

После преобразований мы получили формулу в виде ДНФ (ДНФ – логическая сумма произведений переменных и (или) отрицаний переменных). При проверке, является ли формула противоречием, замечаем, что первый терм не содержит переменной и её отрицания (при этом уже не имеет значения, какой вид имеют следующие термы), следовательно, данная формула не является противоречием.

Пример 4. Проверить, является ли формула Ф=$(\mathrm{A}\to\mathrm{B})\to((\mathrm{C}\vee\mathrm{A})\to$ $\to(\mathrm{C}\vee\mathrm{B}))$ тавтологией с помощью КНФ.

◄ 1) По аналогии с предыдущим примером, приведём формулу к ДНФ:

$$\Phi=(\overline{\mathrm{A}}\vee \mathrm{B})\to(\overline{C}\wedge\overline{A})\vee(C\vee B)=\overline{(\overline{\mathrm{A}}\vee \mathrm{B})}\vee(\overline{C}\wedge\overline{A})\vee(C\vee B)=A\wedge\overline{B}\vee\overline{C}\wedge\overline{A}\vee C\vee B.$$

2) Построим двойственную к Φ:
$\Phi^{*}=(A\wedge\overline{B}\vee\overline{C}\wedge\overline{A}\vee C\vee B)^{*}=(A\vee\overline{B})\wedge(\overline{C}\vee\overline{A})\wedge C\wedge B,$ и снова приводим её к ДНФ: $\Phi^{*}=A\overline{C}CB\vee A\overline{A}CB\vee\overline{B}\overline{C}CB\vee\overline{B}\overline{A}CB$.
3) Найдем двойственную к Φ^{*} и получим основную формулу $\Phi^{**}=\Phi$, находящуюся в КНФ:

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

По полученной КНФ определим, является ли формула ли тавтологией. Так как каждый терм содержит некоторую переменную и её отрицание, то формула является тавтологией.

Приведением Φ *к ДНФ установить, является ли* Φ *тождественно ложной:*

1.9 Φ = $\overline{(A\vee\overline{B})}\to(B\wedge\overline{A})$

1.10 Φ = $(A\to B\wedge\overline{C})\to(A\sim C)$

Приведением Φ *к КНФ установить, является ли* Φ *тавтологией:*

1.11 Φ = $(\overline{A}\to\overline{B})\to((\overline{A}\to B)\to A)$

1.12 Φ = $(A\to B)\to((C\vee A)\to(C\vee B))$

1.13 $\Phi=(A\to B)\to(C\vee A)$

1.14 $\Phi=(A\to B)\to((A\to\overline{B})\to\overline{A})$

1.15 $\Phi=(\overline{A}\to\overline{B})\to(B\to A)$

1.16 $\Phi=(\overline{A}\to\overline{B})\sim(B\sim C)$
Решить задачи с помощью формул логики высказываний

1.17 Иванову, Петрову и Сидорову предъявлено обвинение в соучастии в ограблении банка. Похитители скрылись на поджидавшем их автомобиле. На следствии Иванов показал, что преступники были на синем «Мерседесе», Петров сказал, что был черный джип, а Сидоров утверждал, что это была «Жигули», определенно не синего цвета. Стало известно, что, желая запутать следствие, каждый из них указал правильно либо марку автомобиля, либо его цвет. На каком автомобиле скрылись преступники?

1.18. На вопрос, кто из трех учащихся изучал логику, был получен правильный ответ: если изучал первый, то изучал и второй; но неверно, что если изучал третий, то изучал и второй. Кто из трех учащихся изучал логику?

4. Логичные, или правильные рассуждения. *Логическое рассуждение* представляет собой последовательность высказываний (гипотез, или посылок), на основании которых делается логический вывод D (заключение). Рассуждение называется *логичным (или правильным),* если всякий раз, когда *все* гипотезы истинны, истинно также и заключение. Данное определение равносильно следующему: если заключение ложно, то *хотя бы одна* из гипотез также является ложной. Формальная запись рассуждения $P_1, P_2, \ldots, P_n \vdash D$ называется *клаузой.* Из определения логичности следует, что клауза $P_1, P_2, \ldots, P_n \vdash D$ верная (а рассуждение логично) тогда и только тогда, когда формула

$$P_1 \wedge P_2 \wedge \ldots \wedge P_n \to D$$

является тавтологией, или тогда и только тогда, когда формула

$$P_1 \wedge P_2 \wedge \ldots \wedge P_n \wedge \bar{D}$$

является противоречием (тождественно ложной).

Пример 6.Проверить, является ли рассуждение логичным (правильным). Известно, что всякий раз, когда сверкнёт молния, грянет гром. Молния сверкнула, следовательно, обязательно грянет гром (правило *отделения.*или modusponens).

◀ ***(1 способ).*** Запишем рассуждение в форме клаузы, используя обозначения для высказываний (М – молния сверкнула, Г – гром грянул):
$M \to Г, M \vdash Г$ и составим формулу $Ф = (M \to Г) \wedge M \to Г$, которую проверим на тавтологию. Получим ДНФ составленной формулы
$Ф = \overline{(\bar{M} \vee Г) \wedge M} \vee Г = M \wedge \bar{Г} \vee \bar{M} \vee Г$, которую можно привести к КНФ согласноПримеру 4:
$Ф^{**} = ((M \vee \bar{Г}) \wedge \bar{M} \wedge Г)^{*} = (M\bar{M}Г \vee \bar{Г}\bar{M}Г)^{*} = (M \vee \bar{M} \vee Г)(\bar{Г} \vee \bar{M} \vee Г)$. Следовательно, формула является тавтологией (в каждой элементарной дизъюнкции есть буква и её отрицание), а рассуждение логично.

(2 способ). Для клаузы $M \to Г, M \mid - Г$ составим формулу $\Psi = (M \to Г) \wedge M \wedge \overline{Г}$ и проверим её на противоречие, приведя к ДНФ: $\Psi = (\overline{M} \vee Г) \wedge M \wedge \overline{Г} = (\overline{M} \wedge M \wedge \overline{Г}) \vee (Г \wedge M \wedge \overline{Г})$, следовательно, формула является противоречием, а рассуждение логично.

Проверить, является ли рассуждение логичным

1.19 В бюджете возникнет дефицит, если не повысят пошлины. Если в бюджете возникнет дефицит, то расходы на социальные нужды сократятся. Следовательно, если повысят пошлины, то расходы на социальные нужды не сократятся.

1.20 Обучающиеся на факультете ИНТЕХ студенты способны или прилежны. Если они прилежны, то регулярно занимаются. Значит, если они не занимаются регулярно, то они способны.

§ 2. Метод резолюций в логике высказываний

1. Доказательство по правилу резолюций. Логичность любого рассуждения можно также проверить, используя алгоритм метода *резолюций.* Доказательство использует *правило резолюций*

$$A \vee B, \overline{A} \vee C \mid - B \vee C$$

и следующее определение *вывода формулы по правилу резолюций.*

Пусть каждая из формул $S_1, S_2, \ldots, S_p$ является элементарной дизъюнкцией. Тогда формула D выводится из $S_1, S_2, \ldots, S_p$ *по правилу резолюций,* если существует доказательство (список вывода): $D_1, D_2, \ldots, D_m$, удовлетворяющее требованиям:

1) $D_m = D$.
2) Любая D_i, $i \le m$ является или одной из гипотез $S_1, S_2, \ldots, S_p$, или получена из *предыдущих формул* в списке вывода по правилу резолюций.

Можно доказать *полноту метода резолюций*, означающую, что рассуждение $S_1, S_2, \ldots, S_p \mid - D$ логично тогда и только тогда, когда D выводится из $S_1, S_2, \ldots, S_p$ по правилу резолюций .

2. **Алгоритм метода резолюций**. В более общем случае для доказательства логичности рассуждения $P_1, P_2, \ldots, P_n \mid - D$ используется следующий *алгоритм метода резолюций:*

I. Записать $P_1, P_2, \ldots, P_n \mid - D$ в *форме конъюнктивного противоречия* $P_1, P_2, \ldots, P_n, \overline{D} \mid - Л$ (Л – тождественно-ложная формула).

II. Привести каждую из формул $P_1, P_2, \ldots, P_n, \overline{D}$ к КНФ в *форме предложения* (составив список элементарных дизъюнкций $S_1, S_2, \ldots, S_p$, разделённых запятыми)

III. Вывести Л из $S_1, S_2, \ldots, S_p$ по правилу резолюций.

Пример 1. Доказать клаузу методом резолюций:

$$(A \to C) \wedge (B \to C), C \to A \wedge D \mid -(A \vee B) \to A \wedge D.$$

◀

1. Составим новую клаузу в форме конъюнктивного противоречия $(A \to C) \wedge (B \to C), C \to A \wedge D, \overline{(A \vee B) \to A \wedge D} \big| - Л$.
2. Приведем каждую формулу слева к КНФ в форме предложения:

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

$$C \to A \wedge D = \overline{C} \vee A \wedge D = (\overline{C} \vee A) \wedge (\overline{C} \vee D) \Rightarrow (\overline{C} \vee A), (\overline{C} \vee D);$$

$$\overline{(A \vee B) \to A \wedge D} = \overline{\overline{(A \vee B)} \vee A \wedge D} = (A \vee B) \wedge \overline{A \wedge D} = (A \vee B) \wedge (\overline{A} \vee \overline{D}) \Rightarrow (A \vee B), (\overline{A} \vee \overline{D}).$$

Подставим все полученные предложения вместо соответствующих формул и составим новый вывод:

$$(\overline{A} \vee C), (\overline{B} \vee C), (\overline{C} \vee A), (\overline{C} \vee D), (A \vee B), (\overline{A} \vee \overline{D}) \big| - Л .$$

Доказываем вывод, используя правило резолюций:

1. $(\overline{A} \vee C), (\overline{C} \vee D) \big| - \overline{A} \vee D$,
2. $(\overline{A} \vee \overline{D}), \overline{A} \vee D \big| - \overline{A}$,
3. $\overline{A}, (A \vee B) \big| - B$,
4. $(\overline{C} \vee A), \overline{A} \big| - \overline{C}$,
5. $(\overline{B} \vee C), \overline{C} \big| - \overline{B}$,
6. $\overline{B}, B \big| - Л$.

Доказать клаузы тремя способами, используя проверку на тавтологию, противоречие и метод резолюций:

1.21 $A \to B, B \to C \vert\!- A \to C$ **1.22** $A \to B, \overline{B} \vert\!- \overline{A}$

1.23 $A \to C, B \to C, A \vee B \vert\!- C$ **1.24** $A \to C, B \to D, A \vee B \vert\!- C \vee D$

1.25 $A \to B, A \to C, \overline{C} \vee \overline{B} \vert\!- \overline{A}$ **1.26** $A \to C, B \to D, \overline{C} \vee \overline{D} \vert\!- \overline{A} \vee \overline{B}$

Доказать, используя алгоритм метода резолюций:

1.27 $A \to (B \to C), B \vert\!- (A \to C)$
1.28 $A \to (B \vee C), (B \vee C) \to C \vert\!- (A \to C)$
1.29 $(A \vee B) \to C \vert\!- (B \to C)$
1.30 $A, A \wedge B \to (A \to (C \vee \overline{B})) \vert\!- B \to (C \vee \overline{B})$
1.31 $A \vee B, A \vee B \to \left(A \to C \vee B\right) \vert\!- C \vee B$
1.32 $B \to (\overline{A} \wedge C) \vert\!- \left(A \wedge B\right) \to C$
1.33 $A \to B, C \to D, B \to E, D \to F, E \to \overline{F}, A \to C \vert\!- \overline{\mathrm{A}}$
1.34 $\left(\mathrm{A} \to \mathrm{C}\right) \wedge \left(\mathrm{B} \to \mathrm{C}\right), \mathrm{C} \to \left(\mathrm{A} \wedge \mathrm{D}\right) \vert\!- \left(A \vee B\right) \to \mathrm{A} \wedge \mathrm{D}$

§ 3. Аксиомы и основные теоремы. Правила доказательства и теорема дедукции

1. Задание формальной теории. Логичность рассуждений можно доказывать, используя *формальные аксиоматические теории* (ФАТ). Любая ФАТ определена, если задан алфавит, определено понятие формулы над алфавитом, заданы аксиомы и правила вывода данной теории. Примером такой теории может служить ФАТ с системой аксиом I-XI, правилами подстановки и отделения (m.p.) (см. Приложение II).

2. Определение вывода формулы в ФАТ. Для произвольной ФАТ ключевым является определение вывода формулы.
Формула D *выводится из списка гипотез* $P_1, P_2, \ldots, P_n$ *по правилам вывода* ФАТ, если существует доказательство (список вывода): $D_1, D_2, \ldots, D_m$, составленное по следующим правилам:

1) $D_m = D$.
2) Любая $D_i, i \le m$ является или одной из гипотез $P_1, P_2, \ldots, P_n$, или аксиомой ФАТ, или получена из предыдущих формул в списке вывода по правилам вывода данной ФАТ.

Формула Ф называется *теоремой* ФАТ, если она выводится из аксиом по правилам вывода данной ФАТ. Список основных теорем ФАТ с системой аксиом 1-11 приведён в Приложении II.

Пример 1. Доказать логичность рассуждения в исчислении высказываний:

$A, A \& B \to (A \to (C \vee B)) \mid - B \to (C \vee B)$.

◀ По теореме дедукции (см. Приложение II) , достаточно доказать

$A, A \& B \to (A \to (C \vee B)), B \mid -(C \vee B)$.

2. Доказательство:

$\overset{1}{A}, \overset{2}{B}, \overset{3}{A \wedge B}, \overset{4}{A \wedge B \to (A \to (C \vee B))}, \overset{5}{A \to (C \vee B)}, \overset{6}{C \vee B}$.

Здесь

1- формула из списка гипотез слева от |–.

2- формула из списка гипотез слева от |–.

3- получена из 1 и 2 по правилу с конъюнкцией (Приложение II).

4- формула из списка гипотез слева от |–.

5- получена из 3 и 4 по правилу m.p.

6- получена из 1 и 5 по правилу m.p.

Другое оформление доказательства:

1. $A, B \mid - A \wedge B$ по правилу с конъюнкцией (см. Приложение II),

2. $A \wedge B, A \wedge B \to (A \to (C \vee B)) \mid - A \to (C \vee B)$ – по правилу m.p.

3. $A \to (C \vee B), A \mid -(C \vee B)$ – по правилу m.p.

Пример 2. Доказать теорему в исчислении высказываний.

$\mid - A \vee B \to B \vee A$

◀ Решение данного вида задач сводится к нахождению наиболее подходящей теоремы или аксиомы и дальнейшему преобразованию теоремы или аксиомы к доказываемой формуле с использованием правил вывода данной теории. Основным критерием выбора аксиомы (или теоремы) является сходство её второй части (или второй части во второй части и т.д.) с доказываемой теоремой. Так, для решения данной задачи выбираем аксиому VIII $: \mid -(A \to C) \to ((B \to C) \to ((A \vee B) \to C))$, так как в ней присут-

ствует формула $(A \vee B) \to C$, схожая по структуре с доказываемой теоремой $|-A \vee B \to B \vee A$.
Используем замену С:=` $B \vee A$, тогда по правилу подстановки

$|-(A \to B \vee A) \to ((B \to B \vee A) \to (A \vee B \to B \vee A)))$.

В данном выражении первая скобка $(A \to B \vee A)$ является аксиомой VII, поэтому можно применить правило m.p., которое означает, что если в некоторой теореме $|-A \to B$ известно, что А – теорема (или аксиома), то тогда В-теорема (т.е. $|-B$). После отделения $(A \to B \vee A)$ получим

$|-(B \to B \vee A) \to (A \vee B \to B \vee A)$.

В последней теореме первая скобка $B \to (B \vee A)$ является аксиомой VI, поэтому согласно правилу m.p., получаем требуемое доказательство теоремы:

$|-A \vee B \to B \vee A$

Доказать логичность рассуждения, используя определение вывода формулы в исчислении высказываний:

1.35 $A \to (B \vee C), (B \vee C) \to \overline{\overline{C}}, \overline{\overline{A}} |- C$

1.36 $\overline{\overline{A}} \to (B \wedge C), (B \vee C) \to \overline{\overline{C}}, A |- C$

1.37 $(A \vee B) \to \overline{\overline{C}}, \overline{\overline{B}} |- C$

1.38 $(A \vee B) \to C, A |- \overline{\overline{C}}$

1.39 $A \to (\overline{\overline{B}} \to (A \wedge \overline{D})), B, A |- (A \wedge \overline{D})$

1.40 $A \to (B \to (\overline{A} \vee B)), \overline{\overline{B}}, A |- (\overline{A} \vee B)$

Доказать логичность рассуждения в исчислении высказываний, используя теорему дедукции и другие теоремы теории:

1.41 $A \to (B \vee C), |- ((B \vee C) \to \overline{\overline{C}}) \to (A \wedge B \to C)$

1.42 $A \to (B \to C), B |- (A \to C)$

1.43 $|- ((A \vee B) \to \overline{\overline{C}}) \to (B \wedge D \to C)$

1.44 $|-((A \vee B) \to C) \to (A \wedge B \to \overline{\overline{C}})$

1.45 $A \to (\overline{\overline{B}} \to (A \wedge \overline{D})) |- (A \wedge B \to (\overline{\overline{A}} \wedge \overline{D}))$

1.46 $\overline{\overline{A}} \wedge \overline{\overline{B}}, A \wedge B \to \left(A \to C \vee B\right) |- C \vee B$

1.47 $B \to (\overline{A} \wedge C) |- \left(A \wedge \overline{\overline{B}}\right) \to C$

1.48 $A, A \wedge B \to (A \to (C \vee \overline{B})) \vdash \overline{\overline{B}} \to (C \vee \overline{B})$

1.49 $\vdash D \to (D \vee C) \wedge \overline{\overline{D}}$

1.50 $\vdash B \to (D \to B) \wedge \overline{\overline{B}}$

1.51 $(\mathrm{A} \to \mathrm{C}) \wedge (\mathrm{B} \to \mathrm{C}), \mathrm{C} \to \left(\overline{\overline{\mathrm{A}}} \wedge \mathrm{D}\right) \vdash (A \vee B) \to \overline{\overline{\mathrm{A}}} \wedge \mathrm{D}$

1.52 $\text{П} \to \text{Б}, \Phi \to В \vdash (\overline{Б} \vee \overline{В}) \to (\overline{\text{П}} \vee \overline{\Phi})$

Доказать теоремы в исчислении высказываний:

1.53 $\vdash B \to (A \wedge B \to B)$
1.54 $\vdash A \to ((A \vee B) \to A)$
1.55 $\vdash (B \to A) \to ((A \vee B) \to A)$
1.56 $\vdash A \to (\overline{B} \to A \wedge \overline{B})$
1.57 $\vdash A \to (B \to A) \wedge (B \to A)$
1.58 $\vdash A \to (A \vee B) \wedge (B \to A)$
1.59 $\vdash (A \to B) \to (A \to A \wedge B)$
1.60 $\vdash (B \to A) \to (B \to A \wedge B)$
1.61 $\vdash A \to (B \to B)$
1.62 $\vdash B \to (A \to A)$
1.63 $\vdash ((A \wedge B) \vee B) \to B$
1.64 $\vdash ((A \wedge \overline{\overline{C}}) \vee C) \to \overline{\overline{C}}$
1.65 $\vdash (A \wedge \overline{B}) \to (\overline{B} \wedge A)$
1.66 $\vdash (\overline{A} \vee B) \to (B \vee \overline{A})$
1.67 $\vdash (A \to (B \to A)) \vee (B \to A)$
1.68 $\vdash A \to (A \vee C) \wedge \overline{\overline{A}}$
1.69 $\vdash A \to (B \to B \wedge A)$
1.70 $\vdash (C \to \overline{\overline{C}}) \vee (B \to A)$
1.71 $\vdash (A \wedge B) \to (A \vee B)$
1.72 $\vdash (A \wedge B) \to (B \to A)$
1.73 $\vdash (A \to B) \to ((A \vee B) \to B)$
1.74 $\vdash B \to (A \to A)$
1.75 $\vdash (A \to B) \to (A \to B \wedge A)$
1.76 $\vdash (A \wedge B) \to (B \vee A)$
1.77 $\vdash \overline{B} \to ((A \to B) \to \overline{A})$
1.78 $\vdash (A \to \overline{\overline{B}}) \to (A \to (B \wedge A))$

1.79 $\vdash A \to (B \to B \wedge \overline{\overline{A}})$

1.80 $\vdash (A \to B) \to (A \to B \wedge (B \to A))$

1.81 $\vdash (A \to B) \wedge \overline{B} \to \overline{A}$

1.82 $\vdash (A \to (B \to C)) \wedge (A \to B) \to (A \to C)$

§ 4. Логика предикатов

1. Операции над предикатами. Функция n-переменных $P(x_1, x_2, \ldots, x_n)$ называется *предикатом*, если все переменные $x_i, i = 1, 2, \ldots, n$ принимают значения из некоторых *предметных* множеств (сами переменные также называются *предметными)*, а функция $P(x_1, x_2, \ldots, x_n)$ на каждом наборе значений предметных переменных принимает значение из множества $\{И, Л\}$. Число переменных n называется *местностью* предиката.

К предикатам применимы все операции логики высказываний, которые сохраняют тот же смысл, который был им присвоен в логике высказываний, а также операции связывания кванторами *всеобщности* ($\forall$) и *существования* ($\exists$).

Пусть $P(x)$ – одноместный предикат, где $x \in M$, тогда $(\forall x)P(x)$ – новый 0-местный предикат, который принимает значение истина (И), если $P(x) = И$ для *любого* x из предметного множества М .

Пусть $P(x)$ – одноместный предикат, где $x \in M$, тогда $(\exists x)P(x)$ – новый 0-местный предикат, который принимает значение истина (И), если *существует* хотя бы один элемент x_0 из предметного множества М, такой что $P(x_0) = И$.

В одноместном предикате $P(x)$, где $x \in M$, переменная x называется *свободной*, а в новых 0-местных предикатах $(\forall x)P(x)$, $(\exists x)P(x)$ переменная x называется *связанной*. Данное определение обобщается на случай произвольного числа предметных переменных, например, в формуле $\Phi = ((\exists x P(x, y) \to \forall x Q(y, x)) \to \overline{R}(y))$ переменная x связанная, а y – свободная.

2. **Интерпретация формулы.** *Интерпретация* формулы логики предикатов задана, если для каждой предметной переменной формулы задано конкретное предметное множество, и каждому предикатному символу поставлен в соответствие конкретный предикат.

Две формулы логики предикатов $\Phi_1(x_1, x_2, \ldots, x_n), \Phi_2(x_1, x_2, \ldots, x_n)$ со свободными переменными $x_1, x_2, \ldots, x_n$ *равносильны* в заданной интерпретации, если на каждом наборе значений переменных $x_1, x_2, \ldots, x_n$ значения формул совпадают ($\Phi_1(x_1, x_2, \ldots, x_n) = \Phi_2(x_1, x_2, \ldots, x_n)$).

Пример 1. Проверить являются ли формулы $Ф_1 = P(x_1,x_2) \vee P(x_3,x_2)$, $Ф_2 = P(x_1,x_2) \vee P(x_3,x_1)$ равносильными в заданной интерпретации. Предметное множество для переменных: M={1,2};

Таблица интерпретаций:

X	Y	P
1	1	Л
1	2	И
2	1	Л
2	2	И

- Таблица значений формулы $Ф_1$

x_1	x_2	x_3	$P(x_1,x_2)$	$P(x_3,x_2)$	$Ф_1$
1	1	1	Л	Л	Л
1	1	2	Л	Л	Л
1	2	1	И	И	И
1	2	2	И	И	И
2	1	1	Л	Л	Л
2	1	2	Л	Л	Л
2	2	1	И	И	И
2	2	2	И	И	И

Таблица значений формулы $Ф_2$

x_1	x_2	x_3	$P(x_1,x_2)$	$P(x_3,x_1)$	$Ф_2$
1	1	1	Л	Л	Л
1	1	2	Л	Л	Л
1	2	1	И	Л	И
1	2	2	И	Л	И

2	1	1	Л	И	И
2	1	2	Л	И	И
2	2	1	И	И	И
2	2	2	И	И	И

Так как таблицы значений формул не совпадают, то формулы не равносильны в заданной интерпретации.

1.83 Определить значения высказываний: $P(2), P(3), \forall x P(x), \exists x P(x),$ где $P(x)$ – означает «x - четное число», $x \in \{2,3\}$

1.84 Определить значения высказываний: $P(2), P(4), \forall x P(x), \exists x P(x),$ где $P(x)$ – означает «x – простое число», $x \in \{2,4\}$

1.85 Пусть задана интерпретация предиката
$Q(x,y)$: "$x \le y$", $y \in \{2,4\}, x \in \{2,3\}$.
Построить таблицы значений предикатов
$Q(x,y), \forall x Q(x,y), \forall y Q(x,y), \exists x Q(x,y), \exists y Q(x,y)$

1.86 Задана интерпретация предиката $Q(x,y)$: "$x \ge y$", $y \in \{2,4\}, x \in \{2,3\}$.
Построить таблицы значений предикатов
$Q(x,y), \forall x Q(x,y), \forall y Q(x,y), \exists x Q(x,y), \exists y Q(x,y)$

1.87 Проверить эквивалентность высказываний:
$\forall x \exists y Q(x,y) \sim \forall y \exists x Q(x,y); \forall x \forall y Q(x,y) \sim \forall y \forall x Q(x,y); \exists x \exists y Q(x,y) \sim \exists y \exists x Q(x,y);$
$\exists x \forall y Q(x,y) \sim \forall y \exists x Q(x,y),$

где $Q(x,y)$ *предикат примера* 1.85.

1.88 Проверить истинность высказываний:
$\forall x \exists y Q(x,y) \sim \forall y \exists x Q(x,y); \forall x \forall y Q(x,y) \to \forall y \forall x Q(x,y); \exists x \exists y Q(x,y) \sim \exists y \exists x Q(x,y);$
$\exists x \forall y Q(x,y) \to \forall y \exists x Q(x,y),$

где $Q(x,y)$ *предикат примера* 1.86

1.89 Проверить равносильность формул:
$\Phi_1 = P(x_1,x_2) \vee P(x_2,x_3)$ *и* $\Phi_2 = P(x_1,x_2) \vee P(x_1,x_3)$ в интерпретациях 1 и 2.
Интерпретация 1: предикат P интерпретируется P_1;
Интерпретация 2: предикат P интерпретируется P_2,
где P_1 и P_2 задаются таблицей:

x	y	P_1	P_2
1	1	И	Л
1	2	И	Л
2	1	Л	Л
2	2	Л	И

$x_1, x_2, x_3 \in M = \{1,2\}$

1.90 Проверить равносильность формул
$\Phi_1 = P(x_1,x_2) \to P(x_3,x_2)$ *и* $\Phi_2 = P(x_2,x_1) \sim P(x_1,x_3)$ в интерпретациях 1 и 2
Интерпретация 1: предикат P интерпретируется P_1;
Интерпретация 2: предикат P интерпретируется P_2,
где P_1 и P_2 задаются таблицей:

x	y	P_1	P_2
1	1	И	Л
1	2	И	Л
2	1	Л	Л
2	2	Л	И

3. Формы представления формул логики предикатов. Булева формула логики предикатов находится в *приведённой* форме, если знаки отрицания стоят только перед символами предикатов. Приведённая формула находится в *нормальной* форме, если она или не содержит кванторов, или все кванторы находятся впереди формулы.

Пример 2. Привести формулу $\Phi = \overline{\forall x P(x) \sim \exists y \overline{Q}(y)}$ к приведенной и нормальной форме.

◀1. Запишем формулу в булевой форме, используя законы логики высказываний (Приложение I):

$$\Phi = \overline{\forall x P(x) \sim \exists y \overline{Q}(y)} = \overline{(\forall x P(x) \vee \overline{\exists y \overline{Q}(y)}) \wedge (\overline{\forall x P(x)} \vee \exists y \overline{Q}(y))}$$

2. Спустим отрицания до символов предикатов, используя правила де Моргана, двойного отрицания (Приложение I) и законы вынесения кванторов из под отрицания (Приложение III).

$$\Phi = (\overline{\forall x P(x)} \wedge \exists y \overline{Q}(y)) \vee (\forall x P(x) \wedge \overline{\exists y \overline{Q}(y)}) =$$
$$= (\exists x \overline{P(x)} \wedge \exists y \overline{Q}(y)) \vee (\forall x P(x) \wedge \forall y Q(y))$$

Данная формула является приведенной.

3) Приведём к нормальной форме, сначала используя законы вынесение кванторов за скобки:

$$\Phi = \exists x \exists y (\overline{P}(x) \wedge \overline{Q}(y)) \vee \forall x \forall y (P(x) \wedge Q(y))$$

Теперь применим законы переименования связанных переменных, т.е. в любой части дизъюнкции (например, во второй) переименовываем связанные переменные $x, y : x := t,\ y := z$

$$\Phi = \exists x \exists y (\overline{P}(x) \wedge \overline{Q}(y)) \vee \forall t \forall z (P(t) \wedge Q(z)).$$

Вновь применяя законы вынесения кванторов, получаем нормальную форму формулы

$$\Phi: \quad \Phi = \exists x \exists y \forall t \forall z (\overline{P}(x) \wedge \overline{Q}(y) \vee P(t) \wedge Q(z)).$$

Используя законы логики предикатов найти равносильную приведенную и нормальную форму для Φ*:*

1.91 $\Phi = (\exists x P(x) \to \forall y Q(y)) \to \exists y \overline{Q}(y)$;
1.92 $\Phi = \overline{\exists x (P(x) \to \forall y Q(x, y))}$
1.93 $\Phi = \forall x P(x) \sim \exists y \overline{Q}(y)$
1.94 $\Phi = \left(\exists x P(x) \sim \forall y \overline{Q}(y)\right) \to \left(\forall z R(z)\right)$

Глава 2. ТЕОРИЯ АЛГОРИТМОВ

§ 1. Суперпозиция. Оператор примитивной рекурсии.

1. Частичные числовые функции. Функция $f(x_1, x_2, \ldots, x_n)$ называется *частичной числовой*, если все её аргументы $x_i, i = 1,2,\ldots n$ принимают значения из некоторых подмножеств $M_i \subset \overline{N} = \{0,1,2\ldots\}, i = 1,2,\ldots n$, и для каждого набора значений аргументов $(x_1, x_2, \ldots, x_n)$ значение функции принадлежит $\overline{N} = \{0,1,2\ldots\}$. Если область определения функции $f(x_1, x_2, \ldots, x_n)$ совпадает с множеством $\overline{N}^n = \overline{N} \times \overline{N} \times \ldots \times \overline{N}$, то говорят, что данная функция *всюду определенная*, в противном случае – *частично определенная*.

Пример 1.

◀Функция f(x,y) = x + y- всюду определенная двуместная функция,

функция f(x,y) = x – y - частично определенная, т.к. она определена для

x $\le$ **y**.

2. Элементарные рекурсивные функции. К элементарным рекурсивным функция относятся:

1) Константа 0:0(x) = 0.

2) Функция сдвига на 1: S (x) = x + 1.

3) Тождественная функция(или введения фиктивных переменных):

$$I_m^n(x_1, \ldots, x_m, \ldots, x_n) = x_m \qquad (m \le n).$$

Используя элементарные функции и три оператора: *суперпозиции, примитивной рекурсии и минимизации*, можно построить новые частичные числовые функции.

3. Оператор суперпозиции. Обозначение оператора: $S_m^n(\varphi, f_1, f_2, ..., f_m)$ Частичная числовая функция *n*-переменных $f(x_1, x_2 ... x_n)$ является результатом применения оператора суперпозиции к частичной числовой функции *m*-переменных $\varphi = \varphi(t_1, t_2 ... t_m)$ и частичным числовым функциям $f_i = f_i(x_1, x_2, ..., x_n), i = 1,2,\ldots,m$, если

$$f(x_1, x_2 ... x_n) = \varphi(f_1(x_1, x_2 ... x_n), f_2(x_1, x_2 ... x_n), ..., f_m(x_1, x_2 ... x_n)).$$

Таким образом, результатом суперпозиции является подстановка в функцию от m переменных m функций от n одних и тех же переменных.

Благодаря тождественным функциямтребование, что все $f_i = f_i(x_1, x_2, ..., x_n), i = 1, 2, \ldots, m$ зависят от одинакового числа переменных, не уменьшает ее возможностей. Например,
$f(x_1, x_2) = h(g_1(x_1, x_2), g_2(x_1))$
в стандартном виде можно записать так
$f(x_1, x_2) = S_2^2(h(x_1, x_2), g_1(x_1, x_2), g_2(I_1^2(x_1, x_2)))$.

Используя суперпозицию и тождественные функции, можно переставлять и отождествлять аргументы в функциях:

$$f(x_2, x_1, x_3) = f(I_2^2(x_1, x_2), I_1^2(x_1, x_2), x_3);$$

$$f(x_1, x_1, x_3) = f(x_1, I_1^2(x_1, x_2), x_3).$$

Суперпозицией из элементарных функций можно получать новые функции. Покажем это на следующих примерах.

Пример 2.

1. Пусть$h(x) = S(x)$, $g(x) = 0(x)$. Тогда $f(x) = S_1^1(h(x), g(x)) =$
$= S(0(x)) = S(0(x)) = 1(x) = 1$ (константа 1).

2. Пусть $h(x) = S(x)$, $g(x) = S(x)$. Тогда $f(x) = S_1^1(h(x), g(x)) =$
$= S(S(x)) = S(x+1) = x + 2$ (функция сдвига на 2).

3. Пусть $h(x,y) = I_1^2(x,y)$, $g_1(x) = S(x)$, $g_2(x) = 0(x)$.

Тогда$f(x) = S_2^1(h(x, y), g_1(x), g_2(x)) = I_1^2(S(x), 0(x)) = S(x) = x + 1$.

4. Найти результат суперпозиции:
$f(x, y) = S_3^2(\left|xy - z^2\right|; S(xy), I_2(x, yS(x)), S_1^2(x^3; xy)$

◀ Найдём результат по действиям:

1) $S(xy) = xy + 1$ (по определению функции сдвига значение аргумента увеличивается на единицу).
2) $yS(x) = y(x+1) = xy + y$,

3) $S_1^2(x^3;xy)=x^3y^3$ (здесь x^3– внешняя функция, поэтому подставляем вместо х произведение ху и возводим в куб).

$$f(x,y)=S_3^2(\left|xy-z^2\right|;xy+1,xy+y,x^3y^3)=\left|(xy+1)(xy+y)-(x^3y^3)^2\right|=$$
$$=\left|x^2y^2+xy+y+xy^2-x^6y^6\right|$$

(здесь $|xy-z^2|$ – внешняя функция, остальные – внутренние. По определению оператора суперпозиции вместо переменной x внешней функции подставляем первую внутреннюю функцию, вместо у подставляем вторую, вместо z – третью. Затем раскрываем скобки, приводим подобные слагаемые и записываем ответ).

5. Найти результат суперпозиции:

$$f(x,y)=S_3^2[x^{y(z^2+4)};I_2(x,S(y)),\frac{x}{y^2},S_2^2(I_2(x,y);xy,x+y)].$$

◀ Найдём результат по действиям:

1) $I_2(x,S(y))=S(y)=y+1$ (т.к. у функции проектирования нижний индекс равен двум, то выбираем вторую переменную – функцию сдвига).

2) $S_2^2(I_2(x,y);xy,x+y))=I_2(xy,x+y)=x+y$ (сначала применяем оператор суперпозиции, потом вычисляем функцию проектирования).

3) $S_3^2[x^{y(z^2+4)};y+1,\frac{x}{y^2},x+y]=(y+1)^{\frac{x}{y^2}((x+y)^2+4)}$ (подставляем полученные в предыдущих действиях функции в основной оператор суперпозиции; заменяем переменные во внешней функции $x^{y(z^2+4)}$. Вместо х подставляем первую внутреннюю функцию, вместо у подставляем вторую, вместо z – третью).

Найти суперпозицию функций:

2.1 $S_1^1[S(x),x^2]$

2.2 $S_1^1\{S(x),S_1^1[S(x),O(x)]\}$

2.3 $S_1^1\{S(x),S_1^1[S(x),S(x)]\}$

2.4 $S_2^1[I_2^2(x,y),x,S(y)]$

2.5 $S_2^1\{I_1^2(x,y),S_1^1[S(x),2(x)],x\}$

2.6 $S_1^3\left(S\left(x^2\right),S_2^3\left(x+y,x^3yz,x^{2yz}\right)\right)$

2.7 $S_2^3\left(x^2y^3+x,S\left(S\left(xyz+1\right)\right),I_2^3\left(x,z^{xy},y\right)\right)$

2.8 $S_3^1\left(\left|x-2y+4z\right|,S_1^1\left(2x,S\left(S\left(x\right)\right)\right),x^4,I_1\left(S\left(0\left(x\right)\right)+2x\right)\right)$

4. Оператор примитивной рекурсии. Следующей операцией, позволяющей получать новые функции из элементарных функций, является операция примитивной рекурсии.

Рекурсивным определением функции называется такое ее определение, при котором значение функции для данных аргументов определяется значениями той же функции для предыдущих значений аргументов.

Примером рекурсивного определения являются числа Фибоначчи. Они определяются рекурсивно по двум предыдущим значениям, т.е.

$$f(n+2) = f(n) + f(n+1), \quad \text{причем } f(0) = 1,\ f(1) = 1.$$

Из этого определения получается последовательность чисел: { 1, 1, 2, 3, 5, 8, … }.

Рассмотрим определение оператора примитивной рекурсии для функции одной переменной. Говорят, что частичная числовая функция $f(x)$ является результатом применения оператора $Пр(\varphi,\psi(x,y))$ примитивной рекурсии к частичным числовым функциям φ (константе) и $\psi(x,y)$, т.е. $f(x) = Пр(\varphi,\psi(x,y))$, если f(x) удовлетворяет *схеме примитивной рекурсии*

$$Пр: \quad \begin{cases} f(0)=\varphi \\ f(x+1)=\psi(x,f(x)) \end{cases}$$

Пример 3. Составить схему примитивной рекурсии для $f(x) = Пр(1,x+3y)$ и вычислить $f(4)$.

◄Записываем схему примитивной рекурсии для функции одной переменной. Заметим, что $f(x)$ подставляется вместо переменной y в заданную функцию двух переменных $\psi(x,y)$ в операторе $Пр(\varphi,\psi(x,y))$:

$$\begin{cases} f(0) = 1, \\ f(x+1) = x + 3f(x). \end{cases}$$

В нашем случае

$$\begin{cases} f(0) = 1 \\ f(0+1) = f(1) = 0 + 3f(0) = 3 \\ f(1+1) = f(2) = 1 + 3f(1) = 10 \\ f(2+1) = f(3) = 2 + 3f(2) = 32 \\ f(3+1) = f(4) = 3 + 3f(3) = 99 \end{cases}$$

Рассмотрим определение оператора примитивной рекурсии для функции двух переменных.

Говорят, что частичная числовая функция $f(x,y)$ является результатом применения оператора примитивной рекурсии $Пр(\varphi(x),\psi(x,y,z))$ к частичным числовым функциям $\varphi(x)$ и $\psi(x,y,z)$, т.е. $f(x,y)=Пр(\varphi(x),\psi(x,y,z))$, если f(x) удовлетворяет *схеме примитивной рекурсии*

$$Пр: \quad \begin{cases} f(x,0)=\varphi(x) \\ f(x,y+1)=\psi(x,y,f(x,y)) \end{cases}$$

Пример 4. Составить схему примитивной рекурсии для $f(x,y)=Пр(x+1,2y+xz)$ и вычислить $f(1,3)$.

◄Запишем схему примитивной рекурсии для функции двух переменных. Заметим что функция f(x,y) подставляется вместо переменной z функции $\psi(x,y,z)$:

$$\begin{cases} f(x,0)=\varphi(x)=x+1 \\ f(x,y+1)=\psi(x,y,f(x,y))=2y+xf(x,y) \end{cases}$$

Для нашего случая

$$\begin{cases} f(1,0)=1+1=2 \\ f(1,0+1)=f(1,1)=0+1\cdot f(1,0)=2 \\ f(1,1+1)=f(1,2)=2+1\cdot f(1,1)=4 \\ f(1,2+1)=f(1,3)=4+1\cdot f(1,2)=8 \end{cases}$$

Для общего случая функции n переменных определение оператора примитивной рекурсии формулируется следующим образом. Говорят, что частичная числовая функция $f(x_1,x_2,\ldots,x_n)$ является результатом применения оператора примитивной рекурсии $Пр(\varphi(x_1,x_2,\ldots,x_{n-1}),\psi(x_1,x_2,\ldots,x_{n+1}))$ к частичным числовым функциям $\varphi(x_1,x_2,\ldots,x_{n-1})$ и $\psi(x_1,x_2,\ldots,x_{n+1})$, т.е. $f(x_1,x_2,\ldots,x_n)=Пр(\varphi(x_1,x_2,\ldots,x_{n-1}),\psi(x_1,x_2,\ldots,x_{n+1}))$, если $f(x_1,x_2,\ldots,x_n)$ удовлетворяет *схеме примитивной рекурсии*

$$Пр: \quad \begin{cases} f(x_1,x_2,\ldots,x_{n-1},0)=\varphi(x_1,x_2,\ldots,x_{n-1}) \\ f(x_1,x_2,\ldots,x_n+1)=\psi(x_1,x_2,\ldots,x_n,f(x_1,x_2,\ldots,x_n)) \end{cases}$$

Пример 5. С помощью операции примитивной рекурсии построить функцию f (x, y) по функциям g (x) = x и h (x, y, z) = x + z.

◄Для построения искомой функции применим схему (3). Тогда получим следующие выражения:

f (x, 0) = x,

f (x, 1) = x + f (x, 0) = x + x = 2 x,

f (x, 2) = x + f (x, 1) = x + 2 x = 3 x, и т.д.

Отсюда следует гипотеза, что искомая функция будет иметь вид: f (x, y) = x (y + 1). Данную гипотезу проверяем, составив новую схему рекурсии для найденной функции f (x, y) = x (y + 1). Действительно, f(x, 0) = x, и f (x, y+1) = x (y + 2)= x (y + 1)+x= f (x, y)+x, следовательно,
$f(x,y) = Пр(x, x+z)$.

2.9 Найти *f(7),* если $f(x) = Пр(1, \psi(x,y))$,
где $\psi(x,y) = x + y$.

2.10 Найти *f(6)* если $f(x) = Пр(2, \psi(x,y))$,
где $\psi(x,y) = x + 2y$.

Найти $f(x) = Пр(\varphi, \psi(x,y))$ если:

2.11 $\varphi = 0,\ \psi(x,y) = y+2$

2.12 $\varphi = 2, \psi(x,y) = y+1$

2.13 $\varphi = 0,\ \psi(x,y) = 2x+y+1$

2.14 $\varphi = 1, \psi(x,y) = x+y$

2.15 Найти *f(2,4),* если $f(x,y) = Пр(\varphi(x), \psi(x,y,z))$,
где $\varphi = x\ , \psi(x,y,z) = (x+1)z$

2.16 Найти *f(2,5),* если $f(x,y) = Пр(\varphi(x), \psi(x,y,z))$,
где $\varphi = 1\ , \psi(x,y,z) = x + 2z$

Применяя операцию примитивной рекурсии к функциям $\phi(x)$ *и* $\psi(x,y,z)$ *по переменной* y, *построить функцию* $f(x,y) = Пр(\phi, \psi)$, *записав ее в «аналитической» форме:*

2.17 $\varphi(x) = x\ ,\ \psi(x,y,z) = x + z$

2.18 $\varphi\ (x) = 2^x, \psi(x,y,z) = 2^x z$

2.19 $\varphi\ (x) = 3x,\ \psi(x,y,z) = 2y + z$

2.20 $\varphi(x) = x, \psi(x,y,z) = y + z$

2.21 $\phi(x) = x, \quad \psi(x,y,z) = x + z$

2.22 $\phi(x) = x + 1, \quad \psi(x,y,z) = 2x + z$

2.23 $\phi(x) = 2^x, \quad \psi(x,y,z) = 2^x \cdot z$

2.24 $\phi(x) = 3^x, \quad \psi(x,y,z) = 3^y \cdot z$

2.25 $\phi(x) = 1, \quad \psi(x,y,z) = x \dot{-} y$

2.26 $\varphi(x) = 2, \quad \psi(x,y,z) = \begin{cases} y, \textit{если } x \ge y, \\ 0, \textit{если } x < y \end{cases}$

2.27 $\phi(x) = sg\ x, \quad \psi(x,y,z) = x \cdot sg\ y + z \cdot \overline{sg}\ x$.

Найти $f(x) = Пр(\varphi, \psi(x,y))$ *если:*

2.28 $\phi(x)=1, \quad \psi(x,y)=\dfrac{1}{(3x+4)(3x+1)}+y$

2.29 $\phi(x)=1, \quad \psi(x,y)=\dfrac{1}{(4x+5)(4x+1)}+y$

§ 2. Примитивно-рекурсивные и частично-рекурсивные функции.

1. Примитивно-рекурсивные функции. Функция $f(x_1,x_2,\ldots,x_n)$ называется *примитивно-рекурсивной*, если она может быть получена из трёх элементарных функций с помощью конечного числа применений операторов суперпозиции и примитивной рекурсии.

Этому определению можно придать более формальный индуктивный вид.

1. Функции 0 (x), S(x), I^n_m для всех натуральных m, n, где $m \le n$, являются примитивно- рекурсивными.

2. Если $f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)$, $\varphi(x_1, \ldots, x_m)$ – примитивно-рекурсивные функции, то $f(x_1,x_2...x_n)=S^n_m(\varphi,f_1,f_2,...,f_m)$ – примитивно рекурсивная функция для любых натуральных n, m.

3. Если $\varphi(x_1,x_2,\ldots,x_{n-1})$ и $\psi(x_1,x_2,\ldots,x_{n+1})$ примитивно- рекурсивные функции, то $f(x_1,x_2,\ldots,x_n)=\textit{Пр}(\varphi(x_1,x_2,\ldots,x_{n-1}),\psi(x_1,x_2,\ldots,x_{n+1}))$– примитивно-рекурсивная функция.

4. Других примитивно-рекурсивных функций нет.

Пример 1.

1. Докажем, что функция f(x) = x + n, где n=2,3,… - примитивно-рекурсивная.

◄Представим функцию x + 2 следующим образом:

$$x + 2 = (x + 1) + 1 = S(x) + 1 = S (S(x)).$$

Тогда: $x + 3 = (x + 2) + 1 = S (S(x)) + 1 = S (S (S(x))).$

Из этих соотношений следует, что функция x + 2 получается из элементарной функции S(x) применением к ней один раз операции суперпозиции, функция x + 3 - применением к S(x) два раза операции суперпозиции. Следовательно, функция x + n получится из S(x) применением к ней (n – 1) раз операции суперпозиции, т.е.

$$f(x) = x + n = S(S(\ldots S(x)\ldots)),$$

а это и означает, что заданная функция примитивно-рекурсивная.

2. Докажем, что функция $f(x, y) = x + y$ - примитивно-рекурсивная.

◀Для заданной двухместной функции можно найти схему примитивной рекурсии. Для этого надо найти выражения для $f(x, 0)$ и $f(x, y + 1)$.

Это будут следующие выражения $f(x, 0) = x$, $f(x, y + 1) = f(x, y) + 1$. Следовательно, схема примитивной рекурсии для заданной функции будет иметь вид

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

$$f(x, y + 1) = f(x, y) + 1.$$

$$\varphi(x) = x, \psi(x, y, f(x, y)) = f(x, y) + 1.$$

Запишем функцию ψ в следующем виде $\psi(x, y, z) = z + 1$. Полученные функции φ и ψ, можно получить из элементарныхфункций с помощью суперпозиции следующим образом:

$\varphi(x) = I_1^1(x)$, $\psi(x, y, z) = S_1^3(S(x), I_3^3(x, y, z))$, следовательно, эти функции являются примитивно-рекурсивными. Тогда заданную функцию можно записать так: **x + y** $=$ $\Pi p(x, z+1)$, т.е. она получается по схеме примитивной рекурсии из функций $\varphi(x) = I_1^1(x)$, $\psi(x, y, z) = S_1^3(S(x), I_3^3(x, y, z))$. А так как эти функции примитивно-рекурсивные, то по определению, и заданная функция будет примитивно-рекурсивной.

3. Покажем, что функция $f(x, y) = xy$примитивно-рекурсивная.

◀Схема примитивной рекурсии для нее будет следующая

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

$$f(x, y + 1) = f(x, y) + x.$$

Следовательно, функции φ и ψ будут иметь вид:

$\varphi(x) = 0$, $\psi(x, y, z) = x + z$. Функция $\boldsymbol{\varphi(x)}$ этоэлементарная функция, а функцию ψ можно представить так:

$\psi(x, y, z) = S_2^3(x + y, I_1^3(x, y, z), I_3^3(x, y, z))$, следовательно, функция ψ получается из двуместной примитивно рекурсивной функции **x + y** (см. задачу 2) и других примитивно-рекурсивных функций с помощью операции суперпозиции, поэтому она так же будет примитивно-рекурсивной.

4. Покажем, что возведение в степень f(x, y) = x^y примитивно-рекурсивная функция.

◀Схема примитивной рекурсии для этой функции будет следующая

$$f(x, 0) = 1$$
$$f(x, y+1) = xf(x, y).$$

Функции φ и ψ имеют вид: $\varphi(x) = 1 = S(O(x))$, $\psi(x, y, z) = xz$. Отсюда видно, что $\varphi(x)$ примитивно-рекурсивная функция. Докажем, что и функция ψ примитивно-рекурсивная. Ее можно представить так: $\psi(x, y, z) = S_2^3(xy, I_1^3(x,y,z), I_3^3(x,y,z))$. В предыдущем примере 3 было доказано, что функция xy примитивно-рекурсивная, следовательно, функция ψ получается суперпозицией из примитивно-рекурсивных функций. Это означает, что ψ – примитивно-рекурсивная функция.

5. Покажем, что функция f(x) = x ! – примитивно-рекурсивная.

◀Схема примитивной рекурсии для нее следующая:

$$f(0) = 1,$$
$$f(x+1) = (x+1)\,f(x).$$

Следовательно,

$$\varphi(x) = 1 = S(0), \text{ и } \psi(x, y) = (x+1)y = S_2^2(xy, S(I_1^2(x,y)), I_2^2(x,y)).$$

Для упрощения доказательства в некоторых случаях удобно использовать следующее утверждение, доказываемое в лекционном курсе: *любая частичная числовая линейная функция* n*переменных* $f(x_1, x_2, \ldots, x_n) = Ax_1 + Bx_2 + \ldots + Cx_n + D$, *где* $A, B, \ldots, C, D \in \overline{N} = \{0,1,2,\ldots\}$ *– является примитивно-рекурсивной.*

Пример 2.

Доказать, что функция является примитивно-рекурсивной (при решении использовать, что любая линейная функция является примитивно-рекурсивной): $f(x,y) = 9(3x+2y+1)^2 + 5$

◀Запишем общий вид схемы примитивной рекурсии для функции двух переменных $\begin{cases} f(x,0) = \varphi(x) \\ f(x,y+1) = \Psi(x,y,f(x,y)) \end{cases}$

Для нашего случая

$$\begin{cases} f(x,0)=9(3x+1)^2+5, \\ f(x,y+1)=9(3x+2y+2+1)^2+5=9((3x+2y+1)+2)^2+5=9((3x+2y+1)^2+4(3x+2y+1)+4)+ \\ \qquad +5=9(3x+2y+1)^2+36(3x+2y+1)+36+5=(9(3x+2y+1)^2+5)+108x+72y+36+ \\ \qquad +36=f(x,y)+108x+72y+72 \end{cases}$$

Таким образом, $\varphi(x)=9(3x+1)^2+5$, $\Psi(x,y,z)=z+108x+72y+72$ – линейная функция, следовательно, является примитивно рекурсивной.

Осталось доказать, что
$\varphi(x)=9(3x+1)^2+5$ -- примитивно-рекурсивна. Обозначим данную функцию другой буквой, например $g(x)=9(3x+1)^2+5$

Записываем схему примитивной рекурсии для функции $g(x)=9(3x+1)^2+5$ одной переменной.
В нашем случае

$$\begin{cases} g(0)=9+5=14 \\ g(x+1)=9(3x+3+1)^2+5=9((3x+1)+3)^2+5=9((3x+1)^2+6(3x+1)+9)+5= \\ \qquad =9(3x+1)^2+5+18x+15=g(x)+18x+15. \end{cases}$$

Следовательно, у нас $\varphi=14$, $\psi(x,y)=y+18x+15$. Обе функции являются линейными, следовательно, они примитивно рекурсивны. Следовательно $f(x)=9(3x+1)^2+5$ – примитивно рекурсивная функция.

Окончательно, для первоначальной функции $f(x,y)=9(3x+2y+1)^2+5$ доказано, что она примитивно-рекурсивна.

Пример 3. Доказать, что функция является примитивно-рекурсивной (при решении использовать, что любая линейная функция является примитивно-рекурсивной):

$$f(x)=3^{5x+2y+1}.$$

◀Запишем схему примитивной рекурсии для двух переменных:

$$\begin{cases} f(x,0)=3^{5x+1} \\ f(x,y+1)=3^{5x+2(y+1)+1}=3^{5x+2y+1}\cdot 3^2=f(x,y)\cdot 9 \end{cases}$$

Таким образом, $\varphi(x)=3^{5x+1}$, $\psi(x,y,z)=9z$ – линейная функция, которая является примитивно рекурсивной. Осталось доказать, что $\varphi(x)=3^{5x+1}$ –

примитивно-рекурсивна. Обозначим данную функцию другой буквой, например $g(x)=3^{5x+1}$

Запишем схему примитивной рекурсии к функции одной переменной:

$$\begin{cases} g(0)=3, \\ g(x+1)=3^{5(x+1)+1}=3^{5x+1}\cdot 3^5=g(x)\cdot 3^5. \end{cases}$$

Тогда $\varphi=3$, $\psi(x,y)=3^5 y$ – примитивно рекурсивные функции (как линейные). Следовательно $g(x)=3^{5x+1}$ – примитивно рекурсивная функция. Окончательно, первоначальная функция $f(x)=3^{5x+2y+1}$ является примитивно-рекурсивной.

Отметим, однако, что не все известные в математике функции могут быть постоены с помощью двух операторов – суперпозиции и примитивной рекурсии. Такие функции, как разность $f=x-y$, или частное $f=\dfrac{x}{y}$ могут быть постоены только с помощью третьего оператора – аналога обратной функции в математическом анализе.

2. Оператор минимизации. Пусть задана некоторая частичная числовая функция $f(x_1,x_2,\ldots,x_n)$.

Оператор *минимизации* по переменной $\mathbf{x}_n$ определяет *новую* функцию $g(x_1, \ldots, x_n)$ следующим образом:

Пусть $a=(a_1, \ldots, a_n)$ – произвольный набор целых неотрицательных чисел. Запишем следующее уравнение:

$$f(a_1, \ldots, a_{n-1}, u) = a_n \qquad (1)$$

1. если уравнение (1) *имеет решение* $u_0 \in \overline{N}$ и при всех $u \in \overline{N}$ таких, что $u<u_0$ функция $f(a_1, \ldots, a_{n-1}, u)$ *определена* и ее значение $f(a_1, \ldots, a_{n-1}, u) \neq a_n$, тогда $g(a)=u_0$;

2. если уравнение (1) *не имеет решения* в целых неотрицательных числах, то $g(a)$ не определено;

3. если u_0 – наименьшее целое неотрицательное решение уравнения (1) и *при некотором* $u \in \overline{N}$, $u<u_0$ значение $f(a_1, \ldots, a_{n-1}, u)$ *не определено*, тогда значение $g(a)$ *не определено*.

Оператор минимизации обозначается $g(x_1, \ldots, x_n) = \mu_{x_n}(f(x_1, \ldots, x_n))$.

Пример 4. Применить операцию минимизации к функции одной переменной:

$$f(x) = 2x + 1.$$

◀Уравнение (1) для заданной функции имеет вид:

$$2y + 1 = a_1 \quad (2)$$

Решение этого уравнения имеет вид:

$$y_0 = (a_1 - 1) / 2. \quad (3)$$

Для того, чтобы решение (3) принимало значения целых неотрицательных чисел нужно, чтобы число a_1 принимало только следующие значения: $a_1 \notin \in \{ 1, 3, 5, \dots \}$, т.е. все нечетные числа.

Определим с помощью операции минимизации значение функции $g(x)$ для некоторого значения a_1, например для $a_1 = 9$.

Решение уравнения (2) при $a_1 = 9$ существует и равно согласно (3) $y_0 = 4$.

При всех значениях нечетных чисел меньших чем $y_0 = 4$, т.е. для 1 и 3 функция $f(y)$ определена и ее значения при $y = 1$ и $y = 3$ равны соответственно 3 и 7, т.е. не равны $a_1 = 9$.

Тогда согласно первому пункту определения операции минимизации значение функции $g(9)$ будет равно решению уравнения (3) при $a_1 = 9$, т.е.

$$g(9) = 4.$$

Нетрудно проверить, что для любого другого нечетного значения a_1 будет выполняться первый пункт определения операции минимизации, т.е.

$$g(a_1) = y_0$$

Следовательно, функция полученная операцией минимизации из функции $f(x) = 2x + 1$ имеет вид:

$$g(x) = (x - 1) / 2,$$

т.е. является обратной функцией к функции $f(x)$. Областью определения ее являются все положительные нечетные числа.

Пример 5. Применить операцию минимизации к функции:

$$f(x) = \begin{cases} 3x + 2 \text{...при....} x \neq 2 \\ \text{не .определено..при..} x = 2. \end{cases}$$

◀Уравнение (1) для этой функции имеет вид:

$$3y + 2 = a_1 \qquad (4)$$

Так как число a_1 может принимать только целочисленные значения, которые определяются значениями функции

$$f(y) = 3y + 2 \quad (y \neq 2),$$

то это будет следующее множество:

$$a_1 \in \{ 2, 5, 11, 17, \dots \}.$$

Найдем значения функции $g(x)$ с помощью операции минимизации для этих значений аргумента.

Решением уравнения (4) является:

$$y_0 = (a_1 - 2) / 3. \qquad (5)$$

При $a_1 = 2$ из (5) получим: $y_0 = 0$. Так как нет целочисленных неотрицательных значений $y < y_0 = 0$, то согласно первому пункту определения операции минимизации

$$g(2) = 0.$$

При $a_1 = 5$, решение уравнения (4) будет $y_0 = 1$. Меньше этого значения будет только $y = 0$. Для этого значения функция $f(x)$ определена, причем $f(0) = 3 \neq a_1 = 5$. Следовательно, получим

$$g(5) = 1.$$

При $a_1 = 11$, $y_0 = 3$. Т.к. при $y = 2 < y_0$ значение функции $f(x)$ не определено, то согласно пункту 3 определения операции минимизации и значение функции $g(x)$ не определено.

Для остальных значений a_1 получим результат аналогичный результату при $a_1 = 11$, т.е. функция $g(x)$ определена только для двух значений аргумента: $x = 2$ и $x = 5$.

Оператор минимизации является удобным средством для построения обратных функций. Поэтому в применении к одноместным функциям она иногда называется операцией обращения.

Подведем некоторые итоги. Из элементарных функций с помощью операций суперпозиции и примитивной рекурсии можно получить огромное разнообразие функций, включающих основные функции арифметики, алгебры и анализа (с поправкой на целочисленность). Тем самым выяснено, что эти функции примитивно-рекурсивные, что означает существование определенной процедуры их вычисления; следовательно, их естественно отнести к классу вычислимых функций.

Все примитивно-рекурсивные функции всюду определены. Это следует из того, что элементарные функции всюду определены, а операции суперпозиции и примитивной рекурсии это свойство сохраняют. Функции, построенные с помощью минимизации, могут быть не всюду определенными (частично числовыми).

3. Частично-рекурсивные функции. Функция называется *частично – рекурсивной*, если она может быть получена из элементарных функций с помощью конечного числа применений операторов суперпозиции, примитивной рекурсии и минимизации. Всюду определенные частично-рекурсивные функции называются *общерекурсивными*.

Доказать, что f(x) и f(x, y) – примитивно-рекурсивные функции:

2.30 $f(x,y)=3x+5y+6$

2.31 $f(x)=x^2$

2.32 $f(x)=3x^2+2$

2.33 $f(x,y)=7x^2+2y+4$

2.34 $f(x,y)=2x^2y$

2.35 $f(x,y)=x^3y$

2.36 $f(x,y)=3(x+3y+2)^2+4$

2.37 $f(x,y)=2(3x+2y+1)^2+1$

2.38 $f(x,y)=2\cdot 7^{3x+4y+1}$

2.39 $f(x,y)=4\cdot 5^{4x+2y+2}$

Доказать примитивную рекурсивность следующих функций, используя простейшие функции и функции $sg\,x,\ \overline{sg}\,x,\ x+y,\ x\dot{-}y,\ x\cdot y,\ x\oplus y$:

2.40 $|x-y|$

2.41 $\min(x,y)$

2.42 $\max(x,y)$

2.43 $x^2\cdot y^2\oplus z^3$

2.44 $\left[\dfrac{x}{y}\right]=\begin{cases} x, \text{ если } y=0, \\ \text{целая часть от деления } x \text{ на } y, \text{ если } y\ge 1; \end{cases}$

2.45 a^x, *где* $a\ge 2$ *и натуральное*

2.46 $f_{a,i}(x)=\begin{cases} a, \text{ если } x=i, \\ x \text{ в ином случае,} \end{cases}$ a,i – *какие - либо числа из* N;

2.47 $f(x)=\begin{cases} a_i, \text{ если } x=i,\ i=0,1,\dots,m, \\ c, \text{ в ином случае,} \end{cases}$

здесь $a_0,a_1,\dots,a_m,c$ – *какие - либо натуральные числа*

2.48 $f(x)=\begin{cases}0 \text{ при } x \text{ четном,}\\ x \text{ при } x \text{ нечетном;}\end{cases}$

2.49 $f_m(x)=\begin{cases}1, \text{ если } x=l\cdot m, l=0,1,2,\dots, m\geq 2 \text{ и натуральное,}\\ 0 \text{ в ином случае;}\end{cases}$

2.50 $\left[\sqrt{x}\right]$.

Применить операцию μ к f(x):

2.51 $f(x)=3x+2$

2.52 $f(x)=2x+3$

2.53 $f(x)=\begin{cases}2x+3,\ x\neq 3\\ \text{неопределена при } x=3\end{cases}$

2.54 $f(x)=\begin{cases}3x+4,\ x\neq 3\\ \text{неопределена при } x=3\end{cases}$

2.55 $f(x)=|x-4|$

2.56 $f(x)=2|x-3|$

2.57 $f(x)=\begin{cases}|2x-3|,\ x\neq 3\\ \text{неопределена при } x=3\end{cases}$

2.58 $f(x)=\begin{cases}|x-5|,\ x\neq 6\\ \text{неопределена при } x=3\end{cases}$

2.59 $f(x)=|x-2|+|x-3|$

2.60 $f(x)=|x-5|+|x-4|$

2.61 $f(x)=||x-3|-4|$

2.62 $f(x)=||x-2|-3|$

2.63 $f(x)=5$

2.64 $f(x)=2$

2.65 $f(x)=x\dot{-}2$

2.66 $f(x)=x\dot{-}4$

2.67 $f(x)=x-2$

2.68 $f(x)=2x-1$

2.69 $f(x)=2x\dot{-}1$

2.70 $f(x)=3x\dot{-}2$

2.71 $f(x)=\left[\frac{x}{2}\right]$

2.72 $f(x) = \left[\frac{x}{3}\right]$

2.73 $f(x) = \frac{x}{2}$

2.74 $f(x) = \frac{x}{3}$

2.75 $f(x) = \left[\frac{x}{2}\right] - \left[\frac{x}{3}\right]$

2.76 $f(x) = \left[\frac{x}{3}\right] - \left[\frac{x}{4}\right]$

Найти примитивно рекурсивную функцию (если она существует), из которой однократным применением операции минимизации можно получить частично рекурсивную функцию f*:*

2.77 $f(x_1) = 2 - x_1$;

2.78 $f(x_1) = \frac{x_1}{2}$;

2.79 $f(x_1) = \frac{1}{x_1 + 1}$;

2.80 $f(x_1) = sg\ (x_1 - 1)$;

2.81 $f(x_1, x_2) = x_1 - 2x_2$;

2.82 $f(x_1, x_2) = \frac{x_1 - x_2}{3}$;

2.83 $f(x_1, x_2) = \frac{x_1}{x_2 + 2}$;

2.84 $f(x_1, x_2) = \frac{x_1}{1 - x_1 \cdot x_2}$.

§ 3. Работа машин Тьюринга

Под машиной Тьюринга (машины Т) понимается некоторое *абстрактное* устройство состоящее из следующих частей:

1. *информационная бесконечная лента*, разделенная на ячейки.

В каждой ячейке можно записать только один символ $a_j \in A$ из *внешнего алфавита* A машины Т. Среди символов внешнего алфавита особую роль играет *пустой* символ a_0. Ячейка называется *пустой*, если в ней записан пустой символ. Наиболее часто используется внешний алфавит $A = \{0,1\}$ с пустым символом 0. Функцией информационной ленты является *хранение информации*;

2. *считывающая-записывающая головка*, которая может располагаться напротив любой ячейки и по специальной команде *читать* записанный в этой ячейке символ и *записывать* в нее новый символ;

3. *управляющее устройство*, которое в каждый рассматриваемый момент времени находится в определенном состоянии $q_i \in Q$, где Q – *множество внутренних состояний*, или *внутренний алавит* машиты Т. В данном множестве выделены подмножества *начальных* $Q_1 \subset Q$ и *конечных* $Q_0 \subset Q$ состояний машины Т. Если данные подмножества состоят из одного элемента, то q_1 – начальное состояние (в нём машина начинает работу), а q_0 – конечное, или заключительное состояние (в нём машина заканчивает работу).Управляющее устройство выполняет *три функции*: *переходов, выходов и движения*. В зависимости от состояния $q_i \in Q$ в момент времени ti от символа $a_j \in A$, записанного в просматриваемой ячейке в момент времени t, оно дает команду *на переход* в новое внутреннее состояние $q_l \in Q$, *запись* в эту ячейку нового символа $a_k \in A$ и команду на *движение* считывающей-записывающей головки на одну ячейку. Команда R означает передвижению головки на ячейку вправо, L – передвижение влево, а S – нет передвижения головки.

Замечание. В дальнейшем информационную ленту будем называть просто лентой, считывающую-записывающую головку будем называть *головкой машины*, а вместо – состояние управляющего устройства будем говорить – *состояние машины*.

Машинным словом называется запись символов внешнего алфавита на ленте, с указание первого и последнего *непустого* символа например:

1 0 1 1 00 1.

Конфигурацией на ленте в момент времени t называется машинное слово с указанием расположения головки и состояния машины.

Например, следующая запись означает некоторую конфигурацию машины:

$$\begin{array}{l} 1\ 0\ 1\ 0\ 1\ 1 \\ \uparrow \\ q_2 \end{array}$$

Данная конфигурация означает, что машина находится в состоянии q_2, головка машины расположена напротив ячейки в которой записан 0.

В дальнейшем для записи конфигурации будем применять более компактную запись следующего вида:

$$1\ 0\ 1\ q_2 0\ 1\ 1.$$

Данная конфигурация соответствует вышеприведенной конфигурации.

Замечание. Считается, что во всех ячейках, справа и слева от ячеек где записано машинное слово, записаны пустые символы алфавита (в нашем случае, нули).

Машинной командой называется пятёрка символов: $q_i\ a_j\ q_l\ a_k\ K$, где смысл всех символов описан выше (см. функции управляющего устройства машины Т). Например, запись вида

$$q_2 1\ q_4 0\ R,$$

означает, что если машина находится в состоянии q_2, а головка расположена против ячейки, в которой записана 1, то управляющее устройство дает команду на переход машины в состояние q_4, записать вместо 1 символ 0, а головке – передвинуться на ячейку вправо.

Программой некоторой машины Тьюринга называется совокупность всех машинных команд, которые может выполнить эта машина.

Машина Тьюринга считается заданной, если задан *триплет* $T = \langle A, Q, \Pi \rangle$, где П – программа машины Т.

Работа машины Тьюринга заключается в следующем. Машина работает *дискретно* (пошагово).

1. Для начала работы нужно задать *начальную*, или входную конфигурацию K_1 с *начальным* (входным) словом P_1

2. На каждом шаге происходит последовательный переход от одной *текущей конфигурации*конфигурации к другой. Пусть K_r - конфигурация на ленте в момент времени t, тогда по заданной паре q_i (состояние машины в момент t) и a_j (обозреваемый символ в момент t), машина продолжит работу по команде в программе П, начинающейся с пары $q_i\ a_j$. Результатом применения команды является новая конфигурация K_{r+1}. 3.Работа машины заканчивается когда машина перейдет в одно из своих заключительных состояний, или когда нет команды, начинающейся с пары $q_i\ a_j$. Возникшая при этом конфигурация K_0 называется *заключительной*, а заключи-

тельное машинное слово P_0 - *результатом* применения машины Тьюринга к исходному слову P_1.

Говорят, что машина Тьюринга *применима* к данному слову P_1, если она заканчивает работу, и *не применима* к P_1, если она никогда не приходит в заключительное состояние.

Пример1.

Пусть машина Тьюринга задана следующей программой:

$$q_1\ 0\quad q_0\ 1\ S\ ,$$

$$q_1\ 1\quad q_1\ 1\ R\ .$$

Рассмотрим работу заданной машины по переработке исходного слова

$$1\ 1\ 0\ 1.$$

Пусть на ленте машины задана следующая начальная конфигурация: $q_1$1101. Данная конфигурация означает, что машина находится в состоянии $\mathbf{q}_1$, головка расположена против ячейки в которой записана **1**.

Тогда, выполняя вторую команду программы, машина должна остаться в этом же состоянии, в рассматриваемой ячейке должна быть записана 1 и головка передвинется вправо.

В результате этого получим на ленте машины новуюконфигурацию: 1 $\mathbf{q}_1\mathbf{1}$ 01. Выполняя снова вторую команду программы, от этой конфигурации приходим к следующей: 11 $\mathbf{q}_1\mathbf{0}$ 1. К данной конфигурации применима первая команда программы. Выполняя ее получим следующую конфигурацию: 11 $q_0$11. Полученная конфигурация является заключительной, т.к. машина достигла заключительного состояния.

В этом примере можно сказать, что заданная машина Тьюринга применима к слову 1101, а результатом работы машины над данным словом является слово 1111.

Программа данной машины может быть задана в виде следующей таблицы:

Таблица 1

T	0	1
q_1	q_0 1 S	$q_1 1R$

Пример 2.

Проверить применимость машины Тьюринга, заданной табл. 2, к слову 11001 при следующих начальных конфигурациях:

1. а) $1\,q_1 1\,0\,0\,1$ и в) $1\,1\,q_1\,0\,0\,1$.

Таблица 2

T	0	1
q_1	$q_1 0 L$	$q_0 0\,S$

◂Рассмотрим работу машины для представленных начальных конфигураций.

а) $1\ \mathbf{q}_1\mathbf{1}\ 001$

$1\ \mathbf{q}_0\mathbf{0}\ 001$

в) $1\,1\,\mathbf{q}_1\mathbf{0}\ 0\,1$

$1\,\mathbf{q}_1\mathbf{1}\ 0\,0\,1$

$1\,\mathbf{q}_0\mathbf{0}\ 0\,0\,1$

Из работы машины видно, что она заменяет первую встреченную **1** на **0** и затем останавливается.

Выяснить, применима ли машина с слову P, записать программу таблицей, если в начальной конфигурации головка обозревает самую левую единицу слова

Обозначения: $1^x = \underbrace{11\ldots1}_{x}$, $0^x = \underbrace{00\ldots0}_{x}$

2.85

$$\Pi:\ \begin{cases} q_1 0 q_1 0 R \\ q_1 1 q_2 0 R \\ q_2 1 q_1 0 R \\ q_2 0 q_0 1 S \end{cases}$$

а) $P = 1^3 01$ б) $1^2 0^2 1$ в) 1^6

2.86

$$\Pi:\ \begin{cases} q_1 0 q_2 1 L \\ q_1 1 q_2 1 R, \\ q_2 1 q_1 1 R \end{cases}$$

$а) P = 1^2 0^2 1,\ б) P = 1^6\ в) P = 1^2 0 1^3$

2.87

$$П: \begin{cases} q_1 0 q_2 0 R \\ q_1 1 q_2 1 L \\ q_2 0 q_3 1 R \\ q_2 1 q_3 0 R \\ q_3 1 q_1 1 R \end{cases}$$ а) $P = 1^3 0 1^2$ б) 1^5 в) $1^2[01]^2$

2.88

$$П: \begin{cases} q_1 0 q_1 1 R \\ q_1 1 q_2 0 R \\ q_2 0 q_1 1 R, \\ q_2 1 q_3 1 L \\ q_3 0 q_1 1 L \end{cases}$$ $а) P = [10]^2 1^2, б) P = 10^2 1^2 в) P = 10^3 1$

По заданной начальной конфигурации $К_1$ найти заключительную конфигурацию

2.89

$$П: \begin{cases} q_1 0 q_2 1 R \\ q_1 1 q_2 0 L \\ q_2 0 q_0 1 S \\ q_2 1 q_1 1 L \end{cases}$$ а) $К_1 = 1^2 0 1 q_1 1^2$б) $К_1 = 101 q_1 0 1^2$

2.90

$$П: \begin{cases} q_1 0 q_1 1 L \\ q_1 1 q_2 1 R \\ q_2 1 q_1 10 R \\ q_2 0 q_0 0 L \end{cases},$$ $а) K_1 = 1 q_1 0 1^3$ $б) K_1 = 1 q_1 1^4$

2.91 Создать программу машины, которая из начальной конфигурации 01…1q110…

а) сдвигает слово влево на 1 ячейку,

б) добавляет по 1 справа и слева от слова,

в) если число единиц четно, приписывает слева одну единицу, если нечетно, то две.

При этом в заключительном состоянии головка должна смотреть на крайнюю правую 1 слова.

2.92 Создать программу машины, которая из начальной конфигурации q11111…110

а) сдвигает слово вправо на 1 ячейку,

б) приписывает справа ноль и две единицы,

в) если число единиц четно, приписывает справа одну единицу, если нечетно, то никогда не останавливается

При этом в заключительном состоянии головка должна смотреть на крайнюю левую 1 слова.

2.93 Построить в алфавите $\{0,1\}$ машину Тьюринга, обладающую следующими свойствами (предполагается, что в начальный момент головка обозревает самый левый символ слова, и в качестве пустого символа берется 0):

а) машина имеет одно состояние, одну команду и применима к любому слову в алфавите $\{0,1\}$;

б) машина имеет две команды, не применима ни к какому слову в алфавите $\{0,1\}$ и зона работы на каждом слове бесконечная;

в) машина имеет две команды, не применима ни к какому слову в алфавите $\{0,1\}$ и зона работы на любом слове ограничена одним и тем же числом ячеек, не зависящим от выбранного слова;

г) машина имеет три команды, применима к словам $10^{2^n}1$ $(n \geq 1)$ и не применима к словам $10^{2n+1}1$ $(n \geq 0)$;

д) машина имеет пять команд, применима к словам 1^{3n} $(n \geq 1)$ и не применима к словам $1^{3n+\alpha}$ $(\alpha = 1{,}2$ и $n \geq 0)$;

е) машина применима к словам 1^n01^n, где $n \geq 1$, и не применима к словам 1^m01^n, где $m \neq n, m \geq 1$ и $n \geq 1$.

2.94 По словесному описанию машины Тьюринга построить ее программу (в алфавите $\{0,1\}$).

а) Начав работу с последней единицы массива из единицы, машина «сдвигает» его на одну ячейку влево, не изменяя «остального содержимого» ленты. Головка останавливается на первой единице «перенесенного» массива.

б) начав двигаться вправо от какой-то «начальной» ячейки, головка «находит» первую при таком перемещении ячейку с единицей (если такая ячейка «встретится на пути») и, сделав «один шаг» вправо, останавливается на соседней ячейке. Если в «начальной» ячейке записана единица, то головка останавливается на соседней справа ячейке. Содержимое ленты не меняется.

в) машина начинает работу с самой левой непустой ячейки и отыскивает единицу, примыкающую с левой стороны к первому слева массиву из трех нулей («окаймленному» единицами). Головка останавливается на найденной единице (если такая есть). Содержимое ленты не меняется.

г) При заданном $l \geq 1$ головка машины, начав работу с произвольной ячейки, содержащей единицу, двигается вправо до тех пор, пока не пройдет подряд $l+1$ нулей. Головка останавливается на первой ячейке за этими $l+1$ нулями, напечатав в ней 1. Остальной содержимое ленты не меняется.
д) При заданном $l \geq 1$ головка машины, начав работу с какой-то ячейки и двигаясь вправо, ставит подряд $2l$ единиц и останавливается на последней из них.
е) При заданном $l \geq 1$ головка машины, двигаясь вправо от какой-либо пустой ячейки, находит первый при таком перемещении массив, содержащий не менее l единиц, стирает в нем первые l единиц и останавливается на самой правой из ячеек, в которых были стертые единицы. Остальное содержимое ленты не меняется.

Построить в алфавите {0,1} машину Тьюринга, переводящую конфигурации K_1 в конфигурации K_0:

2.95 $K_1 = q_1 1^n, \quad K_0 = q_0 1^n 0 1^n$ $(n \geq 1)$;
1) $K_1 = q_1 0^n 1^n, \quad K_0 = q_0 [01^n]$ $(n \geq 1)$;
2.96
1) $K_1 = 1^n q_1 0, \quad K_0 = q_0 1^{2n}$ $(n \geq 1)$;
2) $K_1 = 1^n q_1 0 1^m, \quad K_0 = 1^m q_0 0 1^n$ $(m \geq 1, n \geq 1)$.

§ 4. Схемы машин Тьюринга

1. Произведение машин. Пусть машина Тьюринга T_1 имеет **n** внутренних состояний q_i^1 и заключительное состояние q_0^1, а машина T_2 - m состояний q_i^2 и заключительное q_0^2. Произведение этих машин обозначается $\mathrm{T} = \mathrm{T}_1 \cdot \mathrm{T}_2$

Произведением машин T_1 и T_2 называется машина T у которой внутренние состояния определяются правилом:
$\mathrm{q}_i = \mathrm{q}_i^1$ (i = 1, …, n), $\mathrm{q}_{n+j} = \mathrm{q}_j^2$ (j = 1, …, m), заключительное состояние машины T_1 заменяется на начальное состояние машины T_2, т.е. $\mathrm{q}_0^1 = \mathrm{q}_1^2 = \mathrm{q}_{n+1}$, а заключительное состояние T равно заключительному состоянию машины T_2.

Из этих соотношений следует, что работа машины T эквивалентна следующей последовательной работе машин T_1 и T_2. Начинает работу с начальной конфигурацией машина T_1. Достигнув своего заключительного состояния q_0^1 она заканчивает свою работу. Затем начинает работать машина T_2 со своего начального состояния q_1^2, причем головка машины T_2 в этот момент находится против той ячейки где закончила работать машина T_1. Машина T_2 заканчивает свою работу, достигнув своего заключительного состояния.

Таблица машины T получается из таблиц машин T_1 и T_2 путем их объединения. Следующий пример показывает, как это делается.

Пример 1.

Найти таблицу задания машины $T = T_1 \cdot T_2$, где машины T_1 и T_2 заданы табл.1 и 2.

Таблица 1

T_1	0	1
q_1^1	q_2^1 0 L	q_1^1 1 L
q_2^1	q_2^1 0 L	q_0^1 1 S

Таблица 2

T_2	0	1
q_1^2	q_0^2 0 S	q_1^2 0 L

◀Объединяя эти таблицы в одну и учитывая, что $q_0^2 = q_0$, а $q_0^1 = q_1^2$ получим таблицу машины T, в виде табл.3

Таблица 3

T	0	1
q_1^1	q_2^1 0 L	Q_1^1 1 L
q_2^1	q_2^1 0 L	q_1^2 1 S
q_1^2	q_0 0 S	q_1^2 0 L

Переобозначая состояния машины, получим окончательную таблицу машины T в виде табл.4.

Таблица 4

T	0	1
q_1	q_2 0 L	q_1 1 L
q_2	q_2 0 L	q_3 1 S
q_3	q_0 0 S	q_3 0 L

Операция умножения машин Тьюринга *некоммутативна*, т.е. $T_1 \cdot T_2 \neq T_2 \cdot T_1$.

Для данного примера таблица машины $T^* = T_2 \cdot T_1$ представлена табл.5

Таблица 5

T^*	0	1

q_1	q_2 0 S	q_1 0 L
q_2	q_3 0 L	q_2 1 L
q_3	q_3 0 L	q_0 1 S

Сравнивая табл.4 и табл.4 видим, что они не эквивалентны, следовательно: $T_1 \cdot T_2 \neq T_2 \cdot T_1$.

Можно показать, что операция умножения машин *ассоциативна*, т.е.

$$(T_1 \cdot T_2) \cdot T_3 = T_1 \cdot (T_2 \cdot T_3).$$

Через операцию произведения машин можно определить операцию возведения в степень машины T:

n *– ой степенью машины* T называется произведение $T^n = T^{n-1}T$.

Например, $T^2 = T \cdot T$, $T^3 = T \cdot T \cdot T$, и т. д.

В том случае, когда в записи произведения машин, машина стоящая слева имеет два или более заключительных состояния, необходимо указывать какое из них заменяется на начальное состояние машины, стоящей справа.

2. Разветвление машин. Пусть машина T_1 имеет по крайней мере два заключительных состояния q_0^i, q_0^j и n внутренних состояний q_i^1, машина T_2 – m состояний q_i^2, а машина T_3 – l состояний q_i^3. *Разветвление* машин T_2 и T_3 управляемым машиной T_1 обозначается

$$T = T_1 \begin{cases} (i)\, T_2 \\ (j)\, T_3 \end{cases}$$

внутренние состояния T определяются правилом:

$$q_i = \overset{1}{q_i}\ (i = 1, \ldots, n), q_{n+j} = q_j^2 \ \ (j = 1, \ldots, m),$$

$$q_{n+l+j} = \overset{3}{q_j}\ (j = 1, \ldots, l).$$

При этом, согласно введённому обозначению, заключительное состояние q_0^i машины T_1 заменяется на начальное состояние q_1^2 машины T_2, а заключительное состояние q_0^j машины T_1 заменяется на начальное состояние q_1^3 машины T_3. Машина T останавливается в заключительных состояниях машин T_2 и T_3 .

Пример 2. Следующая запись $\quad T = A \begin{cases} ..(1)..B \\ ..(2)..E \end{cases}$

означает, что машина T является разветвлением машин B и E, управляемым машиной A, причем машина B подключается к первому заключительному состоянию машины A, а машина E подключается ко второму заключительному состоянию машины A.

3. Итерация машиныT. *Итерацией* по паре состояний q_0^i и q_j называется машина T(q_0^i, q_j), программа которой получена из программы машины T, если в ней заменить все состояния q_0^i на состояние q_j.

Пример 3. Найти итерацию машины T, заданной табл. 6 по состояниям q_0^1 и q_2.

Таблица 6

T	0	1
q_1	q_1 1 L	q_2 1 L
q_2	q_0^1 0 R	q_0^2 1 R

◀Заменяя в табл.6 состояние q_0^1 на состояние q_2, получим табл.7, определяющую итерацию машины T по этим состояниям.

Таблица 7

	0	1
q_1	q_1 1 L	q_2 1 L
q_2	q_2 0 R	q_0 1 R

Замечание. Если машина T имеет только одно заключительное состояние, то после выполнения итерации получится машина не имеющая заключительного состояния.

4. Схемы машин Тьюринга. В схемах машин Тьюринга употребляются условные обозначения, введённые для произведения и разветвления машин (см. выше). Для итерации вводится следующее обозначение. Пусть имеется схема машин $T = \ldots T_1 \ldots T_2 \ldots$, в которой машина T_1 находится *левее*машины T_2 . Тогда обозначение $T = \ldots \dot{T}_1 \ldots \dot{T}_2 \ldots$означает итерацию машины T по паре состояний T(q_0^2, q_1^1) -заключительное состояние машины T_2заменяется на начальное состояние машины T_1, т.е. $q_0^2 = q_1^1$, при этом различное число точек над парами машин обозначает попарные итерации.

Пример 4. Рассмотрим схему машин:

$$T = \dot{A} \cdot \ddot{B} \begin{cases} ..(1)..\dot{C} \\ ..(2)..B.. \begin{cases} ..(1)..\ddot{D} \\ ..(2)..E \end{cases} \end{cases}$$

В данной схеме обозначены две итерации машин A с C и B с D.

Пример 5. Построить схему машины T:

$$T = \dot{A} \cdot \mathrm{B} \begin{cases} ..(1)..C \\ ..(2)..\dot{D} \end{cases}$$

Машины A, B, C и D задаются следующими таблицами:

A	0	1
q_1	q_0 0 R	q_1 1 R

B	0	1
$(q_2)q_1$	q_0^1 0 S	q_0^2 1 S

C	0	1
(q_3) q_1	q_3 0 L	q_1 1 L
(q_4) q_2	q_2 0 L	q_3 1 L
(q_5) q_3	q_0 0 L	q_3 1 L

D	0	1
(q_6) q_1	q_0 0 S	q_1 0 R

◀Согласно представленной схеме машин, заменяем состояние q_0 машины A на состояние q_1 машины B, состояние q_0^1 машины B на состояние q_1 машины C, состояние q_0^2 машины B на состояние q_1 машины D, состояние q_0 машины D на состояние q_1 машины A.

Объединяя последовательно все таблицы машин, и переобозначив внутренние состояния машин B, C и D, новыми состояниями, указанными в скобках, получим таблицу машины T:

T	0	1
q_1	q_2 0 R	q_1 1 R
q_2	q_3 0 S	q_6 1 S
q_3	q_4 0 L	q_3 1 L
q_4	q_4 0 L	q_5 1 L
q_5	q_0 0 L	q_5 1 L
q_6	q_1 0 S	q_6 0 R

Пример 6. Составить программу по схеме $T = A^2B$, применить Т к слову при заданной начальной конфигурации $K_1 = 11\,q_1\,111011101$.

A	0	1
q_1	$q_0 0R$	$q_1 1R$

B	0	1
q_1	$q_2 1R$	$q_1 1L$
q_2	$q_0 1L$	$q_2 1R$

C	0	1
q_1	$q_0^1 1L$	$q_0^2 0R$

◀Чтобы записать данную схему машин нужно последовательно подключать составляющие машины. Сначала пишем программу машины А. Затем к ней подключаем снова машину А, при этом конечное состояние первой машины становится переходом к начальному состоянию второй машины А. Здесь главное не забыть сместить номера состояния второй машины А, т.к. индексы состояний увеличиваются на 1. Далее конечное состояние второй машины А заменяем на начальное состояние машины В, все индексы состояний В сдвигаем на 2:

$$T = A^2B$$

T	0	1	
q_1	$q_2 0R$	$q_1 1R$	$\}A$
q_2	$q_3 0R$	$q_2 1R$	$\}A$
q_3	$q_4 1R$	$q_3 1L$	$\Big\}B$
q_4	$q_0 1L$	$q_4 1R$	

Теперь, пользуясь данной таблицей, применим машину Т к слову R., не забывая о том, что перед словом и после слова на ленте стоят 0 (по умолчанию).

$K_1 = 11q_1 111011101$ (смотрим на символ, что стоит *после* q_1 видим единицу, находим в таблице клетку на пересечении q_1 и 1; по команде мы остаёмся в состоянии q_1, оставляя единицу, и т. к. стоит символ движения R, то сдвигаемся вправо).

$K_2 = 111q_1 11011101$,

$K_3 = 1111q_1 1011101$,

$K_4 = 11111q_1 011101$

(смотрим на символ, что стоит *после* q_1. Видим ноль, находим в таблице клетку на пересечении q_1 и 0; по команде мы переходим в состояние q_2, пишем ноль и, т. к. стоит R, то сдвигаемся вправо),

$K_5 = 111110q_2 11101$ (смотрим на символ, что стоит *после* q_2. Видим единицу, находим в таблице клетку на пересечении q_2 и 1; по команде мы остаёмся в состоянии q_2, оставляя единицу, и т. к. стоит R, то сдвигаемся вправо),

$K_6 = 1111101q_2 1101$,

$K_7 = 11111011q_2 101$,

$K_8 = 111110111q_2 01$ (смотрим на символ, что стоит *после* q_2. Видим ноль, находим в таблице клетку на пересечении q_2 и 0; по команде мы остаёмся в состоянии q_3, оставляя ноль, и т. к. стоит R, то сдвигаемся вправо),

$K_9 = 1111101110q_3 1$ (смотрим на символ, что стоит *после* q_3. Видим единицу, находим в таблице клетку на пересечении q_3 и 1; по команде мы остаёмся в состоянии q_3, оставляя единицу, и т. к. стоит L, то сдвигаемся влево),

$K_{10} = 111110111q_3 01$ (смотрим на символ, что стоит *после* q_3. Видим ноль, находим в таблице клетку на пересечении q_3 и 0; по команде мы переходим в состояние q_4, пишем единичку, и т. к. стоит R, то сдвигаемся вправо),

$K_{11} = 1111101111q_4 1$ (смотрим на символ, что стоит *после* q_4. Видим единицу, находим в таблице клетку на пересечении q_4 и 1; по комнде мы остаёмся в состоянии q_4, оставляя единицу, и т. к. стоит R, то сдвигаемся вправо. Здесь заканчивается слово на ленте, как уже говорилось, за словом идут 0),

$K_{12} = 11111011111q_4 0$ (смотрим на символ, что стоит *после* q_4. Видим ноль, находим в таблице клетку на пересечении q_4 и 0; по команде мы переходим в конечное состояние q_0 , записывая единицу, и т. к. стоит L, то сдвигаемся влево),

$K_0 = 1111101111\, q_0\, 11$.

Машина закончила свою работу.

Пример 5.Составить программу по схеме машин:

$$T = \dot{B}\ddot{C}\begin{cases}(2)\dot{A}^2\\ (1)C\begin{cases}(1)A\\ (2)\ddot{B}\end{cases}\end{cases}$$

◀Запишем программу произведения машин B и C, подключая к конечному состоянию машины B начальное состояние машины C и сдвигая номера состояний машины C. Затем к первому конечному состоянию машины C подключаем машину A, сдвигая номера её состояний на 4. Потом к конечному состоянию машины A подключаем вторую машина A, также сдвигая номера её состояний. Конечное состояние второй машины A помечаем цифрой 1. Теперь ко второму конечному состоянию машины C снова подключаем машину C при этом номер начального состоянии второй машины C будет равным последнему состоянию, сдвинутому на единицу, т.е. 6. Затем к первому конечному состоянию машины C подключаем машину A и её конечное состоянии помечаем цифрой 2. После этого подключаем ко второму конечному состоянию машины C машину B и её конечное состояние помечаем цифрой 3.

Теперь делаем итерацию машин. Конечное состояние q_0^1 замыкаем на первое состояние первой машины B, поскольку над этими машинами стоит по одной точке. Следующая пара C и B. Конечное состояние q_0^3 замыкаем на начальное состояние машины C. Итерации закончены.

Ответ:

T	0	1	
q_1	$q_2 1R$	$q_1 1L$	B
q_2	$q_3 1L$	$q_2 1R$	
q_3	$q_6 1L$	$q_4 0R$	C
q_4	$q_5 0R$	$q_4 1R$	A
q_5	$q_0^1 0R$	$q_5 1R$	A
q_6	$q_7 1L$	$q_8 0R$	C
q_7	$q_0^2 0R$	$q_7 1R$	A
q_8	$q_9 1R$	$q_8 1L$	B
q_9	$q_0^3 1L$	$q_9 1R$	

2.97 Найти: T_1^3, T_1T_2, T_2T_1 и применить к слову
а) $1^3 0^2 1^2$ б) $1^4 01$

$$T_1: \begin{cases} q_1 0 q_2 0 R \\ q_1 1 q_2 1 R \\ q_2 0 q_0 1 L \\ q_2 1 q_1 0 R \end{cases} \qquad T_2: \begin{cases} q_1 0 q_2 1 R \\ q_1 1 q_1 0 L \\ q_2 0 q_1 1 R \\ q_2 1 q_0 1 S \end{cases}$$

2.98 Найти: T_1^3, T_1T_2, T_2T_1 и применить к слову
а) $q_1 0^2 1^2$ б) $q_1 0^4 1$

$$T_1: \begin{cases} q_1 0 q_2 1 R \\ q_1 1 q_2 0 R \\ q_2 1 q_0 1 L \\ q_2 0 q_1 1 R \end{cases} \qquad T_2: \begin{cases} q_1 0 q_2 0 R \\ q_1 1 q_1 1 L \\ q_2 0 q_1 0 R \\ q_2 1 q_0 1 S \end{cases}$$

Записать таблицу машины T:
2.99

а) $T = B^3$ б) $T = \dot{A}\begin{cases} (1)\ B^2 \\ (2)\dot{C} \end{cases}$, в) $T = \dot{B}\,\ddot{A}\begin{cases} (1)\ \dot{C} \\ (2)A\begin{cases} (1)\ \ddot{C} \\ (2)B \end{cases} \end{cases}$

2.100

а) $T = C^2B^2$ б) $T = C\,\dot{A}\begin{cases} (1)\ C^3 \\ (2)\dot{B} \end{cases}$, в) $T = \dot{C}^2\ddot{A}\begin{cases} (1)\ \ddot{B} \\ (2)BA\begin{cases} (1)\ B \\ (2)\dot{C} \end{cases} \end{cases}$

где

A	0	1
q_1	q_1 1 R	$q_2$0 L
q_2	q_0^1 1 S	q_0^2 0 R

B	0	1
q_1	q_2 1 R	$q_2$0 L
q_2	q_0 1 S	q_1 0 R

C	0	1
q_1	q_1 1 R	q_0 1 S

§ 5. Вычисление функций на машинах Тьюринга

1.**Кодирование чисел и наборов чисел.** Вычисления на машинах Тьюринга будем производить только с числами из множества $\overline{N}$. *Кодом* *числа* $\alpha \in \overline{N}$ назовём машинное слово $k(\alpha) = 1^{\alpha+1} = \underbrace{11\ldots1}_{\alpha+1}$. Например, кодом нуля будет одна единица, кодом единицы – две и т.д. *Кодомнабора чисел* $\tilde{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \overline{N} \times \overline{N} \times \ldots \overline{N} = \overline{N}^n$ назовём машинное слово

$$k(\alpha_1, \alpha_2, \ldots, \alpha_n) = k(\alpha_1)0k(\alpha_2)0\ldots0k(\alpha_n) = 1^{\alpha_1+1}01^{\alpha_2+1}\ldots1^{\alpha_n} =$$
$$= \underbrace{11\ldots1}_{\alpha_1+1}0\underbrace{11\ldots1}_{\alpha_2+1}0\ldots\underbrace{011\ldots1}_{\alpha_n+1}$$

Например, кодом набора (2,0,3) является машинное слово $k(2,0,3) = k(2)0k(0)0k(3) = 1110101111$.

2.**Вычислимость по Тьрингу.** Пусть задана частичная числовая функция $f(x_1, x_2, \ldots, x_n)$. Тогда функция $f(x_1, x_2, \ldots, x_n)$ называется *вычислимой* по Тьюрингу, если существует машина, которая вычисляет функцию следующим образом:

а) *начальная конфигурация*задана следующим образом:
$K_1 = q_1 k(\alpha_1, \alpha_2, \ldots, \alpha_n)$, где $k(\alpha_1, \alpha_2, \ldots, \alpha_n)$ - код произвольного набора $\tilde{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \overline{N} \times \overline{N} \times \ldots \overline{N} = \overline{N}^n$ значений переменных запись x_1, …, x_n.

б) если функция $f(x_1, x_2, \ldots, x_n)$*определена* на наборе $\tilde{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, то машина останавливается в заключительной конфигурации $K_0 = q_0 k(f(\alpha_1, \alpha_2, \ldots, \alpha_n))$, где$k(f(\alpha_1, \alpha_2, \ldots, \alpha_n))$ -- код значения функции $f(x_1, x_2, \ldots, x_n)$ на наборе $\tilde{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$.

в) если функция $f(x_1, x_2, \ldots, x_n)$*неопределена* на наборе $\tilde{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, то машина зацикливается.

Пример 1. Составить таблицу машины Тьюринга, вычисляющую константу 0: 0(x) = 0.

◀Для этой машины *начальная* конфигурация имеет вид: 0 q_1 1 … 1 1 0, а *заключительная*: 0 q_0 1 0.

Из этого следует, что *головка* машины должна передвигаться *вправо* и при этом *заменять* записанные в этих ячейках 1 на 0. Этот процесс будет осуществляться машинной командой: q_1 1 q_1 0 R.

Дойдя до ячейки, где записан 0, она должна *остановиться* напротив этой ячейки, *записать* в нее 1, а машина при этом должна перейти в состояние q_0. Это соответствует следующей машинной команде: q_1 0 q_0 1 S.

Таким образом, программа искомой машины будет состоять из этих двух машинных команд, а таблица машины будет иметь вид:

	0	1
q_1	q_0 1 S	q_1 0 R

Пример 2. Составить таблицу машины Тьюринга, вычисляющую функцию сдвига на 1: $S(x) = x + 1$.

◄При $x = 2$ начальная и заключительная конфигурации этой машины имеют вид:

$$0\, q_1 1\,1\,1\,0 \qquad\qquad 0\, q_0 1\,1\,1\,1\,0.$$

Процесс перехода от начальной конфигурации к заключительной можно осуществить следующим образом. Передвинуть головку машины на одну ячейку влево, затем записать 1 в предыдущей ячейке, не передвигая головки, и перейти в заключительное состояние.

Этот процесс выполняется следующими машинными командами:

$q_1\, 1\, q_1\, 1\, L$ и $q_1\, 0\, q_0\, 1\, S$. Тогда таблица машины будет иметь вид:

	0	1
q_1	q_0 1 S	q_1 1 L

Пример 3. Составить таблицу машины Тьюринга, вычисляющую функцию $f(x) = x + y$.

◄При $x = 3$ и $y = 2$ начальная и заключительная конфигурации этой машины будут иметь вид:

$$0\, q_1 1\,1\,1\,1\,0\,1\,1\,1\,0 \qquad\qquad 0\, q_0 1\,1\,1\,1\,1\,1\,0.$$

Алгоритм перехода от начальной конфигурации к заключительной может быть следующим. В ячейку, где записан*разделительный* 0, записать 1, а в *двух левых ячейках* кода числа x вместо 1 записать 0.

Вначале *заменим* крайнюю *левую* 1 числа x на 0 и *передвинем* головку машины *вправо*. При этом машина должна *перейти в новое состояние*, например в q_2, т.к. если она останется в состоянии q_1, то она в следующие моменты времени *заменит* остальные 1 числа x на 0. Эти действия соответствуют машинной команде: $q_1 1\, q_2 0\, R$.

После выполнения этой команды получим конфигурацию:

$0\, q_2\ 1\,1\,1\,0\,1\,1\,10.$

Затем надо передвигать головку машины *вправо*, проходя все ячейки с 1 *до*разделительного 0, *сохраняя* 1 в этих ячейках. Это соответствует команде $q_2 1 q_2 1 R$, которую машина будет выполнять каждый раз, когда в просматриваемой ячейке записана 1.

Дойдя до ячейки с разделительным нулем, машина будет находиться в состоянии q_2, т.е. получим конфигурацию:

$$0\ 1\ 1\ 1\ q_2 0\ 1\ 11\ 0.$$

Согласно алгоритму в эту ячейку надо *записать* 1. При этом можно головку машины передвинуть *влево*, что будет соответствовать направлению дальнейшего ее передвижения. Остаться в состоянии q_2 нельзя, т.к. тогда головка машины в следующие моменты времени будет передвигаться влево, согласно предыдущей команде. Эти действия будут соответствовать команде $q_2\ 0\ q_3 1\ L$. После выполнения этой команды получим следующую конфигурацию:

$$0\ 1\ 1 q_3\ 1\ 1\ 1\ 1\ 1\ 0.$$

Далее надо передвигать головку машины *влево*вдоль всех ячеек с 1, *сохраняя их* в этих ячейках. Это соответствует команде: $q_3 1\ q_3 1\ L$.

Выполняя эту команду при переходе от ячейки к ячейке, головка машины дойдет до ячейки с 0. Тогда получим конфигурацию:
$0 q_3 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0.$

Согласно алгоритму надо в ячейку с 1 записать 0. Значит надо передвинуть головку машины на одну ячейку вправо. Это соответствует команде: $q_3\ 0\ q_4 0\ R$. Выполнив эту команду, получим конфигурацию:

$$0\ q_4 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0.$$

Осталось в ячейку с левой 1 *записать* 0, перевести машину в *заключительное* состояние и передвинуть головку *вправо*. Это соответствует команде: $q_4 1\ q_0 0R$. После выполнения этой команды получим заключительную коуфигурацию:

$$0\ q_0 1\ 1\ 1\ 1\ 1\ 1\ 0.$$

По приведенным выше машинным командам легко составить таблицу искомой машины, которая будет иметь вид:

Таблица 8

	0	1
q_1	-	q_2 0 R
q_2	q_3 1 L	q_2 1 R
q_3	q_4 0 R	q_3 1 L
q_4	-	q_0 0 R

Пример 4. Составить таблицу машины Тьюринга, вычисляющую функцию:

$$f(x) = \left[\frac{3}{2-x}\right] = \begin{cases} ..1,..x=0, \\ ..3,..x=1, \\ ..\textit{не..существует},..x>1. \end{cases}$$

◀В данном примере работа машины будет разная для разных начальных конфигураций: x = 0, x = 1 и x> 1. Все эти конфигурации в общем виде можно записать так:

$$q_1\,1?\,?\,0. \tag{1}$$

В этой записи вместо ? может быть символ 0 или 1. Для определения того, какой символ находится в ячейке, изображаемой первым ? надо передвинуть головку машины *вправо, сохраняя*(или стирая) 1 в предыдущей ячейке и перевести машину в *другое состояние*. Это соответствует команде:

$$q_1\,1\,q_2\,1\,R. \tag{2}$$

Если в этой ячейке записан 0, то в этом случае из конфигурации (1) получим конфигурацию:

$$0\,1\,q_2\,0. \tag{3}$$

Данная конфигурация означает, что на ленте было записано значение аргумента x = 0. В этом случае значение функции f(0) = 1. Это означает, что заключительная конфигурация должна иметь вид:

$$0q_0^1\,1\,1\,0. \tag{4}$$

Переход от конфигурации (3) к конфигурации (4) можно осуществить следующей командой: $q_2\,0\,q_0^1\,1\,L$.

Если после выполнения команды (2) в следующей ячейке окажется 1, то из конфигурации (1) получим конфигурацию:

$$1q_2\,1?\;0. \tag{5}$$

Для определения записанного символа в следующей справа ячейке в (5), надо передвинуть головку машины *вправо,* сохраняя (или стирая) 1 в предыдущей ячейке и перевести машину в *другое состояние*. Это соответствует команде:

$$q_2\,1\,q_3\,1R. \tag{6}$$

Если в этой ячейке записан 0, то в этом случае из конфигурации (5) получим конфигурацию:

$$1\ 1q_3\ 0\ . \tag{7}$$

Данная конфигурация означает, что на ленте было записано значение аргумента $x = 1$. В этом случае значение функции $f(1) = 3$. Это означает, что заключительная конфигурация должна иметь вид:

$$0\ q_0\ 1\ 1\ 1\ 1\ 0. \tag{8}$$

Переход от конфигурации (7) к конфигурации (8) можно осуществить следующими переходами:

$1\quad 1q_3\ 0$

$0\quad 1q_4\ 1\quad 1\quad 0$

$0\quad q_4\ 1\quad 11\quad 0$

$q_4\ 0\quad 1\quad 1\quad 1\ 0$

$0\quad q_0^2\ 1\quad 1\quad 1\ 1\ 0$

Представленные переходы от одной конфигурации к другой соответствуют следующим машинным командам:

$$q_3\ 0\ q_4\ 1\ L,\qquad q_4\ 1\ q_4\ 1\ L, q_4\ 0\ q_0^2\ 1\ S.$$

Если после выполнения команды (6) в следующей ячейке окажется 1, то из конфигурации (5) получим следующую конфигурацию:

$$01\ 1\ q_3\ 1\ 1\ 1..0. \tag{9}$$

Полученная конфигурация означает, что на ленте было записано значение аргумента $x > 1$. Для таких значений аргумента значение функции *не определено*. Легко видеть, что если машина будет выполнять следующую команду: $q_3\ 1\ q_3\ 1\ S$, то она никогда не достигнет заключительного состояния, т.е. на ленте не будет сформировано значение функции при таких значениях аргумента. Это и означает, что значение функции не определено.

По приведенным командам составим таблицу искомой машины. Она будет иметь следующий вид:

	0	1
q_1		$q_2\ 1\ R$
q_2	$q_0^1\ 1\ L$	$q_3\ 1\ R$
q_3	$q_4\ 1\ L$	$q_3\ 1\ S$
q_4	$q_0^2\ 1\ S$	$q_4\ 1\ L$

2.101 Записать программу машины Тьюринга, вычисляющей функцию:

$a) I_2^3(x_1, x_2, x_3) = x_2$

$б) f(x, y) = x + y + 2$

$$в) f(x) = \frac{4}{2-x} = \begin{cases} 2, \text{ при } x = 0 \\ 4, \text{ при } x = 1 \\ \text{не определено, при } x > 2 \end{cases}$$

$$г) f(x) = x \dot{-} 1 = \begin{cases} 0, \text{ если } x = 0, \\ x - 1, \text{ если } x \geq 1; \end{cases}$$

$$д) f(x) = \left[\frac{4}{3-x} \right] = \begin{cases} 1, \text{ при } x = 0 \\ 2, \text{ при } x = 1 \\ 4, \text{ при } x = 2 \\ \text{не существует, при } x > 2 \end{cases}$$

2.102 Записать программу машины Тьюринга, вычисляющей функцию:

$a) I_2^3(x_1, x_2, x_3) = x_3$

$б) f(x, y) = x + y + 3$

$$в) f(x) = sg\ x = \begin{cases} 0, \text{ если } x = 0, \\ 1, \text{ если } x \geq 1; \end{cases}$$

$$г) f(x) = \left[\frac{3}{2-x} \right] = \begin{cases} 1, \text{ при } x = 0 \\ 3, \text{ при } x = 1 \\ \text{не определено, при } x > 1 \end{cases}$$

$$д) f(x) = \left[\frac{3}{x-1} \right] = \begin{cases} \text{не определено, при } x = 0,1 \\ 3, \text{ при } x = 2 \\ 1, \text{ при } x = 3,4 \\ 0, \text{ при } x > 4 \end{cases}$$

Построить машину Тьюринга, вычисляющую функцию f:

2.103 $f(x) = \overline{sg}\ x = 1 - sg\ x$;

2.104 $f(x) = x \dot{-} 1 = \begin{cases} 0, \text{ если } x = 0, \\ x - 1, \text{ если } x \geq 1; \end{cases}$

2.105 $f(x) = \left[\frac{1}{x} \right] = \begin{cases} 1, \text{ если } x = 1, \\ 0, \text{ если } x \geq 2, \\ \text{не определено, если } x = 0; \end{cases}$

2.106 $f(x) = 3x$;

2.107 $f(x) = 2^{1-x}$;

2.108 $f(x) = \left[x/2\right] = m,\ если\ x = 2m+1,\ m \geq 0$;
2.109 $f(x,y) = x+2$;
2.110 $f(x,y) = x \dot{-} y = \begin{cases} 0,\ если\ x \leq y, \\ x-y,\ если\ x > y; \end{cases}$
2.111 $f(x,y) = x-y$;
2.112 $f(x,y) = 2x+y$;
2.113 $f(x,y) = \dfrac{4-2x}{y^2}$.

Построить машину Тьюринга, вычисляющую функцию f:

2.114 $f(x) = x \dot{-} 2$;
2.115 $f(x) = \left[\dfrac{1}{x-3}\right]$;
2.116 $f(x) = \dfrac{2}{4-x}$;
2.117 $f(x) = x-5$;
2.118 $f(x,y) = x+y$;
2.119 $f(x,y) = \dfrac{x}{2}+y$;
2.120 $f(x,y) = \dfrac{2+x}{2-y}$;
2.121 $f(x,y) = \dfrac{2-x}{2\left|3-y^2\right|}$.

По программе машины Тьюринга T записать аналитическое выражение для функции $f(x)$ и $f(x,y)$, вычисляемых машиной T (в качестве начального состояния берется q_1, а в качестве заключительного – q_0):

1) T:

	q_1	q_2
0	$q_2 1L$	$q_0 0R$
1	$q_1 1R$	$q_2 1L$

; 2) T:

	q_1	q_2
0	$q_2 0R$	$q_1 0L$
1	$q_1 1R$	$q_0 0R$

;

3) T:

	q_1	q_2	q_3
0	$q_2 1R$	$q_0 0L$	$q_0 0R$
1	$q_1 0R$	$q_3 1L$	$q_3 1R$

;4) T:

	q_1	q_2	q_3
0	$q_2 0S$	$q_3 1L$	$q_1 1L$
1	$q_2 1R$	$q_1 0R$	$q_3 1R$

;

5) T:

	q_1	q_2	q_3	q_4
0	$q_2 0R$	$q_3 0R$	$q_0 1S$	$q_2 0R$
1	$q_2 0R$	$q_4 0R$	$q_3 1L$	$q_4 1R$

;

6) T:

	q_1	q_2	q_3	q_4
0	$q_2 0L$	$q_3 0L$	—	$q_1 0R$
1	$q_2 1R$	$q_2 1R$	$q_4 0L$	$q_4 1L$

.

2.122Используя машины T_1, T_2, T_3, T_4 и T_5, построить схему машин для алгоритма $\mathfrak{A}$, соответсвующего переходу из начальной в заключительную конфигурацию (здесь q_{10}, q'_{10}, q_{20}, q_{30}, q_{40}, q_{50} и q'_{50} – заключительные состояния соответствующих машин):

T_1:

	q_{11}	q_{12}
0	$q_{12} 0R$	$q_{10} 0R$
1	$q'_{10} 0R$	$q_{11} 1S$

, T_2:

	q_{11}	q_{12}	q_{23}
0	$q_{22} 0R$	$q_{23} 1R$	$q_{20} 1S$
1	$q_{21} 1R$	$q_{22} 1R$	—

,

T_3:

	q_{31}	q_{32}	q_{33}
0	$q_{32} 0L$	$q_{33} 0L$	$q_{33} 0R$
1	$q_{31} 1L$	$q_{32} 1L$	$q_{33} 1L$

, T_4:

	q_{41}
0	$q_{40} 1S$
1	$q_{41} 1R$

, T_5:

	q_{51}	q_{52}
0	—	$q'_{50} 1S$
1	$q_{52} 1R$	$q_{50} 1S$

:

1) $\mathfrak{A}$: $q_1 1^x \models q_0 1^{2x}$ $(x \geq 0)$;

2) $\mathfrak{A}$: $q_1 1^{x+1} \models q_0 1^{3x}$ $(x \geq 1)$;

§ 6. Алгоритм Маркова. Тезисы Чёрча, Тьюрига и Маркова

1. Алгоритм Маркова. Для рассмотрения алгоритмической системы Маркова надо ввести понятие *алфавита* и *подстановки*.

Абстрактным алфавитом называется любая конечная совокупность объектов, называемых буквами или символами данного алфавита.

Слово «абстрактный» для краткости будем опускать. Алфавит, как любое конечное множество, задается перечислением его элементов, т.е. символов.

Примеры алфавитов: A = { f, d, h, ?, * }, B = { x, y, m, n }.

Под *словом* в данном алфавите будем понимать любую конечную упорядоченную последовательность символов этого алфавита. Число символов в слове называется *длиной* этого слова

В представленном выше алфавите A словами будут следующие последовательности: d, * f, ? hh * ff, а в алфавите B - x, y, y y nxm.

Пусть P и Q некоторые слова в заданном алфавите. Тогда выражение P → Qозначает обозначает *подстановку* т.е. замену слова P на слово Q.

Подстановка P → Q называется *применимой*к слову R, если слово P является *частью* этого слова. Если в некоторое словоR не входит слово P, то к этому слову нельзя применить подстановку P → Q.

Если подстановку можно применить к данному слову, то в результате ее применения к этому слову получится новое слово.

Пример 1.
Пусть задан алфавит **A** = { а, в, с } и подстановка ав → са.

Данную подстановку можно применить к слову савваасв. В результате ее применения к этому слову получим новое слово ссаваасв.

Эту подстановку нельзя применить к слову васаас, т.к. слово ав не является частью этого слова.

Алгоритм Маркова *задан*, если задан алфавитA и система подстановок.

Работа алгоритма заключается в переработке входного слова в выходное слово, которая происходит следующим образом.

1. Для начала работы алгоритма, нужно задать входное слово в алфавите A.

2.Алгоритм работает согласно *правилам очередности* применения заданных подстановок. Пусть в процессе работы получено текущее слово R_i. Просматриваются заданные подстановки в том порядке, в котором они заданы в схеме и находится первая применимая подстановка к слову R_i. Применяя её, получим новое слово R_{i+1}.

Затем к полученному слову повторяем ту же процедуру, т.е. снова последовательно просматриваем подстановки, начиная с *первой*, и первую применимую подстановку применяем к этому слову. В результате получим второе новое слово и т.д.

3. Работа алгоритма заканчивается, когда к полученному на некотором шаге слову нельзя применить никакой из заданных подстановок, или в первый раз была применена *последняя* подстановка.

В обоих случаях считается, что алгоритм *перерабатывает входное* слово в *выходное* слово.

Пример 2.

Создать нормальный алгоритм Маркова, вычисляющий *сумму чисел.*

◂Пусть числа изображаются последовательностью единиц, а знак сложения знаком +. Тогда входным словом будут последовательности единиц, разделенные знаками +.

Например, *входное* слово 11+111+1 будет соответствовать сложению чисел: 2 + 3 + 1.

В результате работы алгоритма выходное слово должно быть в виде последовательности единиц. Для приведенного выше входного слова *выходным словом* должно быть слово 111111.

Рассмотрим *первый* из возможных нормальных алгоритмов Маркова, вычисляющий сумму чисел.

Алфавит: { 1, +, ^ }, где ^ - пустой символ.

Система подстановок:

$$1+ \rightarrow +1$$
$$++ \rightarrow +$$
$$+ \rightarrow \wedge$$

Далее представлена работа алгоритма по переработке входного слова в выходное т.е. последовательность получаемых при этом слов:

11+ 111+1
1+ 1111+1
+11111+1
+1111+11
+111+111
+11+1111
+1+11111
++111111
+111111
111111
111111

Второй из возможных нормальных алгоритмов, вычисляющих сумму чисел и являющийся *эквивалентным*первому алгоритму, имеет следующий вид:

Алфавит: { 1, + },

Система подстановок: +1 → 1
1 → 1

Работа алгоритма производится следующим образом:

11+111+1
11111+1
111111
111111

Пример 3.

Рассмотрим пример нормального алгоритма Маркова, вычисляющего умножение числа на 2, т.е. вычисляющего значенияфункции f(x) = 2x.

◄Пусть знаку умножения соответствует знак *. Тогда один из возможных алгоритмов, осуществляющий данное умножение, будет следующим.

Алфавит: A = { 1, *, a, ^ }, где ^ - пустой символ.

Система подстановок: 1*1 → *111
*1 → a
a1 → ^

Работа данного алгоритма по переработке входного слова 11*11 будет следующая:

1 1*1 1
1 *111 1
*111 111
a 11111
1111

Пример 4.

Рассмотрим задание нормального алгоритма Маркова, вычисляющего произведение двух чисел, т.е. значения функцииf(x,y) = xy.

◀Числа x и y также изображаются последовательностью единиц, а знаку умножения соответствует знак *. Тогда нормальный алгоритм, производящий данное умножение, может быть задан следующей алгоритмической системой.

Алфавит: { 1,*, T, Ф }.

Система подстановок:

$$*11 \to T*1,$$
$$*1 \to T,$$
$$1T \to T1Ф,$$
$$ФT \to TФ,$$
$$Ф1 \to 1Ф,$$
$$T1 \to T,$$
$$TФ \to Ф,$$
$$Ф \to 1$$
$$1 \to 1.$$

Можно создать нормальный алгоритм *эквивалентный* данному, который имеет меньшее число подстановок. Например, такой алгоритмической системой будет следующая система.

Алфавит: { 0, 1, *, T, ^ }, где ^ - пустой символ.

Система подстановок:

$$*1 \to T*$$
$$T* \to \wedge$$
$$1T \to 0T1$$
$$10 \to 01$$
$$T \to \wedge$$
$$0 \to 1$$
$$1 \to 1$$

Применить алгоритм Маркова к заданным словам. Выяснить задачу алгоритма

2.123

$$1-1 \to -$$
$$1- \to 1$$
$$- \to \wedge$$

$$11 \to 1*$$
$$*1* \to *$$
$$1*1 \to *1$$

1) 111–11111
2) 11111–111

2.124 1) 1111111
2) 11111111

2.125

$*11 \to T*1$

$*1 \to T$

$1T \to T1\Phi$

$\Phi T \to T\Phi$

$\Phi 1 \to 1\Phi$

$T1 \to T$

$T\Phi \to \Phi$

$\Phi \to 1$

$1 \to 1$

1) 111*11 2) 1111*11

2.126

$*1 \to T*$

$T* \to \Lambda$

$1T \to 0T1$

$10 \to 01$

$T \to \Lambda$

$0 \to 1$

$1 \to 1$

1) 111*11 2) 1111*11

2. **Общие вопросы теории алгоритмов.** Под алгоритмом понимается конечная совокупность точно сформулированных правил решения некоторого класса задач.

Рассмотрим пример *словесного* алгоритма вычисления *суммы* n чисел:

$$S=\sum_{i=1}^{n} A_i \ .$$

Процесс вычисления этой суммы может быть описан в виде следующего алгоритма:

1. Полагаем S равным нулю и переходим к п.2
2. Полагаем i равным единице и переходим к п.3.
3. Полагаем S равным $S + A_i$ и переходим к п.4.

4. Проверяем, равно ли i числу n. Если $i = n$, то вычисления прекращаем, иначе увеличиваем i на единицу и переходим к п.3.

Алгоритмы, в соответствии с которыми решение поставленных задач сводится к арифметическим действиям, называются *численными* алгоритмами.

Примерами численных алгоритмов могут служить алгоритм вычисления квадратного корня из положительного числа, алгоритм деления чисел и др.

Алгоритмы, в соответствии с которыми решение поставленных задач сводится к логическим действиям, называются ***логическими***.

Примерами логических алгоритмов могут служить алгоритмы поиска минимального числа среди n чисел, поиска определенного пути в графе, поиска пути в лабиринте и др.

Алгоритм – это определенное на некотором языке конечное предписание, задающее дискретную последовательность исполнимых элементарных операций для решения задачи.

Это определение, понятное в *интуитивном смысле*, не является, однако, формальным. Действительно, что означает «элементарная операция» ? Или «предписание»? С какими объектами работает алгоритм: числами, матрицами, словами? … Все это требует уточнения, если мы хотим говорить об алгоритме строго.

Алгоритмы в интуитивном смысле не являются математическими объектами, к ним не применимы формальные методы исследования и доказательства. Поэтому необходима формализация понятия алгоритма. Например, сравнение двух алгоритмов по эффективности, проверка их эквивалентности и т.д. возможны только на основе их формального представления.

Впервые необходимость формального определения алгоритма возникла в связи с проблемой *алгоритмической неразрешимости* некоторых задач. Долгое время математики верили в возможность того, что все строго поставленные математические задачи могут быть алгоритмически решены, нужно только найти алгоритм их решения.

Куртом Геделем (1931г.) было показано, что некоторые математические проблемы не могут быть решены с помощью алгоритмов из некоторого класса. Этот класс (класс *рекурсивных* функций) алгоритмов определяется некоторой формальной конкретизацией понятия алгоритма. Возник вопрос: являются ли алгоритмически неразрешимыми эти проблемы только в рамках этой модели алгоритма или же для решения этих проблем вообще нельзя придумать никакого алгоритма ни в каком смысле?

Общность результатов Геделя зависела от того, совпадает ли примененный им класс алгоритмов с классом всех алгоритмов интуитивном смысле. Поэтому поиск и анализ различных уточнений и формализаций алгоритма и соотношение этих формализаций с интуитивным понятием алгоритма является практически важным.

В теории алгоритмов большое внимание уделяется общим способам задания алгоритмов, характеризующимся свойством универсальности, т.е. таким способам, которые позволяют задать алгоритм, эквивалентный любому наперед заданному алгоритму.

Алгоритмической системой называется всякий общий способ задания алгоритмов.

К настоящему времени предложен ряд формальных определений алгоритмов – алгоритмических систем. После Геделя Алонзо Черч применил формализм, называемый λ-исчислением, Алан Тьюринг предложил абстрактное автоматическое устройство, которое сейчас называется машиной Тьюринга, и определил алгоритм как программу для этой машины, А.А. Марков определил алгоритм как конечный набор правил подстановок цепочек символов и т.д.

Ниже будут рассмотрены различные алгоритмические системы, позволяющие дать более точное определение алгоритма.

При описании алгоритмических систем применяются специальные формализованные средства, которые можно разделить на два вида: *алгебраические* и *геометрические*.

Соответственно этому и *теорию алгоритмов* можно *разделить* на *алгебраическую* и *геометрическую*.

К первому направлению относятся рекурсивные функции, машины Тьюринга, операторные системы Ван-Хао и А.А. Ляпунова и др.

Ко второму направлению относятся нормальные алгоритмы А.А. Маркова, блок-схемный метод и др.

К числу общих свойств алгоритмов относятся дискретность, массовость и результативность.

Дискретность – четкая разбивка алгоритма на отдельные шаги (этапы).

Массовость – применимость алгоритма для множества слов.

Результативность – для любого входного слова из области определения алгоритм через конечное число шагов приводит к выходному слову.

Областью применимости алгоритма называется множество входных слов, для которых алгоритм результативен.

Два алгоритма *эквивалентны*, если совпадают их области применимости и результаты переработки любого слова из этой области.

В общем случае различают еще случайные и самоизменяющиеся алгоритмы.

Алгоритм называется *случайным*, если в системе правил, задающих алгоритм, предусматривается возможность случайного выбора тех или иных слов или правил. Им соответствуют *многозначные* алфавитные операторы.

Алгоритм называется *самоизменяющимся*, если он не только перерабатывает входные слова, но и сам изменяется в процессе такой переработки. Результат работы самоизменяющегося алгоритма на то или иное входное слово зависит не только от этого слова, но и от истории предыдущей работы алгоритма.

3. Тезисы Чёрча, Тьюринга и Маркова.Пусть задана частичная числовая функция $f(x_1,x_2,\ldots x_n)$. Назовём $f(x_1,x_2,\ldots x_n)$*эффективно вычислимой*, если существует интуитивный алгоритм её вычисления (о понятии интуитивного алгорима см. выше). Согласно *тезису Чёрча*, всякая *эффективно вычислимая* функция является *частично-рекурсивной*, т.е. может быть постоена из трёх основных элементарных рекурсивных функций с помощью операторов суперпозиции, примитивной рекурсии и минимизации. Таким образом, тезис Чёрча устанавливает взаимно-однозначное соответстве между произвольным *интуитивным алгоритмом* вычисления функции и последовтельностью применений трёх операторов, определённых в алгоритмической системе рекурсивных функций. *Тезис Тьюринга* утверждает, что для произвольного интуитивного алгоритма можно построить *машину Тьюринга*, которая его реализует. А.А. Марковсформулировал следующий *принцип нормализации*: всякий алгоритм в алфавите А *эквивалентен* некоторому *нормальному* алгоритму в этом же алфавите.

В пользу данных тезисов говорит тот факт, что *никому* еще *не удалось* их опровергнуть.

Литература

1. Судоплатов С.В. Математическая логика и теория алгоритмов: учебник и практикум для академического бакалавриата / С.В. Судоплатов, Е.В. Овчинникова. — 5-е изд., стер. — М.: Изд-во Юрайт, 2016. – 255 с.
2. Алексеев В.Б. Лекции по дискретной математике. М.: Инфра-М, 2012.
3. Нефедов В.Н., Осипова В.А. Курс дискретной математики. МАИ, 1992.
4. Акимов О.Е. Дискретная математика: логика, группы, графы. М.: Лаборатория базовых знаний, 2003. 376 с
5. Гаврилов Г.П., Сапоженко А.А. «Задачи и упражнения по дискретной математике» – М.: Физматлит, 2005. 416 с.
6. Костылев В.Г. Математическая логика и теория алгоритмов. Конспект лекций. – М.: МГТУ «Станкин», 2003. 88 с.
7. Новиков П.С. Элементы математической логики. – М.: Наука, 1973.
8. Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теории алгоритмов. – М.: Физматлит, 2001.

Приложение I

1. Законы коммутативности $A \circ B = B \circ C, \circ \in \{\wedge, \vee, \sim\}$;

2. Законы ассоциативности $A \circ (B \circ C) = (A \circ B) \circ C, \circ \in \{\wedge, \vee, \sim\}$;

3. Законы дистрибутивности:
– конъюнкции относительно дизъюнкции
$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$;
– дизъюнкции относительно конъюнкции
$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$;

4. Законы идемпотентности:
$A \wedge A = A$ – идемпотентность $\wedge$;
$A \vee AA$ –идемпотентность $\vee$;
5. Закон двойного отрицания: $\overline{\overline{A}} = A$;
6. Законы де Моргана:
$\overline{A \wedge B} = \overline{A} \vee \overline{B}$ - первый закон де Моргана;
$\overline{A \vee B} = \overline{A} \wedge \overline{B}$ - второй закон де Моргана;
7. Законы поглощения:
$A \wedge (A \vee B) = A$ – первый закон поглощения
$A \vee (A \wedge B) = A$– второй закон поглощения
8. Законы поглощения с отрицанием:
$A \wedge (\overline{A} \vee B) = A \wedge B$, $A \vee (\overline{A} \wedge B) = A \vee B$;
9. Законы склеивания:
$(A \wedge B) \vee (A \wedge \overline{B}) = A$
$(A \vee B) \wedge (A \vee \overline{B}) = A$
10. Связи между логическими операциями:
$A \sim B = (A \to B) \& (B \to A) = (A \& B) \vee (\overline{A} \& \overline{B}) =$
$= (A \vee \overline{B}) \& (\overline{A} \vee B)$;
$A \to B = \overline{A} \vee B = \overline{A \& \overline{B}}$;
11. Законы логических констант:
1. $A \wedge \overline{A} =$ Л;
2. $A \vee \overline{A} =$ И;
3. $A \vee$ Л $= A$;
4. $A \wedge$ И $= A$;
5. $A \wedge$ Л $=$ Л;
6. $A \vee$ И $=$ И;

Приложение II

Система аксиом ФАТ:

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

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

III. $A \wedge B \to A$

IV. $A \wedge B \to B$

V. $(A \to B) \to ((A \to C) \to (A \to B \wedge C))$

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

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

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

IX. $(A \to B) \to (\overline{B} \to \overline{A})$

X. $A \to \overline{\overline{A}}$

XI. $\overline{\overline{A}} \to A$

Правила:

3 Подстановки: любую букву, входящую в аксиому (теорему) ФАТ можно заменить формулой и получить новую теорему ФАТ.

4 Отделения (modus ponens (m.p.)): $\mathrm{A}, \mathrm{A} \to \mathrm{B} \vdash \mathrm{B}$

Теоремы ФАТ (выводятся из аксиом по правилам вывода)

Теорема 1 :Пусть дано $\vdash \varPhi$, тогда $\vdash B \to \varPhi$, где В - любая формула

Теорема 2. $\vdash$A→A

Теорема дедукции: Пусть Г-список гипотез, А, В формулы.

Если Г, А $\vdash$ В тогда $\varGamma \vdash A \to B$.

Теорема 3 $\vdash$(A→B)→((B→C) →(A→C))

Теорема 4 $\vdash$ (A→(B→C)) →(B→(A→C))

Теорема 5. $\vdash A \to (B \to A \wedge B)$.

Теорема 6(а): $\vdash (A\to(B\to C))\to(A\wedge B\to C)$.

Теорема 6 (б): $\vdash (A\wedge B\to C)\to(A\to(B\to C))$

Правила вывода ФАТ (выводятся из аксиом по правилам вывода)

1. Правилосиллогизма: $A\to B,\ B\to C \vdash A\to C$

2. Правилоперестановки: $A\to(B\to C)\vdash B\to(A\to C)$

3. Правила с конъюнкцией:

1) $A,\ B\vdash A\wedge B$

2) $A\wedge B\vdash A$

3) $A\wedge B\vdash B$

4) $A\wedge B\vdash B\wedge A$

4. Правила соединения, разъединения посылок

Соединения посылок - $A\to(B\to C)\vdash A\&B\to C$

Разъединения посылок - $A\&B\to C\vdash A\to(B\to C)$

8. Правила с дизъюнкцией

1) $A\vdash A\vee B$

2) $B\vdash A\vee B$

3) $A\vee B\vdash B\vee A$

4) Доказательство разбором случаев. Пусть Г-список гипотез, А, В, формулы. Если $\Gamma, A\vdash C$ и $\Gamma, B\vdash C$, то $\Gamma, A\vee B\vdash C$

Приложение III.

Законы логики предикатов.

1) Перенос квантора через отрицание: Пусть $P(x)$ - формула логики предикатов, в которой x - свободная переменная, тогда

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

2) Вынесение квантора за скобки: Пусть $P(x), Q(x)$ - формулы логики предикатов, в которых x - свободная переменная, тогда

$$(\forall x)(P(x) \wedge Q(x)) = (\forall x)P(x) \wedge (\forall x)Q(x);$$
$$(\exists x)(P(x) \vee Q(x)) = (\exists x)P(x) \vee (\exists x)Q(x);$$

3) Пусть Q не зависит от x, тогда:

$$(\forall x)(P(x) \wedge Q) = (\forall x)P(x) \wedge Q;$$
$$(\forall x)(P(x) \vee Q) = (\forall x)P(x) \vee Q;$$
$$(\exists x)(P(x) \vee Q) = (\exists x)P(x) \vee Q;$$
$$(\exists x)(P(x) \wedge Q) = (\exists x)P(x) \wedge Q;$$

4) Перестановка одноименных кванторов.

Пусть x,y – свободные переменные формулы P(x,y), тогда

$$(\forall x)(\forall y)P(x,y) = (\forall y)(\forall x)P(x,y);$$
$$(\exists x)(\exists y)P(x,y) = (\exists y)(\exists x)P(x,y);$$

5) Переименование связанных переменных:

Пусть x – связанная переменная формулы A, тогда x можно заменить любой другой буквой, не входящей в A, в кванторе и всюду в области действия квантора.

Материалы для самостоятельной работы

Список теоретических вопросов для повторения лекционного курса

Математическая логика

1. Высказывания: простые, сложные, примеры высказываний. Истинностное значение высказывания.

2. Определение логичного рассуждения. Примеры.

3. Логичные рассуждения и проверка на тавтологию (как связаны)

4. Логичные рассуждения и проверка на противоречие (как связаны)

5. Дать определение конъюнкции и импликации, дизъюнкции и эквиваленции.
6. Определение формулы алгебры высказываний.
7. Свойство коммутативности и дистрибутивности. Какие логические связки им удовлетворяют?
8. Свойство ассоциативности и идемпотентности. Какие логические связки им удовлетворяют? Правила де Моргана.
9. Представление эквиваленции и импликации булевыми формулами.
10. Равносильные формулы. Формулы поглощения и расщепления.
11. Тавтология и противоречие. Свойства констант (6 законов).
12 .Двойственная формула. Принцип двойственности.
13. Проблема разрешимости в логике высказываний.
14. Определение КНФ. Как проверить, что формула является тавтологией с помощью КНФ?
15. Определение ДНФ. Как проверить, что формула является противоречием с помощью ДНФ?
16. Правило резолюций. Вывод формулы по правилу резолюций в логике высказываний (определение)
17. Алгоритм метода резолюций в логике высказываний
18 .Как задать аксиоматическую теорию (4 пункта).
19.Определение вывода формулы в аксиоматической теории.
20.Правила подстановки и m.p.
21.Теорема дедукции.
22. Правила силлогизма и разъединения посылок.
23. Правила перестановки и соединения посылок.
24. Полнота и непротиворечивость аксиоматической теории (определения). Независимость системы аксиом.

25. Определение предиката. Предметное множество и предметные переменные. Местность предиката.
26. Связывание кванторами всеобщности и существования (определение, примеры).
27. Интерпретация формулы. Равносильность формул в данной интерпретации.
28. Равносильность формул на множестве и в логике предикатов. Перенос квантора через отрицание.
29. Правила вынесения квантора за скобки.
30. Приведенная и нормальная формы формул (определение).
31. Выполнимость и общезначимость формулы логики предикатов.
32. Проблема разрешимости в логике предикатов. Теорема Черча-Тьюринга.

Теория алгоритмов

1. Элементарные рекурсивные функции
2. Оператор $S_m^n(\varphi, f_1, f_2, \ldots, f_m)$.
3. Общая схема примитивной рекурсии для функции 1 переменной.
4. Общая схема примитивной рекурсии для функции 2 переменных.
5. Оператор минимизации для функции 1 переменной.
6. Какая функция называется примитивно-рекурсивной.
7. Какая функция называется частично-рекурсивной.
8. Функции информационной ленты машины Тьюринга.
9. Функции считывающей-записывающей головки машины Тьюринга.
10. Функции управляющего устройства машины Тьюринга.
11. Внешний алфавит и внутренние состояния машины Тьюринга.
12. Конфигурация на ленте машины Тьюринга.
13. Команда машины Тьюринга.
14. Программа машины Тьюринга.
15. Применимость и неприменимость машины Тьюринга к данному слову.
16. Произведение машин Тьюринга.
17. Итерация машины Тьюринга по паре состояний.

18. Разветвление машин Тьюринга.

19. Схемы машин Тьюринга.

20. Кодирование чисел и наборов чисел для машин Тьюринга.

21. Вычислимость по Тьюрингу.

22. Программа вычисления 0(x).

23. Программа для S(x).

24. Программа вычисления $I_2(x, y, z)$.

25. Нормальный алгоритм Маркова: алфавит, подстановка, правила очередности применения подстановок.

26. Пример алгоритма Маркова.

27. Тезисы Черча, Тьюринга, Маркова.

28. Временная сложность в наихудшем случае и асимптотическая сложность алгоритма

29. Классы P, NP

ВС (доказать)

1. Теорема1 исчисления высказываний
2. Теорема 2 исчисления высказываний
3. Правило силлогизма
4. Теорема 3 исчисления высказываний
5. Правило перестановки
6. Теорема 4 исчисления высказываний
7. Правила с конъюнкцией
8. Теорема 5 исчисления высказываний
9. Правило соединения посылок
10. Теорема 6 (а) исчисления высказываний
11. Правило разъединения посылок
12. Теорема 6 (б) исчисления высказываний

13. Правило резолюций в логике предикатов. Приложение: решение примера «родственные отношения»

14. Доказать примитивную рекурсивность функций

$x + y, xy, x^y, x!, x \div y, |x - y|, \min(x, y), \max(x, y), sg(x)$.

15. Самоприменимые машины. Доказать неразрешимость проблемы самоприменимости машин Тьюринга

16. Проблема применимости к входному слову. Доказать неразрешимость данной проблемы для машин Тьюринга.

Тестовые задания для подготовки к контрольным работам

<u>Задание к КР №1</u>

1. Приведением к КНФ (ДНФ) установить является ли формула тавтологией (противоречием).

2. и 3.— доказать утверждение в исчислении высказываний

4. Проверить равносильность формул в интерпретациях I,II :

x	y	P_1	P_2
1	1	И	Л
1	2	Л	И
2	1	Л	Л
2	2	И	И

$x_1, x_2, x_3 \in \{1,2\}$

5. Для формулы Ф найти равносильную ей приведенную форму (нормальная, сколемовская и клаузальная формы дают повышение баллов)
6. Доказать методом резолюций
7. Доказать клаузу примера 3 другими способами, используя проверку на противоречие или тавтологию, а также методом резолюций.

Примерные варианты контрольной работы 1

Вариант 1

1. $\Phi = (A \sim B \vee \overline{C}) \to (A \sim \overline{C})$
2. $\overline{B \to \overline{\overline{A}}} \to A$
3. $A \to D \wedge \overline{\overline{B}} \mid\!- A \wedge B \to D$
4. $\Phi_1 = \overline{P(x_1, x_2)} \wedge P(x_3, x_2)$ и $\Phi_2 = P(x_3, x_1) \sim P(x_2, x_3)$
5. $\Phi = \left(\exists x \forall z P(x,z) \sim \exists y \overline{Q}(y)\right) \to \left(\exists p \forall z R(z,p)\right)$
6. $П \to Б, Ф \to В, Б \vee В \mid\!- \overline{П} \vee \overline{Ф}$

Задание к к.р. №2 (Рекурсивные функции и машины Т)

1. Построить результат суперпозиции
2. Составить схему примитивной рекурсии и найти f(3) или f(1,2)
3. Доказать примитивную рекурсивность функции
4. Найти $g(x) = \mu_x f(x)$
5. Записать программу для Т и применить к заданной конфигурации

6. Написать программу T по схеме машин
7. Написать программу вычисления $f(x)$ на машине Тьюринга, привести тестовые расчеты

Программы машин

A	0	1
q_1	$q_0\, 0L$	$q_1\, 1\, L$

D	0	1
q_1	$q_2\, 1\, L$	$q_0^1\, 0\, R$
q_2	$q_0^2\, 1\, R$	$q_2\, 1\, L$

B	0	1
q_1	$q_2\, 1\, L$	$q_1\, 1\, R$
q_2	$q_0\, 1\, R$	$q_2\, 1\, L$

C	0	1
q_1	$q_0^1\, 1\, R$	$q_0^2\, 0\, L$

E	0	1
q_1	$q_1\, 1\, L$	$q_0\, 0\, R$

Примерные варианты контрольной работы 2

Вариант №1

1) $h(x,y) = S_3^2(I_2^3(x,y,z), y \cdot S(x), x + S(S(y)), I_1^2(x,y))$

2) $f(x) = \textit{Пр}(2, x + 2y\ \ )$

3) $f(x,y) = 3^{3x+y}$

4) $f(x) = \begin{cases} 5x+2, & x \neq 3 \\ \textit{не определена}, & x = 3 \end{cases}$

5) $T = A^2 B^2, \quad 11011q_1 1$

6) $T = E\ \ \dot{C} \begin{cases} (1)\ A\ \ B \\ (2)\ \ddot{D} \begin{cases} (1)\ \dot{E} \\ (2)\ \ddot{A} \end{cases} \end{cases}$

7) f(x)=[3x/(3–x)]

Учебное пособие

Задачи по математической логике и теории алгоритмов

Елисеева Ю.В.,
Красикова Екатерина Михайловна,
Костылев В.Г.,
Уварова Людмила Александровна

Подписано в печать 25.12.2017
Формат 60x90 1/16 Бумага офсетная.
Печ. л. 5,0. Уч.-изд. л. 5,0.
Тираж 300 экз. Заказ № 437

Издательство «Янус-К»
127411, Москва, Учинская ул., д.1

Отпечатано в ООО «ИНФОРМ-СОФТ»
119034, Москва, Еропкинский пер., д.16

ISBN 5-8037-0725-2
9 785803 707257