**КОНТРОЛЬНАЯ РАБОТА ПО**

**МАТЕМАТИЧЕСКОЙ ЛОГИКЕ И ТЕОРИИ АЛГОРИТМОВ.**

**ЗАДАНИЕ №1: ЛОГИКА ВЫСКАЗЫВАНИЙ.**

Данное высказывание записать в виде формулы логики высказываний. Построить отрицание данного высказывания в виде формулы, не содержащей внешних знаков отрицания. Перевести на естественный язык.

**ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ №1.**

**Вариант 0.** Если слово ставится в начале предложения, то оно пишется с большой буквы.

**Вариант 1.** Если число оканчивается нулем, оно делится на 5.

**Вариант 2.** Тело, лишенное опоры, падает на землю.

**Вариант 3.** Иван и Петр знают Федора.

**Вариант 4.** Эта книга полезная и интересная.

**Вариант 5.** Этот актер играет в театре и не играет в кино.

**Вариант 6.** Если собаку дразнить, она укусит.

**Вариант 7.** Если вы владеете английским языком, вы справитесь с этой работой.

**Вариант 8.** Если функция нечетная, то ее график симметричен относительно начала координат.

**Вариант 9.** Если число делится на 3, то сумма его цифр делится на 3.

**ТЕОРИЯ И ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЯ №1.**

**1.1. Определение высказывания.**

***Определение 1.1.*** *Высказыванием* называется повествовательное языковоепредложение, относительно которого можно сказать истинно оно или ложно.

***Пример 1.1.***

Следующие утверждения являются высказываниями:

А) “Москву основал Юрий Долгорукий”.

Б) “В прямоугольном треугольнике квадрат гипотенузы равен сумме квадратов катетов”.

В) 22=5.

Высказывания А) и Б) истинны, а высказывание В) ложно.

***Пример 1.2.***

Следующие утверждения не являются высказываниями:

а) *a*+*b*=2.

б) “Математика – интересный предмет”.

В логике высказываний нас интересует не суть высказывания, а его истинность или ложность. Мы говорим, что существуют два истинностных значения: истина и ложь (“И” и “Л”). Двухэлементное множество {И, Л} есть *множество истинностных значений.* Высказывания будем обозначать большими буквами: *A*,*B*,*C*,*X*,*Y*,... Выражение *А*=И означает, что высказывание *А* истинно, а *X*=Л означает, что высказывание *X* ложно.

**1.2. Операции над высказываниями. Алгебра высказываний.**

Введем операции над высказываниями:

*Отрицанием* высказывания *А* называется высказывание *А*, которое истинно тогда и только тогда, когда высказывание *А* ложно. Чтобы составить отрицание *А* достаточно в разговорном языке сказать “неверно, что *А*”.

***Пример 1.3.***

*А*=“Каспаров – чемпион мира по шахматам”.

*А*=“Неверно, что Каспаров – чемпион мира по шахматам”.

Отрицание определяется следующей таблицей истинности (таб.1.1):

Таблица 1.1.

|  |  |
| --- | --- |
| *А* | *А* |
| Л  И | И  Л |

*Конъюнкцией* двух высказываний *А* и *B* называется высказывание *А*&*B*, истинное тогда и только тогда, когда истинны оба высказывания *А* и *B*. В разговорной речи конъюнкции соответствует союз “и”.

***Пример 1.4.***

*А*=“Треугольник прямоугольный”.

*B*=“Треугольник равнобедренный”.

*А*&*B*=“Треугольник прямоугольный и равнобедренный”.

Конъюнкция определяется следующей таблицей истинности (таб.1.2):

Таблица 1.2.

|  |  |  |
| --- | --- | --- |
| *А* | *B* | *А*&*B* |
| Л  Л  Л  И | И  Л  И  И | Л  Л  Л  И |

*Дизъюнкцией* двух высказываний *А* и *B* называется высказывание *А*∨*B*, ложное тогда и только тогда, когда ложны оба высказывания *А* и *B*. В разговорной речи конъюнкции соответствует союз “или”.

***Пример 1.5.***

*А*=“Иванов юрист”.

*B*=“Иванов экономист”.

*А*∨*B*=“Иванов юрист или экономист”.

Дизъюнкция определяется следующей таблицей истинности (таб.1.3):

Таблица 1.3.

|  |  |  |
| --- | --- | --- |
| *А* | *B* | *A*∨*B* |
| Л  Л  Л  И | И  Л  И  И | Л  И  И  И |

*Импликацией* двух высказываний *А* и *B* называется высказывание *А**B*, ложное тогда и только тогда, когда *А* истинно, а *B* ложно. Импликации соответствуют следующие выражения разговорной речи:

“*А* влечет за собой *B*”; или “из *А* следует *B*”; или “если *А*, то *B*”.

***Пример 1.6.***

*А*=“Треугольник равносторонний”.

*B*=“В треугольнике все углы равны”.

*А**B*=“Если треугольник равносторонний, то все углы равны”.

Импликация определяется следующей таблицей истинности (таб.1.4):

Таблица 1.4.

|  |  |  |
| --- | --- | --- |
| *А* | *B* | *А**B* |
| Л  Л  Л  И | И  Л  И  И | И  И  Л  И |

Импликация играет важную роль в логике высказываний. При учете смыслового содержания высказывания (а не только значений истинности), оборот “если, то” подразумевает причинно-следственную связь.

Истинность импликации означает лишь то, что, если истинна посылка, то истинно и заключение. При ложной посылке заключение всегда истинно. Так, истинными являются следующие импликации: “Если в доме 5 этажей, то Иванов живет в квартире 50”; “Если идет снег, то 22=5”.

***Пример 1.7.***

Рассмотрим четыре высказывания:

*A*=“Дважды два четыре”=И;

*B*=“Дважды два пять”=Л;

*C*=“Снег белый”=И;

*D*=“Снег черный”=Л.

Образуем четыре импликации:

*А**C*=“Если дважды два четыре, то снег белый”=ИИ=И;

*B**C*=“Если дважды два пять, то снег белый”=ЛИ=И;

*А**D*=“Если дважды два четыре, то снег черный”=ИЛ=Л;

*B**D*=“Если дважды два пять, то снег черный”=ЛЛ=И.

*Эквивалентностью* двух высказываний *А* и *B* называется высказывание *А*~*B*, истинное тогда и только тогда, когда оба высказывания *А* и *B* одновременно истинны или ложны. Говорят, что *А* эквивалентно *B* или *A* имеет место тогда и только тогда, когда имеет место *B*.

***Пример 1.8.***

*А*=“Треугольник равнобедренный”.

*B*=“В треугольнике углы при основании равны”.

*А*~*B*=“Треугольник является равнобедренным тогда и только тогда, когда углы при основании равны”.

Эквивалентность определяется следующей таблицей истинности (таб.1.5):

Таблица 1.5.

|  |  |  |
| --- | --- | --- |
| *А* | *B* | *А*~*B* |
| Л  Л  Л  И | И  Л  И  И | И  Л  Л  И |

Высказывания вместе с определенными для них операциями образуют *алгебру высказываний.*

**1.3. Формулы логики высказываний. Равносильность формул.**

***Определение 1.2.*** *Формула логики высказываний* определяется индуктивно следующим образом:

1. Любая высказывательная переменная, а также константы “И”, “Л” есть формула.

2. Если *A* и *B* – формулы, то *А*, *A*∨*B*, *A*&*B*, *А**B*, *А*~*B* есть формулы.

3. Ничто, кроме указанного в пунктах 1 – 2, не есть формула.

Две формулы называются *равносильными*, если на всех одинаковых наборах переменных значения этих формул совпадают.

Равносильность формул *A* и *B* будем обозначать следующим образом: *A**B*.

Для того, чтобы установить равносильность формул, можно составить таблицы значений для каждой формулы и сравнить их. Для равносильных формул эти таблицы совпадают. Другой способ установления равносильности формул заключается в использовании некоторых установленных равносильностей формул логики высказываний.

При доказательстве равносильности формул можно использовать *принцип двойственности.*

Символы “&”, “∨” называются *двойственными*.

Формула *F*\* называется *двойственной* формуле *F*, если она получена из *F* одновременной заменой всех символов “&”, “∨” на двойственные.

Например, *F*=*A*∨(*B*&*C*);

*F*\*=*A*&(*B*∨*C*).

*Принцип двойственности.*

Если *F**G*, то *F*\**G*

Все законы равносильности, имеющие место для формул булевых функций, справедливы и для формул логики высказываний, причем единице соответствует истинностное значение “И”, а нулю – “Л”. Приведем эти законы.

Для любых формул *A*,*B*,*C* справедливы следующие равносильности:

1. *Коммутативность*.

а) *A*&*B**B*&*A* (для конъюнкции);

б) *A*∨*B**B*∨*A* (для дизъюнкции).

2. *Ассоциативность*.

а) *A*&(*B*&*C*)(*A*&*C*)&*C* (для конъюнкции);

б) *A*∨(*B*∨*C*)(*A*∨*B*)∨*C* (для дизъюнкции).

3. *Дистрибутивность*.

а) *A*&(*B*∨*C*)*A*&*B*∨*A*&*C* (для конъюнкции относительно дизъюнкции);

б) *A*∨(*B*&*C*)(*A*∨*B*)&(*A*∨*C*) (для дизъюнкции относительно конъюнкции).

4. *Закон де Моргана*.

а) (*A*&*B*)∨*B* (отрицание конъюнкции есть дизъюнкция отрицаний);

б) (*A*∨*B*)*A*&*B* (отрицание дизъюнкции есть конъюнкция отрицаний).

5. *Идемпотентность*.

а) *A*&*A**A* (для конъюнкции);

б) *A*∨*A**A* (для дизъюнкции).

6. *Поглощение.*

а) *A*&(*A*∨*B*)*A* (1–ый закон поглощения);

б) *A*∨*A*&*B**A* (2–ой закон поглощения).

7. *Расщепление* (склеивание).

а) *A*&*B*∨*A*&(*B*)*A* (1–ый закон расщепления);

б) (*A*∨*B*)&(*A*∨*B*)*A* (2–ой закон расщепления).

8. *Двойное отрицание*.

(*A*)*A*.

9. *Свойства констант*.

а) *A*&И*A*; б) *A*&ЛЛ; в)*A*∨ИИ; г) *A*∨Л*A*; д) ЛИ; е) ИЛ.

10. *Закон противоречия*.

*A*&*A*Л.

11. *Закон “исключенного третьего”.*

*A*∨*A*И.

12. *A**B**A*∨*B*(*A*&*B*).

13. *A*~*B*(*A**B*)&(*B**A*)(*A*&*B*)∨(*A*&¬*B*)А∨*B*)&(*A*∨*B*).

Каждая из перечисленных равносильностей может быть доказана с помощью таблиц значений функций, составленных для выражений, стоящих слева и справа от символа “”.

Справедливы также обобщенные законы дистрибутивности и обобщенные законы де Моргана:

14. (*A*1∨*A*2∨...∨*An*)&(*B*1∨*B*2∨...∨*Bm*) 

*A*1&*B*1∨*A*1&*B*2∨...∨*A*1&*Bm*∨...∨*An*&*B*1∨*An*&*B*2∨...∨*An*&*Bm*.

15. (*A*1&*A*2&...&*An*)∨(*B*1&*B*2&...&*Bm*) 

(*A*1∨*B*1)&(*A*1∨*B*2)&...&(*A*1∨*Bm*)&...&(*An*∨*B*1)&(*An*∨*B*2)&...&(*An*∨*Bm*).

16. (*A*1&*A*2&...&*An*)*A*1∨*A*2∨...∨*An*.

17. (*A*1∨*A*2∨...∨*An*)*A*1&*A*2&...&*An*.

В равносильностях 1 – 17 в качестве *A*,*B*,*Ai*,*Bi* могут быть подставлены любые формулы и, в частности, переменные.

***Пример 1.9.***

Доказать равносильность формул логики высказываний:

(*А**B*)&(*A*∨*B*)*B*.

Преобразуем левую часть, последовательно используя равносильности

12, 14, 10, 5а, 9г, 6б:

(*А**B*)&(*A*∨*B*)*А*∨*B*)&(*A*∨*B*)*А*&*A*∨*А*&*B*∨*B*&*А*∨*B*&*B**А*&*B*∨*B*&*А*∨*B**B*.

Равносильность доказана.

**1.4. Запись сложного высказывания в виде формулы логики высказываний.**

Если имеется несколько высказываний, то при помощи логических операций можно образовывать различные новые высказывания. При этом исходные высказывания принято называть *простыми*, а вновь образованные высказывания – *сложными.*

***Пример 1.10.***

Рассмотрим простые высказывания:

*A*="Будет холодное лето".

*B*="Будет дождливое лето".

*C*="Будет засушливое лето".

*D*="Будет хороший урожай".

Формула (*A*&*B*∨*C*)*D* соответствует сложному высказыванию:

''Если будет холодное и дождливое или засушливое лето, урожай будет плохим".

Язык логики высказываний удобен для записи математических утверждений. Всякая теорема имеет вид импликации: А*B* (прямая теорема); *B*А (обратная теорема); *B*А (противоположная теорема).

***Пример 1.11.***

*A*=“Треугольник прямоугольный”.

*B*=“Квадрат одной стороны равен сумме квадратов двух других сторон”

*А**B* (прямая теорема) = “Если треугольник прямоугольный, то квадрат одной стороны равен сумме квадратов двух других сторон”.

*B**А* (обратная теорема) = “Если квадрат одной стороны равен сумме квадратов двух других сторон, то треугольник прямоугольный”.

*B**А* (противоположная теорема) = “Если квадрат одной стороны не равен сумме квадратов двух других сторон, то треугольник не прямоугольный”.

В данном случае все три теоремы верны.

Равносильность *А**B**B**А* есть основание метода доказательства от противного. Например, для доказательства теоремы: “Если треугольник равнобедренный, то углы при основании равны” (*А**B*) достаточно доказать теорему: “Если углы при основании не равны, то треугольник не равнобедренный” (*B**А*).

Используя равносильные преобразования, можно получать различные формулировки одного и того же суждения, а также отрицаний суждений.

***Пример 1.12.***

Дано высказывание “Если политик обещает невыполнимое, то он обманывает людей”:

а) записать его в виде формулы логики высказываний;

б) произвести отрицание данного высказывания, так, чтобы результат не содержал внешних знаков отрицания; полученную при этом формулу записать на естественном языке.

Введем следующие высказывания:

*A*=”Политик обещает невыполнимое”.

*B*=“Политик обманывает людей”.

Данное нам высказывание может быть записано в виде формулы: *А**B*.

Построим отрицание высказывания, воспользовавшись равносильностью 12:

(*А**B*)*A*&*B*.

На естественном языке это может быть выражено следующим образом:

“Политик обещает невыполнимое, но он не обманывает людей”.

**1.5. Нормальные формы.**

В алгебре высказываний используют две нормальные фор­мы: *дизъюнктивную и конъюнктивную нормальные формы* *формулы*(ДНФ и КНФ).

ДНФ формулы есть формула, равносильная исходной формуле логики высказываний и записанная в виде дизъюнкции элементарных конъюнкций переменных, т.е.

*F*=*K*1∨*K*2∨*K*3∨..., где *Ki*=*A*&*B*&*C*&...

КНФ формулы есть формула, равносильная исходной формуле логики высказываний и записанная в виде конъюнкции элементарных дизъюнкций переменных, т.е.

*F*=*D*1&*D*2&*D*3&..., где *Di*=*A*∨*B*∨*C*∨...

Наибольшее распространение в логике высказываний по­лучили формулы вида КНФ, элементарные дизъюнкции которых *Di* принято называть *дизъюнктами*, а члены каждого дизъюнкта *A*,*B*,*C* – *атомами*.

***Пример 1.13.***

Указать, в каких нормальных формах находятся следующие формулы логики высказываний.

a) *A* – ДНФ и КНФ.

b) (*A*∨*B*)&*C* – КНФ.

c) *A*∨*B*∨*C* – ДНФ и КНФ.

d) (*A*∨*B*)&(*A*∨*C*) – КНФ.

e) *A*∨*B*&*C* – ДНФ.

f) *A*&*B*&*C* – ДНФ и КНФ.

g) *A*&*B*∨*A*&*C* – ДНФ.

Для каждой формулы логики высказываний функции *F* имеется равносильная ей дизъюнктивная нормальная форма (ДНФ) и конъюнктивная нормальная форма (КНФ).

*Алгоритм приведения формул логики высказываний к ДНФ (КНФ).*

*Шаг 1.* Все подформулы *F* вида *A**B* (т.е. содержащие импликацию) заменяем на *A*∨*B* или на (*A*&*B*) (в соответствии с равносильностью 12 раздела 1.3).

*Шаг 2.* Все подформулы *F* вида *A*~*B* (т.е. содержащие эквивалентность) заменяем на (*A*&*B*)∨(*A*&*B*) или на (*A*∨*B*)&(*A*∨*B*) (в соответствии с равносильностью 13).

*Шаг 3.* Все отрицания, стоящие над сложными подформулами, опускаем по законам де Моргана (в соответствии с равносильностями 4, 19, 20).

*Шаг 4.* Устраняем все двойные отрицания над формулами (в соответствии с равносильностью 8).

*Шаг 5.* Осуществляем раскрытие всех скобок по закону дистрибутивности конъюнкции относительно дизъюнкции для ДНФ (в соответствии с равносильностями 3а и 17) или по закону дистрибутивности дизъюнкции относительно конъюнкции для КНФ (в соответствии с равносильностями 3б и 18).

*Шаг 6.* Для получения более простой формулы целесообразно использовать равносильности 5,6,7,9,10,11.

***Пример 1.14.***

Дана формула *F*=(*A*&*B*)&(*A*∨*B*).

Привести формулу к виду ДНФ:

1) *F*=(*A*∨*B*)&(*A*∨*B*);

2) *F*=(*A*&*A*)∨(*A*&*B*)∨(*B*&*A*)∨(*B*&*B*);

3) *F*=(*A*&*B*)∨(*B*&*A*).

***Пример 1.15.***

Дана формула *F*=(*A*(*B*∨*C*))*D*.

Привести формулу к виду КНФ:

1) *F*=(*A*∨(*B*∨*C*))*D*;

2) *F*=(*A*∨(*B*∨*C*))∨*D*;

3) *F*=(*A*&(*B*)&*C*)∨*D*;

4) *F*=(*A*∨*D*)&(*B*∨*D*)&(*C*∨*D*).

Если каждая элементарная конъюнкция (или элементарная дизъюнкция) формулы содержат символы всех переменных, то такая формула называется *совершенной*. Есть *совершенные дизъюнктивные нормальные формы* формулы (СДНФ) и *совершенные конъюнктивные нормальные формы* формулы (СКНФ).

***Пример 1.16.***

Указать, в каких нормальных формах находятся формулы логики высказываний трех переменных.

a) *X&Y&Z* – СДНФ и КНФ;

b) *X*&*Y&Z*∨*X&Y*&*Z* – СДНФ;

c) *X*∨*Y*∨*Z* – СКНФ и ДНФ;

d) *X*&*Z* – ДНФ и КНФ;

e) (*X*∨*Y*∨*Z*)*&*(*X*∨*Y*∨*Z*) – СКНФ;

f) *X*∨*Y*∨*Z* – СКНФ и ДНФ;

g) (*X*∨*Y*)&(*X*∨*Z*) – КНФ.

Каждая формула, не равная тождественно “Л”, может быть приведена к СДНФ, которая является единственной с точностью до перестановки дизъюнктивных членов.

Каждая формула, не равная тождественно “И”, может быть приведена к СКНФ, которая является единственной с точностью до перестановки конъюнктивных членов.

*Алгоритм приведения формулы булевой функции к СДНФ*

*Шаг 1.* Используя алгоритм построения ДНФ, находим формулу *F*, являющуюся ДНФ данной формулы.

*Шаг 2.* Если в элементарную конъюнкцию *Ki* формулы *F* не входит ни переменная *A*, ни ее отрицание *A*, то на основании 1-го закона расщепления (равносильность 7а) заменяем *Ki* на (*Ki*&*A*)∨(*Ki*&*A*).

*Шаг 3.* В каждой элементарной конъюнкции переставляем конъюнктивные члены так, чтобы для каждого *i* (*i*=1,...,*n*) на *i*-ом месте была либо переменная *Ai*, либо ее отрицание *Ai*.

*Шаг 6.* Устраняем возможные повторения конъюнктивных членов согласно закону идемпотентности для дизъюнкции: *Ki*∨*Ki**Ki*.

***Пример 1.17.***

*F*=*A*&*B*∨*A*&*C*&*D*∨*A*&*B*&*C*&*D*.

Преобразовать формулу к виду СДНФ:

1) *F*=*A*&*B*&*C*∨*A*&*B*&*C*∨*A*&*B*&*C*&*D*∨*A*&*B*&*C*&*D*∨*A*&*B*&*C*&*D*;

2) *F*=(*A*&*B*&*C*&*D*)∨(*A*&*B*&*C*&*D*)∨(*A*&*B*&*C*&*D*)∨(*A*&*B*&*C*&*D*)∨

∨(*A*&*B*&*C*&*D*)∨(*A*&*B*&*C*&*D*)∨(*A*&*B*&*C*&*D*).

Алгоритм нахождения СКНФ полностью повторяет алгоритм нахождения СДНФ, если произвести двойственную замену “&” на “∨” и “∨” на “&”.

***Пример 1.18.***

*F*=(*A*∨*B*))&(*A*∨*B*∨*C*∨*D*).

Преобразовать формулу к виду СКНФ:

1) *F*=(*A*∨*B*∨*C*)&(*A*∨*B*∨*C*)&(*A*∨*B*∨*C*∨*D*);

2) *F*=(*A*∨*B*∨*C*∨*D*)&(*A*∨*B*∨*C*∨*D*)&(*A*∨*B*∨*C*∨*D*)&

&(*A*∨*B*∨*C*∨*D*)&(*A*∨*B*∨*C*∨*D*.

Совершенные нормальные формы удобно записывать, используя таблицы истинности, по значениям переменных и значению логической функции.

*Алгоритм представления логической функции, заданной таблицей, формулой в СДНФ.*

*Шаг 1.* Выбираем в таблице все наборы переменных *A*1,*A*2,...,*An*, для которых значение *F* равно “И”.

*Шаг 2.* Для каждого такого набора (строки таблицы) составляем конъюнкцию переменных, причем в эту конъюнкцию переменная *Ai* записывается без изменений (т. е *Ai*), если ее значение равно “И” и со знаком отрицания (т. е *Ai*), если ее значение равно “Л”.

*Шаг 3.* Составляем дизъюнкцию всех полученных конъюнкций. В результате получится формула данной функции в СДНФ.

Для получения формулы в СКНФ следует воспользоваться следующим алгоритмом.

*Алгоритм представления логической функции, заданной таблицей, формулой в СКНФ*

*Шаг 1.* Выбираем в таблице все наборы переменных *A*1,*A*2,...,*An*, для которых значение *F* равно “Л”.

*Шаг 2.* Для каждого такого набора (строки таблицы) составляем дизъюнкцию переменных, причем в эту дизъюнкцию переменная *Ai* записывается без изменений (т. е *Ai*), если ее значение равно “Л” и со знаком отрицания (т. е *Ai*), если ее значение равно “И”.

*Шаг 3.* Составляем конъюнкцию всех полученных дизъюнкций. В результате получится формула данной функции в СКНФ.

***Пример 1.19.***

Записать СДНФ и СКНФ для функции, заданной таблицей истинности (таб.1.6):

Таблица 1.6.

|  |  |  |  |
| --- | --- | --- | --- |
| *А* | *B* | *C* | *F*(*A*,*B*,*C*) |
| Л  Л  Л  Л  И  И  И  И | Л  Л  И  И  Л  Л  И  И | Л  И  Л  И  Л  И  Л  И | И  Л  Л  И  И  Л  Л  И |

a) Формула СДНФ:

*F*(*A*,*B*,*C*)=*А*&*B*&*C*∨*А*&*B*&*C*∨*А*&*B*&*C*∨*А*&*B*&*C*;

b) Формула СКНФ:

*F*(*A*,*B*,*C*)=(*A*∨*B*∨*C*)&(*A*∨*B*∨*C*)&(*A*∨*B*∨*C*)&(*A*∨*B*∨*C*).

*Замечание.* Т. к. всего строк в таблице функции 2*n*, то, если число дизъюнктивных членов в СДНФ равно *p*, а число конъюнктивных членов в СКНФ равно *q*, то *p*+*q*=2*n*.

Так, для функции, рассмотренной в примере 1.19, *n*=3, *p*=4, *q*=4, *p*+*q*=8=23.

**ЗАДАНИЕ №2: ЛОГИКА ПРЕДИКАТОВ.**

Данное суждение записать в виде формулы логики предикатов. Построить отрицание данного суждения в виде формулы, не содержащей внешних знаков отрицания. Перевести на естественный язык.

**ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ №2.**

**Вариант 1.** Все рыбы живут в воде.

**Вариант 2.** Некоторые абитуриенты поступили в институт.

**Вариант 3.** Студент ответил на некоторые вопросы.

**Вариант 4.** Автобус останавливается на всех остановках.

**Вариант 5.** Некоторые зрители не любят некоторых артистов.

**Вариант 6.** В этой местности иногда бывает снег.

**Вариант 7.** Не все металлы твердые.

**Вариант 8.** Некоторые студенты получают стипендию.

**Вариант 9.** Некоторые книги полезны.

**Вариант 0 (10).** Существуют непрерывные функции, которые не являются дифференцируемыми.

**ТЕОРИЯ И ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЯ №2.**

**2.1. Определение предиката. Кванторы.**

***Определение 2.1.*** *Предикатом P*(*x*1,*x*2,...,*xn*) называется функция, аргументы которой определены на некотором множестве *М*: *x*1,*x*2,...,*xn**M*, а сама она принимает два значения: “И” (истина) и “Л” (ложь). Таким образом, предикат осуществляет отображение *М*{И,Л}.

Переменные *x*1,*x*2,...,*xn* называются *предметными переменными*, а множество *M* – *предметной областью*.

Если все переменные *x*1,*x*2,...,*xn* принимают конкретные значения, то предикат есть не что иное, как высказывание, Таким образом, высказывание является частным случаем предиката. Можно сказать, что предикат есть высказывание, зависящее от параметров.

***Пример 2.1.***

а) *P*(*x*)=“*x* – четное число”. Здесь *М* – множество целых чисел, *x**M*.

б) *A*(*x*,*y*,*z*)=“*x*,*y*,*z* лежат на одной окружности”. Здесь М – множество точек плоскости, *x*,*y*,*z**M*

в) *B*(*x*,*y*)=“*x* старше *y*”. Здесь *M* – множество людей, *x*, *y* *M*.

Предикат от *n* переменных называется *n-местным предикатом.* Высказывание есть 0-местный предикат.

Как видно из примера 2.1, одноместный предикат отражает свойство некоторого объекта, а многоместный предикат выражает отношение между многими объектами.

Над предикатами можно производить обычные логические операции и получать при этом другие предикаты. Таким образом можно говорить об *алгебре предикатов.*

***Пример 2.2.***

Пусть *A*(*x*) – предикат “*x* делится на 3”, а *B*(*x*) – предикат “*x* делится на 2”. Тогда *A*(*x*)∨*B*(*x*) – предикат “*x* делится на 3 или на 2”, а *A*(*x*)&*B*(*x*) – предикат “*x* делится на 3 и на 2”.

Кроме операций логики высказываний, в логике предикатов используются особые логические символы – *кванторы* (были введены немецким математиком Г. Фреге)*.*

*Квантор общности.* Пусть *P*(*x*) – некоторый предикат, определенный для каждого *x* ∈ *М*. Тогда выражение ∀*xP*(*x*) является истинным высказыванием, если *P*(*x*) истинно для всякого *x*∈*М* и ложным в противном случае. Символ ∀*x* называется *квантором общности.* Выражение ∀*xP*(*x*) читается: “Для всех *x* имеет место *P*(*x*)”. В обычной речи квантору общности соответствуют слова: все, всякий, каждый, любой. Возможно отрицание квантора общности: ∀*xP*(*x*): “Не для всех *x* имеет место *P*(*x*)”.

***Пример 2.3.***

Пусть *P*(*x*) – предикат “*x* – четное число”. Тогда ∀*xP*(*x*) есть высказывание “Всякое *x* – четное число"=“Все числа – четные", которое истинно на множестве *M* четных чисел и ложно, если *М* содержит хотя бы одно нечетное число, например, если *M* – множество целых чисел. Отрицание ∀*xP*(*x*) есть высказывание “Не всякое *x* – четное число"=“Не все числа – четные", которое истинно на множестве целых чисел и ложно на множестве четных чисел.

*Квантор существования.* Пусть *P*(*x*) – некоторый предикат, *x*∈*М*. Тогда выражение ∃*xP*(*x*) является истинным высказыванием, если *P*(*x*) истинно хотя бы для одного *x*∈*М* и ложным в противном случае. Символ “∃” называется *квантором существования.* Выражение ∃*xP*(*x*) читается: “Существует *x*, для которого имеет место *P*(*x*)”. В обычной речи квантору существования соответствуют слова: некоторый, несколько. Возможно отрицание квантора существования: ∃*xP*(*x*): “Не существует *x*, для которого имеет место *P*(*x*)”.

Кванторы существования и общности называются *двойственными* кванторами.

***Пример 2.4.***

Пусть, как и в примере 2.3, *P*(*x*) – предикат “*x* – четное число”. Тогда ∃*xP*(*x*) есть высказывание “Некоторые *x* – четные числа”=“Существуют четные числа”, которое истинно на множестве *M*, содержащем хотя бы одно четное число и ложно, если *М* содержит только нечетные числа. Высказывание ∃*xP*(*x*)=“Неверно, что некоторые *x* – четные числа”=“Не существует четных чисел” истинно на множестве *M*, содержащем только нечетные числа и ложно, если *М* содержит хотя бы одно четное число.

Буква *x*, стоящая справа от квантора, называется *кванторной переменной* и должна присутствовать обязательно. Переменная, стоящая под знаком квантора, называется также *связанной переменной.* Несвязанная переменная называется *свободной.* Выражения∀*xP*(*x*) и ∃*xP*(*x*) не зависят от *x* и имеют вполне определенные значения. Поэтому переименование связанной переменной, т. е. переход, например, от выражения ∀*xP*(*x*) к ∀*yP*(*y*) не меняет его истинностного значения.

Кванторы могут применяться и к многоместным предикатам. При этом число свободных переменных уменьшается на единицу. Одноместный предикат при связывании переменной квантором становится 0-местным предикатом, т. е. высказыванием.

**2.2. Формулы логики предикатов. Равносильность формул.**

***Определение 2.2.*** *Формула логики предикатов* определяется индуктивно следующим образом:

1. Любая формула логики высказываний есть формула логики предикатов.

К новым формулам логики предикатов относятся следующие выражения:

2. Если *x*,*y*,*z*,... – предметные переменные, то предикаты *P*(*x*),*Q*(*x*,*y*),..., а также выражения с кванторами ∀*xP*(*x*), ∃*xR*(*x*), ∀*x*∃*yQ*(*x*,*y*), ... есть формулы.

3. Если *A* и *B* – формулы, то *A*, *A*∨*B*, *A*&*B*, *A**B*, *A*~*B* есть формулы, в которых свободные переменные формул *A* и *B* остаются свободными, а связанные переменные формул *A* и *B* остаются связанными.

4. Ничто, кроме указанного в пунктах 1 – 3, не есть формула.

Пусть *A* – формула, содержащая свободную переменную *x*. Тогда ∀*xA*, ∃*xA* – формулы, причем в первом случае *A* является *областью действия квантора общности,* а во втором – *областью действия квантора существования.*

***Пример 2.5.***

1. Следующие выражения являются формулами логики предикатов:

а) *A*&*B**C*, где *A*, *B*, *C* – высказывания.

б) ∀*x*∃*yQ*(*x*,*y*,*z*)&∀*x*∃*yP*(*x*,*y*,*u*).

Проанализируем последовательно это выражение.

Предикат *Q*(*x*,*y*,*z*) – формула;

Выражение ∀*x*∃*yQ*(*x*,*y*,*z*) – формула; переменные *x*, *y* – связанные, переменная *z* – свободная.

Предикат *P*(*x*,*y*,*u*) – формула.

Выражение ∀*x*∃*yP*(*x*,*y*,*u*) – формула; переменные *x*, *y* – связанные, переменная *u* – свободная.

Выражение ∀*x*∃*yQ*(*x*,*y*,*z*)&∀*x*∃*yP*(*x*,*y*,*u*) – формула; переменные *x*,*y* – связанные, переменные *z*, *u* – свободные.

2. Выражение ∀*x*∃*yP*(*x*,*y*,*z*)*Q*(*x*,*y*,*z*) формулой не является. Действительно, выражение ∀*x*∃*yP*(*x*,*y*,*z*) есть формула, в которой переменные *x* и *y* связанные, а переменная *z* свободная. Выражение *Q*(*x*,*y*,*z*) также формула, но в ней все переменные *x*,*y*,*z* свободные.

***Определение 2.3.*** Формулы *F* и *G*, определенные на некотором множестве *М*, называются *равносильными* *на этом множестве*, если при любых подстановках констант вместо переменных они принимают одинаковые значения.

***Определение 2.4.*** Формулы, равносильные на любых множествах, будем называть просто *равносильными.*

Переход от одних формул к равносильным им другим формулам логики предикатов может быть произведен по следующим правилам:

1. Все равносильности, имеющие место для логики высказываний, переносятся на логику предикатов.

***Пример 2.6.***

а) ∃*x*(*A*(*x*)∀*yB*(*y*))∃*x*(*A*(*x*)∨∀*yB*(*y*)).

б) ∀*xA*(*x*)(*B*(*z*)∀*xC*(*x*))(∀*xA*(*x*))∨*B*(*z*)∨∀*xC*(*x*).

в) (∃*xA*(*x*)∀*yB*(*y*))*C*(*z*)(∃*xA*(*x*)∀*yB*(*y*))∨*C*(*z*)

((∃*xA*(*x*))∨∀*yB*(*y*)∨*C*(*z*)∃*xA*(*x*)&(∀*yB*(*y*))∨*C*(*z*).

2.Перенос квантора через отрицание.

Пусть *A* – формула, содержащая свободную переменную *x*. Тогда

(∀*xA*(*x*)∃*x*(*A*(*x*)). (2.1)

(∃*xA*(*x*))∀*x*(*A*(*x*)). (2.2)

Правило переноса квантора через знак отрицания можно сформулировать так: знак отрицания можно ввести под знак квантора, заменив квантор на двойственный.

Справедливость равносильностей (2.1) и (2.2) вытекает из смысла кванторов. Так, левая часть (2.1) может быть прочитана следующим образом: “Неверно, что для всякого *x* имеет место *A*(*x*). В правой же части (2.1) утверждается: “Существует *x*, для которого *A*(*x*) не имеет места”. Очевидно, что оба утверждения одинаковы. В левой и правой частях (2.2) соответственно содержатся одинаковые утверждения: “Неверно, что существует *x*, для которого имеет место *A*(*x*)” и “Для всех *x* не имеет места *A*(х)”.

Пользуясь равносильностями (2.1) и (2.2), а также равносильностями логики высказываний, можно для каждой формулы найти такую равносильную ей формулу, в которой знаки отрицания относятся к элементарным высказываниям и элементарным предикатам.

***Пример 2.7.***

(∃*x*(*A*(*x*)∀*yB*(*y*))(∃*x*(*A*(*x*)∨∀*yB*(*y*))∀*x*((*A*(*x*)∨∀*yB*(*y*)))

∀*x*(*A*(*x*)&∀*yB*(*y*))∀*x*(*A*(*x*)&∃*y**B*(*y*)).

3. Вынос квантора за скобки.

Пусть формула *A*(*x*) содержит переменную *x*, а формула *B* не содержит переменной *x*, и все переменные, связанные в одной формуле, связаны в другой. Тогда

∀*xA*(*x*)*B*∀*x*(*A*(*x*)*B*). (2.3)

∀*xA*(*x*)&*B**x*(*A*(*x*)&*B*). (2.4)

∃*xA*(*x*)*B*∃*x*(*A*(*x*)*B*). (2.5)

∃*xA*(*x*)&*B*∃*x*(*A*(*x*)&*B*). (2.6)

Докажем формулу (2.3). Пусть формула ∀*xA*(*x*)*B* истинна на некотором множестве изменения переменных *М* и при некоторых фиксированных значениях свободных переменных. Тогда либо формула ∀*xA*(*x*), либо формула *B* истинна. Если истинна формула ∀*xA*(*x*), то формула *A*(х) истинна для всякого х, принадлежащего *М* и, следовательно, формула *A*(*x*)*B* тоже истинна для всякого *х* из *М*. Но тогда истинна формула ∀*x*(*A*(*x*)*B*).

Если формула ∀*xA*(*x*)*B* ложна, то ложны формулы ∀*xA*(*x*) и *B*. Следовательно, так как *B* не зависит от х, для всякого х∈М формула *A*(*x*)*B* ложна. Но тогда ложна формула ∀*x*(*A*(*x*)*B*).

Равносильности (2.4) – (2.6) доказываются аналогично.

4. Дистрибутивность квантора общности относительно конъюнкции и квантора существования относительно дизъюнкции.

Пусть формула *B*, так же, как и формула *A*, зависит от *х*. Тогда

∀*xA*(*x*)&∀*xB*(*x*)∀*x*(*A*(*x*)&*B*(*x*)). (2.7)

∃*xA*(*x*)∃*xB*(*x*)∃*x*(*A*(*x*)*B*(*x*)). (2.8)

Докажем (2.7). Пусть правая часть (2.7) истинна, т. е. ∀*x*(*A*(*x*)&*B*(*x*))=И. Тогда для любого *х*0∈*М* истинно значение *A*(*x*0)&*B*(*x*0). Поэтому значения *A*(*x*0) и *B*(*x*0) одновременно истинны для любого *х*0. Следовательно, истинна формула ∀*xA*(*x*)&∀*xB*(*x*).

Если же правая часть (2.7) ложна, то для некоторого *х*0∈*М* либо значение *A*(*x*0), либо значение *B*(*x*0) ложно. Значит, ложно либо∀*xA*(*x*0), либо∀*xB*(*x*0). Следовательно, ∀*xA*(*x*)&∀*xB*(*x*) ложно.

Равносильность (2.8) доказывается аналогично.

Дистрибутивные законы для квантора общности относительно дизъюнкции и квантора существования относительно конъюнкции, вообще говоря, не имеют места, т. е. формулы.

∀*xA*(*x*)∀*xB*(*x*) и ∀*x*(*A*(*x*)*B*(*x*)), а также ∃*xA*(*x*)&∃*xB*(*x*) и ∃*x*(*A*(*x*)&*B*(*x*)).

не являются равносильными, хотя они могут быть равносильными на некоторых множествах *М*.

***Пример 2.8.***

Показать, что формулы ∀*x*(*A*(*x*)*B*(*x*)) и ∀*xA*(*x*)∀*xB*(*x*)) не равносильны.

Пусть *М* – множество натуральных чисел, *A*(*х*)=“*х* – четное число”, *B*(*х*)=“*х* – нечетное число”. Тогда

∀*x*(*A*(*x*)*B*(*x*))=“Всякое натуральное число четное или нечетное”=И.

∀*xA*(*x*)=“Всякое натуральное число – четное”=Л,

∀*xB*(*x*)=“Всякое натуральное число – нечетное”=Л,

∀*xA*(*x*)∀*xB*(*x*)=“Всякое натуральное число четное или всякое натуральное число нечетное”=Л, т. е. формулы ∀*x*(*A*(*x*)*B*(*x*)) и ∀*xA*(*x*)∀*xB*(*x*) не равносильны.

***Пример 2.9.***

Показать, что формулы ∃*x*(*A*(*x*)&*B*(*x*)) и ∃*xA*(*x*)&∃*xB*(*x*) не равносильны.

Пусть *A*(х)=“У х голубые глаза”, *B*(х)=“У х черные глаза”. Тогда

∃*x*(*A*(*x*)&*B*(*x*))=“У некоторых голубые и черные глаза”=Л,

∃*xA*(*x*)=“У некоторых голубые глаза”=И,

∃*xB*(*x*)=“У некоторых черные глаза”=И,

∃*xA*(*x*)&∃*xB*(*x*)=“У некоторых голубые, и у некоторых черные глаза”=И:

т. е. формулы ∃*x*(*A*(*x*)&*B*(*x*)) и ∃*xA*(*x*)&∃*xB*(*x*) не равносильны.

5. Перестановка одноименных кванторов.

∀*x*∀*yA*(*x*,*y*)∀*y*∀*xA*(*x*,*y*). (2.9)

∃*x*∃*yA*(*x*,*y*)∃*y*∃*xA*(*x*,*y*). (2.10)

Разноименные кванторы переставлять, вообще говоря, нельзя.

***Пример 2.10.***

Пусть *М* – множество натуральных чисел, *A*(*x*,*y*)=“*x*>*y*”.

а) ∀*x*∀*yA*(*x*,*y*)=“Для всех *x* и *y* имеет место *x*>*y*”=Л;

∀*y*∀*xA*(*x*,*y*)=“Для всех у и х имеет место *х*>*y*”=Л;

∀*x*∀*yA*(*x*,*y*)∀*y*∀*xA*(*x*,*y*).

б) ∃*x*∃*yA*(*x*,*y*)=“Существуют такие *х* и *у*, что *х*>*y*”=И;

∃*y*∃*xA*(*x*,*y*)=“Существуют такие *y* и *x*, что *х*>*y*”=И;

∃*x*∃*yA*(*x*,*y*)∃*y*∃*xA*(*x*,*y*).

в) ∃*x*∀*yA*(*x*,*y*)=“Существует такое *х*, что для всякого у имеет место *x*>*y*”=Л (утверждается существование максимального числа на множестве натуральных чисел);

∀*y*∃*xA*(*x*,*y*)=“Для всякого *y* существует такое х, что *x*>*y*”=И;

∃*x*∀*yA*(*x*,*y*)≠∀*y*∃*xA*(*x*,*y*).

г) *A*(*х*,*у*)=“Книгу х читал человек у”.

∀*x*∃*yA*(*x*,*y*)=“Каждую книгу читал кто-нибудь”=И (например автор книги читал свою книгу);

∃*y*∀*xA*(*x*, *y*)=“Существует человек, который читал все книги”=Л;

∀*x*∃*yA*(*x*,*y*)≠∃*y*∀*xA*(*x*,*y*).

6. Переименование связанных переменных.

Заменяя связанную переменную формулы *A* другой переменной, не входящей в эту формулу, всюду: в кванторе и в области действия квантора, получим формулу, равносильную *A*.

***Пример 2.11.***

*A*=∀*xF*(*x*)∃*xG*(*x*).

Заменяя связанную переменную *x* на *y* в первом члене импликации и на *z* во втором, получим равносильную формулу:

*B*=∀*yF*(*y*)∃*zG*(*z*).

*A**B*.

7. В п. 4. была доказана дистрибутивность квантора общности относительно конъюнкции и квантора существования относительно дизъюнкции (тождества (2.7) и (2.8)). Этот факт означает, что в вышеуказанных случаях соответствующие кванторы могут быть вынесены за скобки и помещены впереди формулы, что и демонстрируют тождества (2.7) и (2.8).

Рассмотрим теперь случай, когда закон дистрибутивности, вообще говоря не применим. Сначала рассмотрим формулу ∀*xA*(*x*)∀*xB*(*x*) и применим правило переименования переменных. Получим

∀*xA*(*x*)∀*xB*(*x*)∀*xA*(*x*)∀*yB*(*y*). (2.11)

Так как ∀*yB*(*y*) не зависит от *x*, справедлива равносильность (2.3), причем *B*=∀*yB*(*y*). Поэтому в соответствии с (2.3) можно вынести за скобки ∀*x*:

∀*xA*(*x*)∀*yB*(*y*)∀*x*(*A*(*x*)∀*yB*(*y*)). (2.12)

Так как *A*(*x*) не зависит от *y*, справедлива равносильность (2.3), причем на этот раз *B*=*A*(*x*). Поэтому в соответствии с (2.3) можно вынести за скобки ∀*y*:

*A*(*x*)∀*yB*(*y*)∀*y*(*A*(*x*)*B*(*y*)) (2.13)

Учитывая (2.11), (2.12), (2.13), получим:

∀*xA*(*x*)∀*xB*(*x*)∀*x*∀*y*(*A*(*x*)*B*(*y*)). (2.14)

Таким образом за скобки выносятся два квантора ∀*x* и ∀*y*, а выражение в скобках не содержит знаков квантора.

Проведем аналогичные выкладки для формулы ∃*xA*(*x*)&∃*xB*(*x*):

∃*xA*(*x*)&∃*xB*(*x*)∃*xA*(*x*)&∃*yB*(*y*)∃*x*(*A*(*x*)&∃*yB*(*y*))∃*x*∃*y*(*A*(*x*)&*B*(*y*)). (2.15)

Аналогично можно доказать следующие равносильности:

∀*xA*(*x*)∃*xB*(*x*)∀*x*∃*y*(*A*(*x*)*B*(*y*)). (2.16)

∀*xA*(*x*)&∃*xB*(*x*)∀*x*∃*y*(*A*(*x*)&*B*(*y*)). (2.17)

**2.3. Приведенные и нормальные формулы.**

***Определение 2.5.*** Формулы, в которых из логических символов имеются только символы “&”, “” и “”причем символ “”встречается лишь перед символами предикатов, называются *приведенными* формулами.

***Пример 2.12.***

1. *A*(*x*)&*B*(*x*,*y*).
2. ∀*xA*(*x*)∃*x**B*(*x*,*y*).
3. (*A*(*x*)&*B*(*x*,*y*)).
4. ∀*xA*(*x*)∃*x**B*(*x*,*y*).
5. (∀*xA*(*x*)∃*x**B*(*x*,*y*)).

Первые две формулы в соответствии с определением являются приведенными, остальные не являются приведенными. В третьей формуле знак отрицания стоит перед формулой, а не перед символами предикатов. В четвертой формуле используется недопустимый для приведенной формулы символ импликации “”В пятой формуле знак отрицания стоит перед формулой и используется недопустимый для приведенной формулы символ импликации

***Теорема 2.1.*** Для каждой формулы существует равносильная ей приведенная формула, причем множества свободных и связанных переменных этих формул совпадают.

Действительно, пользуясь равносильностями логики высказываний, можно получить формулу, содержащую только символы “&”, “” и “”Применяя затем правило переноса квантора через знак отрицания, можно получить равносильную приведенную формулу. Такая приведенная формула называется приведенной формулой данной формулы. Строгое доказательство теоремы 2.1 содержится, например, в [6].

***Пример 2.13.***

Рассмотрим третью, четвертую и пятую формулы примера 2.12 и получим для них приведенные формулы.

Для третьей формулы по закону де Моргана:

 (*A*(*x*)&*B*(*x*,*y*))*A*(*x*)*B*(*x*,*y*).

Для четвертой формулы:

∀*xA*(*x*)∃*x**B*(*x*,*y*)∀*xA*(*x*)∃*x**B*(*x*,*y*)∃*x**A*(*x*)∃*x**B*(*x*,*y*).

Для пятой формулы:

(∀*xA*(*x*)∃*x**B*(*x*,*y*))∃*x**A*(*x*)∃*x**B*(*x*,*y*))

∃*x**A*(*x*))&∃*x**B*(*x*,*y*))∀*xA*(*x*)&∀*xB*(*x*,*y*).

***Определение 2.6.*** Приведенная формула называется *нормальной*, если она не содержит символов кванторов или все символы кванторов стоят впереди.

***Пример 2.14.***

1. ∀*x*∃*y*(*A*(*x*)*B*(*x*,*y*)) – нормальная формула.
2. ∀*x*(*A*(*x*))&∃*yB*(*x*,*y*) – приведенная формула, не являющаяся нормальной.

**2.4. Выражение суждения в виде формулы логики предикатов.**

Существуют две задачи, определяющие связь между суждениями и формулами логики предикатов:

1) выражение суждения в виде формулы логики предикатов;

2) интерпретация формулы логики предикатов.

Рассмотрим первую задачу.

*Суждение* – это мысль, в которой утверждается наличие или отсутствие свойств предметов, отношений между предметами.

*Простым суждением* назовем суждение, в котором нельзя выделить часть, в свою очередь являющуюся суждением. Среди простых суждений выделяют *атрибутивные суждения* и *суждения об отношениях.*

В атрибутивных суждениях выражается наличие или отсутствие у предметов некоторых свойств. Например, "Иванов - спортсмен", "Все сладкоежки любят конфеты", "Ни один студент нашей группы не знает испанский язык", "Некоторые океаны имеют пресную воду".

Все атрибутивные суждения можно разделить на следующие типы: "*a* есть *P*", "Все *S* есть *P*", "Ни один *S* не есть *P*", "Некоторые *S* есть *P*", "Некоторые *S* не есть *P*". Эти суждения следующим образом переводятся на язык логики предикатов:

"*a* есть *P*" – *P*(*a*);

"Все *S* есть *P*" – ∀*x*(*S*(*x*)*P*(*x*));

"Ни один *S* не есть *P*" – ∀*x*(*S*(*x*)*P*(*x*));

"Некоторые *S* есть *P*" – ∃*x*(*S*(*x*)&*P*(*x*));

"Некоторые *S* не есть *P*" – ∃*x*(*A*(*x*)&*P*(*x*)).

Полезно понять и запомнить следующее правило: если кванторная переменная связана квантором общности (∀), то в формуле используется знак импликации ()а если кванторная переменная связана квантором существования (∃), то в формуле используется знак конъюнкции (&).

***Пример 2.15.***

Перевести на язык логики предикатов следующие суждения:

а) Веста – собака.

Заменим имя "Веста" символом "в" и введем предикат *P*(*x*)="*x* – собака".

Наше суждение можно выразить формулой: *P*(в).

б) Всякая логическая функция может быть задана таблицей.

Введем предикаты *S*(*x*)="*x* – логическая функция"; *P*(*x*)="*x* может быть задана таблицей".

Наше суждение можно выразить формулой: ∀*x*(*S*(*x*)*P*(*x*)).

в) Ни один народ не хочет войны.

Введем предикаты *S*(*x*)="*x* – народ"; *P*(*x*)="*x* хочет войны".

Наше суждение можно выразить формулой: ∀*x*(*S*(*x*)*P*(*x*)).

г) Некоторые журналисты были в космосе.

Введем предикаты *S*(*x*)="*x* – журналист"; *P*(*x*)="*x* был в космосе".

Наше суждение можно выразить формулой: ∃*x*(*S*(*x*)&*P*(*x*)).

д) Некоторые современники динозавров не вымерли.

Введем предикаты *S*(*x*)="*x* – современник динозавров"; *P*(*x*)="*x* вымер".

Наше суждение можно выразить формулой: ∃*x*(*A*(*x*)&*P*(*x*)).

Суждения об отношениях выражают отношения между двумя, тремя и т. д. объектами. При переводе этих суждений в формулы используют многоместные предикаты и правила, рассмотренные выше. При переводе отрицаний суждений на язык формул применяется правило переноса квантора через знак отрицания и другие равносильные преобразования.

***Пример 2.16.***

Суждение “Некоторые студенты сдали все экзамены” записать в виде формулы логики предикатов. Построить отрицание данного суждения в виде формулы, не содержащей внешних знаков отрицания.

Перевести на естественный язык.

Введем предикаты: *A*(*x*)=“*x* – студент”; *B*(*y*)=“*y* – экзамен”, *C*(*x*,*y*)=”*x* сдал экзамен *y*”. Тогда предложение “Некоторые студенты сдали все экзамены” можно записать в виде следующей формулы:

∃*x*∀*y*(*A*(*x*)&*B*(*y*)*C*(*x*,*y*)).

Построим отрицание этой формулы, применяя равносильные преобразования:

∃*x*∀*y*(*A*(*x*)&*B*(*y*)*C*(*x*,*y*)))∀*x*∃*y*((*A*(*x*)&*B*(*y*)*C*(*x*,*y*))

∀*x*∃*y*(*A*(*x*)&*B*(*y*)&*C*(*x*,*y*).

Это предложение можно прочитать следующим образом:

“Каждый студент не сдал хотя бы один экзамен”.

Язык логики предикатов удобен для записи математических предложений: определений, теорем, необходимых и достаточных условий (см., например, [5]).

***Пример 2.17.***

Записать на языке логики предикатов следующее определение предела числовой последовательности: "Число *a* является пределом числовой последовательности {*an*}, если для любого положительного числа ε существует такой номер *n*0, что для всех натуральных чисел *n*, больших или равных *n*0, справедливо неравенство: |*an*–*a*|<ε".

Введем предикаты: *P*(ε)="ε>0"; *Q*(*n*)="*n* – натуральное число"; *R*(*n*,*n*0)="*n**n*0"; *S*(*n*,ε)="|*an*–*a*|<ε".

Определение предела последовательности может быть записано следующей формулой:

∀ε∃*n*0∀*n*(*P*(ε)&*Q*(*n*)&*Q*(*n*0)&*R*(*n*,*n*0)*S*(*n*,ε)).

***Пример 2.18.***

Записать в виде формулы логики предикатов великую теорему Ферма (была доказана в 1996 г. Э. Вайлсом (*Andrew* *Wiles*)): "Для любого целого *n*>2 не существует натуральных чисел *x*,*y*,*z*, удовлетворяющих равенству: *xn*+*yn*=*zn*".

Введем предикаты:

*N*(*x*)="*x* – натуральное число"; *M*(*x*)="*x*>2"; *P*(*x*,*y*,*z*,*n*)="*xn*+*yn*=*zn*".

Для любых чисел *x*,*y*,*z*,*n* условие (посылка) теоремы Ферма есть конъюнкция *N*(*x*)&*N*(*y*)&*N*(*z*)&*N*(*n*)&*M*(*n*), а заключение есть *P*(*x*,*y*,*z*,*n*). Поэтому теорема Ферма формулируется следующим образом:

∀*x*∀*y*∀*z*∀*n*(*N*(*x*)&*N*(*y*)&*N*(*z*)&*N*(*n*)&*M*(*n*)*P*(*x*,*y*,*z*,*n*)).

Если теорема имеет вид ∀*x*(*P*(*x*)*Q*(*x*)), то предикат *Q*(*x*) является следствием предиката *P*(*x*). При этом предикат *Q*(*x*) называется необходимым условием предиката *P*(*x*), а предикат *P*(*x*) – достаточным условием предиката *Q*(*x*).

***Пример 2.19.***

Запишем в виде формулы логики предикатов утверждение: "Если число делится на 6, то оно делится на 3".

Введем предикаты *P*(*x*)="*x* делится на 6"; *Q*(*x*)="*x* делится на 3". Наше утверждение формулируется следующим образом: ∀*x*(*P*(*x*)*Q*(*x*)).

Предикат *P*(*x*) (делимость на 6) является достаточным условием предиката *Q*(*x*) (делимость на 3). Предикат *Q*(*x*) (делимость на 3) является необходимым условием предиката *P*(*x*) (делимость на 6).

**ЗАДАНИЕ №3: ФОРМАЛЬНЫЕ АКСИОМАТИЧЕСКИЕ ТЕОРИИ (ИСЧИСЛЕНИЯ).**

Установить правильность рассуждения, построив вывод исчисления высказываний.

**ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ №3.**

**Вариант 1.** Если светофор зеленый, то проезд разрешен. Проезд не разрешен. Значит, светофор не зеленый.

**Вариант 2.** Если нет понятых, то обыск производить нельзя. Обыск можно производить. Следовательно, есть понятые.

**Вариант 3.** Плох тот солдат, который не мечтает стать генералом. Этот солдат хороший. Следовательно, он мечтает стать генералом.

**Вариант 4.** Если треугольник равносторонний, то он равнобедренный. Если треугольник равнобедренный, то углы при основании равны. Следовательно, если треугольник равносторонний, то углы при основании равны.

**Вариант 5.** Если спутник Земли пролетает над Южным полюсом, то он пролетает над Антарктидой. Этот спутник не пролетает над Антарктидой. Следовательно, он не пролетает над Южным полюсом.

**Вариант 6.** Петров студент или школьник. Он не школьник. Следовательно, он студент.

**Вариант 7.** Если солнце взошло, то настало утро. Утро не настало. Значит, солнце не взошло.

**Вариант 8.** Если Федор победит на соревнованиях, он войдет в сборную страны. Если он войдет в сборную страны, то будет участвовать в чемпионате мира. Следовательно, если Федор победит на соревнованиях, он будет участвовать в чемпионате мира.

**Вариант 9.** Если эта собака не обучена, ее нельзя отпускать без поводка. Эту собаку можно отпускать без поводка. Значит, эта собака обучена.

**Вариант 0 (10).** Если я устал, то не могу готовиться к занятиям. Если я не смогу приготовиться к занятиям, я не напишу контрольную работу. Я устал. Значит, я не напишу контрольную работу.

**ТЕОРИЯ И ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЯ №3.**

**3.1. Принципы построения формальных теорий.**

Формальная аксиоматическая теория считается заданной, если заданы:

1*. Символы.* Задано некоторое счетное множество символов теории.

2*. Формулы.* Определено некоторое множество формул, или правильно построенных выражений. Формулы задают язык теории.

2. *Аксиомы.* Выделяется множество формул, называемых аксиомами теории. Это множество может быть, как конечным, так и бесконечным.

4. *Правила вывода.* Задаются правила вывода как некоторые отношения на множестве формул. Если формулы *A*1,*A*2,…,*Ak*,*B* находятся в отношении *R*, то формула *B* называется *непосредственно выводимой* из *A*1,*A*2,…,*Ak* по правилу *R*. Это часто записывается следующим образом: .

*Выводом* формулы *B* из множества формул *Г*={*A*1,*A*2,…,*An*} называется последовательность формул *B*1,*B*2,…,*Bm* , такая, что *Bm* есть *B*, и для любого *i*, *i*=1,2,…,*m*, *Bi* – либо аксиома, либо формула из *Г*, либо непосредственно выводима из предыдущих формул. *B*1,*B*2,…,*Bi*–1. Это обозначается так: *Г*├*B* (*B* есть следствие *Г*) или *A*1,*A*2,…,*An*├*B*. Формулы *A*1,*A*2,…,*An* называются *допущениями* или *гипотезами*, про формулу *B* говорят, что она *выводима* из *A*1,*A*2,…,*An*.

Если *Г* – пустое множество, то *A* – *теорема*, а ее вывод называется *доказательством.* В этом случае пишут ├*A*.

Во всякой формальной теории существуют три проблемы, связанные с системой аксиом:

1) проблема непротиворечивости;

2) проблема независимости;

3) проблема полноты.

Для того, чтобы доказать, что система аксиом *непротиворечива*, необходимо и достаточно доказать, что какова бы ни была формула *F*, выводимая в рассматриваемой теории, формула ¬*F* не является выводимой в этой теории.

Доказательство *независимости* системы аксиом состоит в доказательстве невыводимости никакой из аксиом системы из других аксиом.

Проблема *полноты* состоит в доказательстве следующего факта. Какова бы ни была аксиома, не содержащаяся среди аксиом данной теории, добавление ее к исходной системе приводит к тому, что расширенная система аксиом становится противоречивой.

**3.2. Исчисление высказываний.**

В соответствии с общими принципами построения формальных систем (исчислений) исчисление высказываний определяется следующим образом.

1 *Символы* исчисления высказываний включают в себя:

а) знаки логических операций “”, “”

б) буквы *Xi* с целыми положительными индексами *i*;

в) скобки и запятую – (,).

2. *Формулами* исчисления высказываний являются:

а) все переменные *Xi*;

б) если *A* и *B* – формулы, то *A* – формула и *A**B* – формула.

Хотя для исчисления высказываний выбраны только два логических символа “”и “” и только два типа формул *A* и *A**B*, можно с помощью следующих известных равносильностей ввести и другие логические символы и формулы:

*A*&*B**A**B*);

*A**B**A**B*;

*A*~*B*((*A**B*)(*B**A*)).

3. *Аксиомы* исчисления высказываний. Существуют различные системы аксиом исчисления высказываний, обладающие свойствами непротиворечивости, независимости и полноты. Будем использовать следующую систему аксиом:

А1. *A*(*B**A*);

А2. (*A*(*B**C*))((*A**B*)(*A**C*));

А3. (*B**A*)((*B**A*)*B*).

Непосредственной проверкой можно убедиться, что аксиомы есть тождественно-истинные формулы. Например, для аксиомы А1:

*A*(*B**A*)*A**B**A*И.

4. *Правило вывода* в исчислении высказываний одно – *modus* *ponens* (*m*.*p*.) – правило заключения:

, или *A*, *A**B* ├*B*.

Аксиомы исчисления высказываний являются формулами. Аксиомы и формулы можно рассматривать как схемы, так что любую входящую в них переменную можно заменять формулами.

***Пример 3.1.***

Если в правиле *modus* *ponens* переменную *B* заменить формулой *A*&*B*, получим правило вывода

.

Всякую выведенную в исчислении высказываний формулу можно рассматривать как правило вывода, которое может быть присоединено к уже имеющимся правилам.

Вывод формулы представляет собой последовательность формул, сопровождаемых указаниями, является ли данная формула гипотезой, аксиомой или получена из других формул по некоторому правилу вывода. Принято вначале выписать все гипотезы и слева указывать номер шага вывода.

***Пример 3.2.***

Построим вывод формулы *A*├*B**A*.

(1) *A* – гипотеза;

(2) *A*(*B**A*) – аксиома А1;

(3) *B**A* – из (1) и (2) по *m*.*p*.

Очевидно, что любую равносильную формулу можно рассматривать как правило вывода. Например, закон де Моргана может быть представлен как следующее правило вывода: . Равносильность *A**B**B**A* порождает *закон контрапозиции:* .

С учетом сказанного перечислим правила вывода исчисления высказываний.

1. Введение конъюнкции: .

2. Удаление конъюнкции:  и .

3. Отрицание конъюнкции: .

4. Введение дизъюнкции:  и .

5. Удаление дизъюнкции:  и .

6. Отрицание дизъюнкции: .

7. Введение импликации: .

8. Удаление импликации:  (*m*.*p*.) и .

9. Отрицание импликации: .

10. Введение эквивалентности: .

11. Удаление эквивалентности:  и .

12. Введение отрицания: .

13. Удаление отрицания: .

14. Закон контрапозиции: .

Для построения выводов в исчислении высказываний полезной оказывается следующая теорема.

***Теорема дедукции*** (без доказательства)***.*** Пусть *Г* – множество формул, *A* и *B* – формулы, и имеет место вывод: *Г*, *A*├*B*. Тогда имеет место следующий вывод: *Г*├*A**B*.

Таким образом, если нужно вывести формулу вида *A**B* из множества формул (возможно, пустого), можно использовать дополнительное допущение *A*.

Важным следствием теоремы дедукции является *правило силлогизма* (дается без доказательства):

***Правило силлогизма*** (транзитивный вывод)***.***

*A**B*, *B**C*├*A**C*.

Рассмотрим примеры построения вывода в исчислении высказываний.

***Пример 3.3.***

а) Обосновать вывод *A*(*B**C*), *A*&*B*├*C*.

1. *A*(*B**C*) – гипотеза;
2. *A*&*B* – гипотеза;
3. *A* – из (2) и правила удаления конъюнкции;
4. *B**C* – из (1), (3) и *m*.*p*.
5. *B* – из (2) и правила удаления конъюнкции;
6. *C* – из (4), (5) и *m*.*p*.

б) Обосновать правильность следующего рассуждения, построив вывод:

Если число целое, то оно рациональное; Если число рациональное, то оно действительное. Число целое. Значит, оно действительное.

Сначала формализуем наше рассуждение, введя следующие высказывания:

*A*=“число целое”.

*B*=“число рациональное”.

*C*=“число действительное”.

Нужно построить следующий вывод: *A**B*, *B**C*, *A*├*C*.

Построим этот вывод.

1. *A**B* – гипотеза;
2. *B**C* – гипотеза;
3. *A* – гипотеза;
4. *A**C* – из (1) и (2) по правилу силлогизма;
5. *C* – из (3) и (4) по *m*.*p*.

в) Обосновать правильность следующего рассуждения, построив вывод:

Если бы Иван был умнее Петра, он решил бы эту задачу. Иван не решил эту задачу. Значит, он не умнее Петра.

Формализуем наше рассуждение, введя следующие высказывания:

*A*=“Иван умнее Петра”.

*B*=“Иван решил эту задачу”.

Построим следующий вывод: *A**B*, *B*├*A*.

(1) *A**B* – гипотеза;

(2) *B* – гипотеза;

(3) *B**A* – из (1) по закону контрапозиции;

(4) *A* – из (3) и (2) по *m*.*p*.

**ЗАДАНИЕ №4: НЕЧЕТКАЯ ЛОГИКА.**

Определить степень равносильности формул*.* и при условии, что  и принимают значения степеней истинности из множества {0,2;0,3}.

**ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ №4.**

|  |  |  |
| --- | --- | --- |
| № |  |  |
| 1 | & |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  | & |
| 5 |  |  |
| 6 | & |  |
| 7 | & |  |
| 8 |  |  |
| 9 |  | & |
| 0 (10) |  |  |

**ТЕОРИЯ И ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЯ №4.**

**4.1. Нечеткие множества.**

В 1965 году американский математик Лотфи Заде (*L*. *Zade*) опубликовал статью “Нечеткие множества” (“*Fuzzy* *sets*”). Было дано новое определение понятия множества, предназначенное для описания сложных плохо определенных систем, в которых наряду с количественными данными присутствуют неоднозначные, субъективные, качественные данные

Понятие множества лежит в основе всех математических конструкций, и статья Заде породила новое научное направление. Произошло “раздвоение” математики, появились нечеткие функции, нечеткие отношения, нечеткие уравнения, нечеткая логика и т. д. Эти понятия широко используются в экспертных системах, системах искусственного интеллекта. Изложим основные понятия нечетких множеств.

***Определение 4.1.*** Нечетким множеством  на множестве *X* назовем пару (*X*,*mA*), где *mA*(*x*) – функция, каждое значение которой *mA*(*x*)∈[0,1] интерпретируется как степень принадлежности точки *x*∈*X* множеству. Функция *mA* – называется *функцией принадлежности множества* *.*

Для обычного четкого множества *A* можно положить

*mA*(*x*) = 

Таким образом, обычное множество является частным случаем нечеткого множества.

Функцию принадлежности, как и всякую функцию, можно задавать таблично или аналитически.

***Пример 4.1*.**

Приведем пример нечеткого множества , которое формализует понятие "несколько", ясного лишь на интуитивном уровне.

Пусть *X*={1,2,3,…,*n*,…} – множество натуральных чисел, а функция *mA*(*x*) задана таблицей:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *x* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| *mA*(*x*) | 0 | 0,1 | 0,6 | 0,8 | 1 | 1 | 0,9 | 0,7 | 0,2 | 0 |

Аналогично можно ввести понятия "много", "мало", "около 100", "почти 20", и т.д.

Переменные, значениями которых являются нечеткие множества, называются *лингвистическими*. Это основной тип переменных в языке людей.

***Пример 4.2.***

Пусть *X*=(0,∞) – множество положительных чисел, а функция *mA*(*x*) задана формулой:

*mA*(*x*) = 

График этой функции изображен на рис. 4.1.

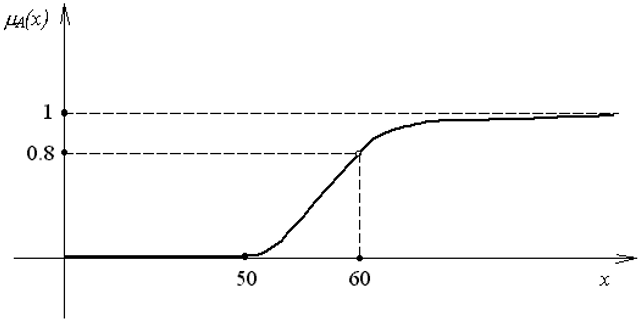


Рисунок 4.1.

Если переменную *x* интерпретировать как возраст, то нечеткое множество  соответствует понятию "старый". Аналогично можно ввести понятия "молодой", "средних лет" и т. д.

***Пример 4.3.***

Переменная "расстояние" принимает обычно числовые значения. Однако в предложениях естественного языка она может фигурировать как лингвистическая со значениями: "малое", "большое", "среднее", "около 5 км" и т. д. Каждое значение описывается нечетким множеством. Пусть речь идет о поездках на такси по городу. В качестве универсального множества *X* можно взять отрезок [0,100] км и задать функцию принадлежности значений так, как показано на рис. 4.2.

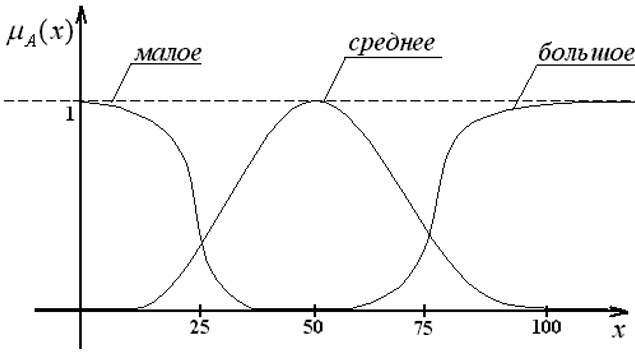


Рисунок 4.2.

**Операции с нечеткими множествами.**

Введем операции с нечеткими множествами аналогично операциям с обычными множествами.

Пусть  и  – два нечетких множества с функциями принадлежности *mA*(*x*) и *mB*(*x*).

В табл. 4.1 приведены названия основных операций, их лингвистический смысл и формула для определения функции принадлежности множества , которое является результатом соответствующей операции.

Таблица 4. 1.

|  |  |  |
| --- | --- | --- |
| Операции | Лингвистический смысл | Формула для *mC*(*x*) |
| Пересечение =∩ | И | *min*(*mA*(*x*),*mB*(*x*)) |
| Объединение =∪ | Или | *max*(*mA*(*x*),*mB*(*x*)). |
| Дополнение | Не | 1–*mA*(*x*) |
| Концентрация | очень | [*mA*(*x*)]2 |
| Размывание | не очень | [*mA*(*x*)]1/2 |

Нечеткое множество называется *пустым*, если *mA*(*x*) = 0 для всех *x*∈*X*.

***Пример 4.4.***

Пусть *X* – множество студентов,  – множество пожилых людей. Множество  – пустое, *mA*(*x*)=0 для всех *x*∈*X*, так как пожилых студентов, вообще говоря, не бывает.

Введенные для нечетких множеств операции позволяют конструировать сложные понятия из простых: очень много, не старше и не моложе и т. д. По аналогии с четкими множествами определяется отношение включения множества  в множество , а именно  является подмножеством тогда и только тогда, когда *mA*(*x*)≤ *B*(*x*) для всех *x*∈*X*.

Мы видим, что понятие нечеткого множества носит субъективный характер, такова и его формализация. Результаты, полученные с помощью аппарата алгебры нечетких множеств, должны носить качественный характер. Большей объективности выводов можно добиться, получив оценки функции принадлежности *mA*(*x*) путем опроса экспертов.

**4.2. Нечеткие высказывания.**

***Определение 4.2.*** *Нечетким высказыванием* называется высказывание *степень истинности* которого μ() можно оценить числом из интервала [0,1], μ()∈[0,1]. Если μ()=0,5, то высказывание называется *индифферентным.*

***Определение 4.3.*** *Нечеткой высказывательной переменной* называется нечеткое высказывание , степень истинности которого может меняться в интервале [0,1].

Так как степень истинности нечеткого высказывания не связана с сутью высказывания, будем в дальнейшем отождествлять нечеткое высказывание с его степенью истинности аналогично тому, как обычное четкое высказывание отождествлялось с его истинностью или ложностью (см. п. 1. 1). Нечеткие высказывания и степень их истинности будем обозначать большими буквами с тильдой: ,,, и т. д.

На множестве нечетких высказываний вводятся логические операции, аналогичные операциям алгебры высказываний.

1. *Отрицание* нечеткого высказывания:

=1–. (4.1)

1. *Конъюнкция* нечетких высказываний:

&=*min*(,). (4.2)

1. *Дизъюнкция* нечетких высказываний:

=*max*(,). (4.3)

1. *Импликация* нечетких высказываний:

=*max*(1–,). (4.4)

1. *Эквивалентность* нечетких высказываний:

~=*min*(*max*(1–,), *max*(,1–)). (4.5)

Старшинство операций принято в поядке1) – 5).

***Пример 4.5.***

Найти степень истинности высказывания

=()~(&)) при =0,8; =0,3.

Порядок действий определяется старшинством операций и скобками.

1. &=*min*(0,8;0,3)=0,3.

2. (&)=*max*(1–0,8;0,3)=0,3.

3. =*max*(0,8;0,3)=0,8.

4. =*min*(*max*(1–0,8;0,3), *max*(0,8;1–0,3))=*min*(0,3;0,8)=0,3.

Множество нечетких высказываний вместе с введенными на них операциями образуют *алгебру нечетких высказываний.*

***Определение 4.4.*** *Нечеткой логической формулой* называется:

а) любая нечеткая высказывательная переменная;

б) если  и  – нечеткие логические формулы, то

, &, , , ~ – тоже нечеткие логические формулы.

***Определение 4.5.*** Пусть (1,2,…,*n*) и (1,2,…,*n*) – две нечеткие логические формулы. *Степенью равносильности формул*  и называется величина

μ(,)={(α1,α2,…,α*n*)~(α1,α2,…,α*n*)} (4.6)

Здесь логические операции конъюнкции и эквивалентности имеют смысл, определенный выше для логических операций над нечеткими высказываниями, причем конъюнкция берется по всем наборам степеней истинности (α1,α2,…,α*n*) нечетких переменных (1,2,…,*n*).

Множество всех наборов степеней истинности (α1,α2,…,α*n*) нечетких переменных (1,2,…,*n*) назовем *полной областью определения Cn*. Очевидно, что множество *Cn* имеет мощность континуума в отличие от двузначной логики высказываний, где число всех наборов переменных конечно и равно 2*n*.

Если μ(,)=0,5, то нечеткие формулы  и  называются *индифферентными.*

Если μ(,)>0,5, то нечеткие формулы  и  называются *нечетко равносильными.*

Если μ(,)<0,5, то нечеткие формулы  и  называются *нечетко неравносильными.*

***Определение 4.6.*** *Степенью неравносильности формул*  и  называется величина

**(,)=1–μ(,).

***Пример 4.6***

Определить степень равносильности формул*.*

=,=&при условии, что  и принимают значения степеней истинности из множества {0,1;0,2}.

Перечислим все возможные наборы значений  и 

*A*1={0,1;0,1}; *A*2={0,1;0,2}; *A*3={0,2;0,1}; *A*4={0,2;0,2}.

Запишем формулы  и с учетом (4.1), (4.2), (4.4):

=*max*(1–,); =&1–&1–*min*(,).

Вычислим формулы  и  на каждом из четырех наборов *A*1 –*A*4:

1=*max*(1–0,1;0.1)=0,9.

2=*max*(1–0,1;0,2)=0,9.

3=*max*(1–0,2;0,1)=0,8.

4=*max*(1–0,2;0,2)=0,8.

1=1–*min*(0,1;0.1)=0,9.

2=1–*min*(0,1;0,2)=0,9.

3=1–*min*(0,2;0,1)=0,9.

4=1–*min*(0,2;0,2)=0,8.

Вычислим теперь степень равносильности формул  и  в соответствии с (4.6):

Для этого сначала вычислим (α1,α2,…,α*n*)~(α1,α2,…,α*n*)} для всех наборов *A*1–*A*4:

В соответствии с (4.5) имеем

~=*min*(*max*(1–,), *max*(,1–)).

Поэтому

1~1=*min*(*max*(1–0,9;0,9),*max*(0,9;1–0,9))=0,9.

2~2=*min*(*max*(1–0,9;0,9),*max*(0,9;1–0,9))=0,9.

3~3=*min*(*max*(1–0,8;0,9),*max*(0,8;1–0,9))=0,8.

4~4=*min*(*max*(1–0,8;0,8),*max*(0,8;1–0,8))=0,8.

Окончательно по (4.6) получим

μ(,)={(α1,α2,…,α*n*)~(α1,α2,…,α*n*)}=0,9&0,9&0,8&0,8=*min*(0,9;0,9;0,8;0,8)=0,8.

Формулы  и  нечетко равносильны.

На других наборах степеней истинности нечетких переменных  и  формулы  и могут быть нечетко неравносильны.

***Определение 4.7.*** Пусть (1,2,…,*n*) и (1,2,…,*n*) – две нечеткие логические формулы, рассмотренные на некотором множестве *M* изменения нечетких переменных 1,2,…,*n*. *Областью нечеткой равносильности* *формул*  и  называется подмножество множества *M*, на котором формулы  и  нечетко равносильны.

***Пример 4.7.***

Вернемся к примеру 4.7. Для этого примера множество *M* состоит из девяти наборов:

*M*={{0,1;0,1};{0,1;0,2};{0,2;0,1};{0,2;0,2}}.

На каждом наборе формулы  и  нечетко равносильны, так как μ(,)>0,5. Поэтому областью нечеткой равносильности будет все множество *M*.

***Определение 4.8.*** Если формула (1,2,…,*n*) на всех наборах переменных 1,2,…,*n* из некоторого множества *M* имеет степень истинности большую или равную 0,5, то она будет на нем *нечетко истинной*. Обозначается это так: =.

***Определение 4.9.*** Если формула (1,2,…,*n*) на всех наборах переменных 1,2,…,*n* из некоторого множества *M* имеет степень истинности меньшую или равную 0,5, то она будет на нем *нечетко ложной*. Обозначается это так: =.

***Пример 4.8.***

Покажем, что = и &= для всех значений нечеткой переменной :

0≤≤1.

Учитывая (4,1), (4.2), (4. 3), имеем

=*max*(,)=*max*(,1–)≥0,5.

&=*min*(,)=*min*(,1–)≤0,5.

**4.3. Нечеткие предикаты.**

***Определение 4.10.*** *Нечетким предикатом *(*x*1,*x*2,...,*xn*) называется нечеткая формула, переменные которой определены на некотором множестве М, *x*1,*x*2,...,*xn**M*, а сама она принимает значения из интервала [0,1].

Нечеткий предикат от *n* переменных называется *n-местным нечетким предикатом.* Нечеткое высказывание , задаваемое степенью истинности

μ()∈[0,1] является одноместным нечетким предикатом.

***Пример 4.9.***

Пусть М={0,1,2,3}. Зададим нечеткий предикат следующим образом:

**(*x*,*y*)=*xy*/9. Его значения определяются следующим образом:

**(0,*y*)=**(*x*,0)=0;

**(1,1)=1/9;

**(1,2)=**(2,1)=2/9;

**(2,2)=4/9;

**(1,3)=**(3,1)=1/3;

**(2,3)=**(3,2)=2/3;

**(3,3)=1;

***Определение 4.11.*** *Нечеткими кванторами “”* и “” называются логические символы, которые придают включающим их выражениям следующий смысл:

**(*x*1,*x*2,...,*xn*)=**(*x*1,*x*2,...,*xn*)=**(*x*1,*x*2,...,*xn*).

**(*x*1,*x*2,...,*xn*)=**(*x*1,*x*2,...,*xn*)=**(*x*1,*x*2,...,*xn*).

***Пример 4.10.***

Найдем значения степени истинности формул **(*x*,1) и **(*x*,1) для примера 4.9:

**(*x*,1)=*min*{**(0,1);**(1,1);**(2,1);**(3,1)}=*min*{0;1/9;2/9;1/3}=0.

**(*x*,1)=*max*{**(0,1);**(1,1);**(2,1);**(3,1)}=*max*{0;1/9;2/9;1/3}=1/3.

По аналогии с четкими предикатами вводятся также остальные понятия для нечетких предикатов.

**ЗАДАНИЕ №5: АЛГОРИТМЫ.**

Составить программу машины Тьюринга, которая заданное слово *P*вх преобразует в слово *P*вых.

**ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ №5.**

|  |  |  |
| --- | --- | --- |
| № | *P*вх | *P*вых |
| 1  2  3  4  5  6  7  8  9  0 (10) | 100  101  101  101  110  110  110  111  111  111 | 010  1010  1011  010  1100  1101  001  1110  1111  0001 |

**ТЕОРИЯ И ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЯ №5.**

**5.1. Определение алгоритма.**

Алгоритм можно определить, как некоторую процедуру, однозначно приводящую к результату. Это интуитивное определение, которое, однако, позволяет безошибочно определить, является ли рассматриваемый процесс алгоритмом или нет. Примеры алгоритмов:

1. Сортировка массива чисел в порядке возрастания.

2. Вычисление таблицы значений булевой функции, заданной формулой.

3. Вычисление чисел Фибоначчи по рекуррентному соотношению.

4. Решение системы линейных алгебраических уравнений методом исключения Гаусса.

Основные требования к алгоритмам:

1. Алгоритм применяется к *исходным данным* и дает *результаты*. Кроме того, в процессе работы алгоритма могут появляться промежуточные данные. Итак, каждый алгоритм имеет дело с *данными*: исходными, промежуточными и выходными. Данными могут быть числа, векторы, матрицы, массивы, формулы, рисунки (в графических системах).

2. Данные для своего размещения требуют *памяти.* Память состоит из ячеек*,* так что каждая ячейка может содержать один символ алфавита данных. Таким образом, объем данных и требуемая память согласованы.

3. Алгоритм состоит из отдельных *элементарных шагов* или *действий.* Причем множество различных шагов, из которых составлен алгоритм, конечно. Типичный пример множества элементарных действий – система команд ЭВМ.

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

5. Алгоритм должен удовлетворять требованию *результативности*, т. е. остановки после конечного числа шагов. В таком случае говорят, что алгоритм *сходится*.

Любая практическая задача требует предварительного задания исходных данных. Как правило, можно задать некоторое характерное число *n*. Например, для задачи сортировки массива чисел по возрастанию *n* – число чисел в массиве, для задачи решения системы линейных уравнений *n* – число уравнений. Характерное число задачи определяет *размерность задачи* как величину массива исходных данных.

С ростом характерного числа размерность задачи возрастает. Введем понятие скорости роста для функций, зависящих от целочисленного параметра *n*.

***Определение 5.1.*** Функции *f*(*n*) и *g*(*n*) имеют *одинаковую скорость роста*, если при достаточно больших *n*, начиная с некоторого *n*0, выполняется условие:

*C*1*g*(*n*)≤*f*(*n*)≤*C*2*g*(*n*),

где *C*1, *C*2 – некоторые константы.

***Определение 5.2.*** Скорость роста функции *f*(*n*) *ограничена снизу* скоростью роста функции *g*(*n*), если при достаточно больших *n*, начиная с некоторого *n*0, выполняется условие:

*C*1*g*(*n*)≤*f*(*n*),

где *C*1 – некоторая константа.

***Определение 5.3.*** Скорость роста функции *f*(*n*) *ограничена сверху* скоростью роста функции *g*(*n*), если при достаточно больших *n*, начиная с некоторого *n*0, выполняется условие:

*f*(*n*)≤*C*2*g*(*n*),

где *C*2 – некоторая константа.

***Определение 5.4.*** Скорость роста функции *f*(*n*) *больше* скорости роста функции *g*(*n*), если для любой сколь угодно большой константы *C*2 существует некоторое *n*0, начиная с которого выполняется условие:

*f*(*n*)≥*C*2*g*(*n*).

Для того чтобы более наглядно представить скорости роста функций, их сравнивают со скоростями роста хорошо известных функций. В качестве таковых чаще всего используют степенные функции *na*. При *a* = 1 скорость роста функции *na* называют *линейной*, при *a* = 2 – квадратичной, при *a* = 3 – кубической и т. д. Скорость роста вида *na* называют *полиномиальной*. Очевидно, что при возрастании *a* скорость роста тоже увеличивается. Для некоторых функций скорости роста превосходят в пределе при *n* → ∞ любую полиномиальную скорость. Такими функциями являются, например, 2*n*, *en*, *n*! Скорости такого типа называют *экспоненциальными*.

Обозначим через *r*(*n*) скорость роста размерности задачи. В задаче вычисления таблицы значений булевой функции *n* переменных скорость роста определяется таблицей значений переменных. Так как различных наборов переменных 2*n*, а каждый набор состоит из *n* символов, то размерность задачи равна *n\**2*n* и скорость роста будет экспоненциальной. В задаче решения системы линейных алгебраических уравнений методом исключения Гаусса наиболее быстро растет число элементов матрицы системы уравнений размером *n* × *n*. Поэтому скорость роста размерности этой задачи будет квадратичной, *r*(*n*) = *n*2.

Размерность задачи определяет память, необходимую для представления исходных данных в ЭВМ. Кроме того, необходима дополнительная память для размещения промежуточных данных. Величина этой памяти зависит от конкретного алгоритма и ее, как правило, нетрудно рассчитать.

Рассмотрим время реализации алгоритма – время счета.

Пусть при выполнении некоторого алгоритма выполняются элементарные операции *t*1,*t*2,…,*tk* (арифметические, логические и другие). Среднее время выполнения этих операций обозначим через *t*1,*t*2,…,*tk*. По аналогии с размерностью задачи введем понятие скорости роста числа выполняемых операций в зависимости от характерного числа *n*. Обозначим их для операций *t*1,*t*2,…,*tk* через *g*1(*n*),*g*2(*n*),…,*gk*(*n*). Без доказательства приведем следующее утверждение:

При *n* → ∞ скорость роста общего времени счета *T*(*n*) равна максимальной из скоростей роста числа элементарных операций *g*1(*n*),*g*2(*n*),…,*gk*(*n*) *независимо* от среднего времени их выполнения *t*1,*t*2,…,*tk*.

***Определение 5.5.*** Скорость роста общего времени счета *T*(*n*) называется *вычислительной сложностью* или просто *сложностью алгоритма*.

Обозначим сложность алгоритма через *f*(*n*). В зависимости от сложности все алгоритмы делятся на несколько классов.

***Определение 5.6.*** *Полиномиальными* называются алгоритмы, сложность которых ограничена некоторым полиномом.

***Пример 5.1*.**

Рассмотрим задачу определения максимального элемента в массиве из *n* чисел. Поскольку число операций сравнения постоянно и равно *n*–1, сложность алгоритма *f*(*n*)=*n*.

***Определение 5.7.*** *Экспоненциальными* называются алгоритмы, сложность которых при возрастании *n* превышает полином любой степени.

В примерах 5.2 и 5.3 точные алгоритмы имеют экспоненциальную точность.

***Пример 5.2*.**

Рассмотрим задачу коммивояжёра. Необходимо обойти *n* городов и вернуться в исходный пункт, так чтобы суммарный путь был минимальным. Количество всех возможных вариантов обхода равно 0.5*n*! Следовательно, сложность точного решения, основанного на переборе всех вариантов, равна *f*(*n*) = *n*!

***Пример 5.3*.**

Рассмотрим задачу вычисления конъюнктивной нормальной формы (КНФ) булевой функции *n* переменных. Количество всех наборов переменных равно 2*n*. Количество всех операций при переборе всех дизъюнкций пропорционально *n\**2*n*. Следовательно, сложность алгоритма *f*(*n*)=*n\**2*n*.

Экспоненциальные алгоритмы практически могут быть реализованы только при малых значениях *n* (обычно при *n*<10).

Задачи, для которых существуют точные алгоритмы решения полиномиальной сложности, называются *задачами класса P*.

Задачи, для которых не удается найти точные алгоритмы решения полиномиальной сложности, составляют *класс NP*.

Для многих задач класса *NP* выполняется свойство сводимости, состоящее в том, что данный алгоритм выражается при помощи полиномиального числа операций через другой алгоритм, имеющий полиномиальную сложность.

**5.2. Машина Тьюринга.**

До недавнего времени интуитивного понятия алгоритм было достаточно, и термин алгоритм употреблялся в связи с вычислительной процедурой решения конкретной задачи. Утверждение о существовании алгоритма решения задачи вытекало из описания этого алгоритма.

В начале XX века встал вопрос о выводимости и эффективных вычислениях. Понятие алгоритма стало объектом математических исследований и нуждалось в строгом определении. Возникли задачи, по-видимому, не имеющие алгоритмического решения.

Первые работы по уточнению понятия алгоритм появились в 1936 – 1937 годах. Это были работы Тьюринга, Поста, Маркова, Чёрча. Было предложено несколько определений понятия алгоритм. Впоследствии было показано, что все они равносильны.

Одной из первых и весьма удачных попыток дать точный математический эквивалент интуитивного представления об алгоритме было введение понятия *машины Тьюринга* в 1937 году, за 9 лет до появления первой ЭВМ.

Машина Тьюринга – абстрактная машина. Это математическая модель идеализированного вычислительного устройства.

Машина Тьюринга состоит из ленты и управляющего устройства со считывающей и записывающей головки (каретки) (рис. 5.1).

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

Рисунок 5.1.

Лента жестко закреплена слева и бесконечна справа. Иногда считают, что лента не ограничена справа и слева. Лента разделена на ячейки, которые нумеруются натуральными числами 1,2,….

В каждую ячейку ленты заносятся символы внешнего алфавита машины Тьюринга

*A*={*a*0,*a*1,...*an*}. (5.1)

Один из символов (пробел) соответствует незаполненной, пустой ячейке.

Головка может передвигаться вдоль ленты влево и вправо. Когда она неподвижна, то стоит против некоторой ячейки ленты; говорят, что головка обозревает эту ячейку.

За единицу времени, которая называется шагом, головка может сдвинуться на одну ячейку влево или вправо. Кроме того, головка может также распознать содержимое обозреваемой ячейки, может заносить символ внешнего алфавита в текущую ячейку и может стирать содержимое текущей ячейки или, что – то же самое, записывать туда пробел.

Управляющее устройство может находиться в одном из множества дискретных состояний:

*Q*={*q*0,*q*1,...*qm*}. (5.2)

Множество *Q* называется внутренним алфавитом машины Тьюринга или алфавитом внутренних состояний.

*Словом* называется последовательность *W*=*ai*1,*ai*2,…,*ais*символов, записанных в ячейках ленты, где *ai*1 – символ, находящийся в самой левой непустой ячейке, а *ais*– символ, находящийся в самой правой непустой ячейке. Количество символов *s* в слове называется *длиной слова*.

Пусть в некоторый момент времени *t* на ленте записано слово *W*, управляющее устройство находится в состоянии *qi*, акаретка – напротив символа *aim* слова *W*. *Конфигурацией* машины в момент времени *t* называется последовательность *K*=*ai*1,…,*ai*(*m*–1),*qi*,*aim*…,*ais*. Конфигурации в начале и в конце работы называют соответственно *начальной* и *заключительной*.

***Пример 5.4.***

Пусть на ленте записано слово *abcde*, управляющее устройство находится в состоянии *qi* и каретка стоит против символа *d*. Конфигурация в этом случае запишется так:

*abcqide*.

Так как машина Тьюринга имеет конечный алфавит и конечное число внутренних состояний, то очевидно, что она может выполнять конечное число действий.

Если управляющее устройство в некоторый момент времени находится в состоянии *qi*, обозревается символ *aj*, в следующий момент времени записывается символ *ar*, управляющее устройство переходит в состояние *qk*, и каретка сдвигается, то говорят, что машина выполняет команду

*ajqi*→*arSqk*, (5.3)

где *S* – сдвиг, *S*=*L*, если сдвиг влево, *S*=*R*, если сдвиг вправо, *S*=*C*, если каретка остается на месте.

Совокупность всех команд, которые может выполнить машина, называется ее программой. Условие однозначности требует, чтобы для любого *j* и любого *i* имеется только одна команда вида (5.3). Каждая машина Тьюринга полностью определяется своим алфавитом, внутренними состояниями и программой.

Итак, машина Тьюринга есть совокупность

*M*=<*A*,*Q*,*P*>, (5.4)

где *A* – внешний алфавит (5.1), *Q* – алфавит внутренних состояний (5.2),

*P* – программа (5.3).

***Пример 5.5.***

Машина с внешним алфавитом *A*={1,*a*}, алфавитом внутренних состояний *Q*={*q*1,*q*2} и программой

1*q*1→1*Rq*1,

*aq*1→1*Rq*1,

из любой начальной конфигурации будет работать бесконечно, заполняя единицами всю ленту вправо от начальной точки.

Порядок работы машины Тьюринга часто задается в виде таблицы.

В каждый столбец верхней строчки заносятся символы внутреннего алфавита, в каждую строчку первого столбца – символы внешнего алфавита. В ячейках на пересечении других столбцов и строчек помещаются команды.

Если на пересечении какой-либо строки и какого-либо столбца мы получим пустую клетку, то это означает, что в данном внутреннем состоянии данный символ встретиться не может.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *A*/*Q* | *q*0 | *q*1 | … | *qi* | *qn* |
| *a*0  *a*1  …  *aj*  …  *am* |  |  |  | *ajKqi* |  |

Формат команды: *aKq*, где:

*a* – новое содержание текущей ячейки (новый символ внешнего алфавита, который заносится в текущую ячейку);

*K* – команда лентопротяжного механизма машины Тьюринга (влево, вправо, стоп);

*q* – новое внутреннее состояние машины Тьюринга.

Работа машины на основании заданной программы происходит следующим образом.

Предположим, что в данный момент времени машина Тьюринга находится во внутреннем состоянии *qi*, а в обозреваемой кареткой ячейке ленты находится символ *aj*.

Тогда машина переходит к выполнению команды *ajKqi* в ячейке, на пересечении столбца *qi* и строки *aj*:

1) в текущую ячейку ленты заносится новый символ *aj*(возможно, тот же самый).

2) происходит сдвиг головки влево (*K*=влево), или сдвиг головки вправо (*K*=вправо), или головка остается на месте, т. е. происходит остановка машины (*K*=стоп).

3) машина переходит в новое внутреннее состояние *qi*.

Возможные случаи останова машины Тьюринга:

1) в ходе выполнения программы машина дойдет до выполнения команды остановки; программа в этом случае считается выполненной, машина останавливается – происходит результативная остановка.

2) машина никогда не останавливается, происходит зацикливание.

***Пример 5.6.***

Пусть внешний алфавит *A*={0,1,2}, а множество внутренних состояний состоит лишь из одного состояния *Q*={*q*0}. Необходимо построить машину Тьюринга, которая в произвольной записи, начиная из любой ячейки, двигаясь вправо, находит первый нуль и останавливается. Такая машина может быть задана таблицей 5.1.

Таблица 5.1.

|  |  |
| --- | --- |
| *a* | *q*0 |
| 0  1  2 | 0*Cq*0  1*Rq*0  2*Rq*0 |

Действительно, пусть, например, вначале машина находится в состоянии

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 1 | 1 | 2 | 0 | 1 | 2 | 2 |

Рисунок 5.2.

Головка обозревает символ 1. В соответствии с табл. 5.2 выполняется команда 1*Rq*0, т. е. в обозреваемую ячейку записывается тот же самый символ 1 и головка смещается вправо (рис 5.3).

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 1 | 1 | 2 | 0 | 1 | 2 | 2 |

Рисунок 5.3.

Теперь головка снова обозревает символ 1 и в соответствии с табл. 5.2 выполняется команда 1*Rq*0, т. е. т. е. в обозреваемую ячейку записывается тот же самый символ 1 и головка смещается вправо (рис 5.4).

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 1 | 1 | 2 | 0 | 1 | 2 | 2 |

Рисунок 5.4.

Теперь головка обозревает символ 2 и в соответствии с табл. 5.2 выполняется команда 2*Rq*0, т. е. т. е. в обозреваемую ячейку записывается тот же самый символ 2 и головка смещается вправо (рис 5.5).

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 1 | 1 | 2 | 0 | 1 | 2 | 2 |

Рисунок 5.5.

Теперь головка обозревает символ 0 и в соответствии с табл. 5.2 выполняется команда 0*Cq*0 т. е. в обозреваемую ячейку записывается тот же самый символ 0 и машина останавливается.

***Пример 5.7.***

Построим машину Тьюринга, которая слово А*B*) преобразует в слово А&*B*, а слово А&*B*) преобразует в слово А*B*, что соответствует законам де Моргана. Такая машина может быть задана таблицей 5.2.

Внешний алфавит *A*={А,*B*,,&,(,),\_} (символ \_ соответствует пустой ячейке), а множество внутренних состояний состоит лишь из одного состояния *Q*={*q*0}.

Таблица 5.2.

|  |  |
| --- | --- |
| *A* | *q*0 |
|   *A*  *B*    &  (  )  \_ | \_*Rq*0  *ARq*0  *Rq*0  &*Rq*0  *Rq*0  *Rq*0  *BRq*0  \_*Cq*0 |

Данные машины Тьюринга – это слова во внешнем алфавите ленты. На ленте записывается и исходные данные и конечный результат. На ленте могут быть записаны слова, а также последовательности слов. В последнем случае между словами ставится специальный символ-разделитель, им может быть пробел или символ \*.

Натуральное число *a* представляется словом 1…1= 1*a*, состоящим из *a* единиц. Например, числу 3 соответствует слово 111.

***Пример 5.8.***

Построим машину Тьюринга, которая производит сложение двух натуральных чисел *a* и *b*. Сложить два числа *a* и *b* – это значит слово 1*a*\*1*b* преобразовать в слово 1*a*+*b*. Это можно сделать, удалив в записи *a*\**b* символ разделителя (\*) и сдвинув первое слагаемое ко второму. Такая машина может быть задана таблицей 5.3. Внешний алфавит *A*={1,\*,\_}, где “\*” – символ разделителя, а \_ – символ пустой ячейки (пробел). Множество внутренних состояний состоит из трех состояний *Q*={*q*0,*q*1,*q*2}.

Таблица 5.3.

|  |  |  |  |
| --- | --- | --- | --- |
| *A* | *q*0 | *Q*1 | *q*2 |
| 1  \*  \_ | \_*Rq*1  \_*Rq*1 | 1*Rq*1  1*Lq*2  \_*Cq*1 | 1*Lq*2  \_1*Rq*1 |

Начальное и конечное состояния ленты для случая *a*=2, *b*=3 представлено на рис. 5.6 *a*) и *b*)

*a*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 1 | 1 | \* | 1 | 1 | 1 |  |

*b*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | 1 | 1 | 1 | 1 | 1 |  |

Рисунок 5.6.

**5.3. Вычислимые по Тьюрингу функции.**

Будем рассматривать функции *f* от одной или нескольких переменных, заданных на множестве *N*={0,1,2,…,*n*,…} натуральных чисел или его подмножествах (частичные функции) и принимающие значения на множестве *N*.

***Определение 5.8.*** Функция *f*(*x*1,*x*2,…,*xn*) называется *вычислимой*, если существует алгоритм, позволяющий вычислять ее значения для тех переменных, для которых она определена, и работающий бесконечно, если функция для данного набора переменных не определена.

***Определение 5.9.*** Функция *f*(*x*1,*x*2,…,*xn*) называется *вычислимой по Тьюрингу*, если существует машина Тьюринга, вычисляющая эту функцию.

Переменные можно располагать в виде слов с разделителями

\*11…1\*11…1\*……\*11…1\*

***Пример 5.9.***

Запись \*111\*11\*1\* соответствует трем переменным *x*1,*x*2,*x*3, равным, соответственно, 3, 2 и 1

Функция также записывается словом, состоящим из единиц.

Пример 5.8 представляет функцию двух переменных *f*(*a*,*b*)=*a*+*b*.

***Тезис Тьюринга.*** Всякий алгоритм можно реализовать машиной Тьюринга.

Тезис Тьюринга доказать нельзя. Это утверждение означает, что математическое понятие вычислимой по Тьюрингу функции является идеальной моделью интуитивного понятия алгоритма. Этот тезис подтверждается опытом. По своему характеру тезис Тьюринга напоминает математические законы механики, которые точно так же не могут быть доказаны, но, открытые Ньютоном, многократно подтверждены опытом. В силу тезиса Тьюринга невозможность построения машины Тьюринга означает отсутствие алгоритма решения данной проблемы.

Изучение машин Тьюринга закладывает фундамент алгоритмического мышления, сущность которого состоит в том, что нужно уметь разделять процесс вычисления на простые составляющие шаги. В машине Тьюринга такое разделение доведено до предельной простоты. В современной ЭВМ алгоритмический процесс разделяется не на столь мелкие составляющие, как в машине Тьюринга. Наоборот, есть стремление укрупнить выполняемые машиной процедуры. Например, операция сложения в машине Тьюринга – целая программа, а в ЭВМ это простейшая функция.