Как алгоритмы формируют наш мир - Видео(Ютубе)

Шаповалов Сергей

2015

Зміст

Как алгоритмы формируют наш мир 5
Раздел 1 5
Тема 8 5
Раздел 2 5
Тема 10 5
Основные определения, свойства и способы задания 5
10.1. Понятие алгоритма. Эволюция толкования и характеристики 6
10.2. Способы задания алгоритмов 7
Розділ 1 Класична математична логіка 8
Тема 1 8
Ключові терміни: 8
Основные понятия логики высказываний 8
1.1. Основные определения 8
1.2. Высказывания и логические связки 9
1.3. Условные и эквивалентные высказывания 10
1.4. Интерпретация формул логики высказываний 11
1.5. Проблема решения в алгебре высказываний. 12
Функциональная полнота множества логических операций 12
1.6. Дедуктивні висновки у логіці висловлювань 12
Тема 2 15
Числення висловлювань 15
2.1. Формальна аксіоматична теорія L 15
2.2. Теорема дедукції 16
2.3. Побудова доведень у логіці висловлювань 17
2.4. Аксіоматичний метод 18
2.5. Конструктивний метод 19
2.6. Метод доведення від супротивного 20
2.7. Метод резолюцій 20
Тема 3 22
Логіка предикатів першого порядку 22
3.1. Основні поняття логіки предикатів 22
3.2. Квантори 23
3.3. Формули логіки предикатів 23
3.4. Рівносильність формул логіки предикатів 24
3.5. Закони і тотожності логіки предикатів 24
3.6. Властивості числення предикатів першого порядку 25
Тема 4 27
Аксіоматичні системи логік 27
4.1. Система аксіом і правил виведення 27
4.2. Випереджені нормальні форми 28
4.3. Побудова доведень в аксіоматичній системі 29
4.4. Метод ідентифікації 30
4.5. Метод резолюцій 30

Тема 5 Некласична математична логіка 31
Нечітка логіка 31
5.1. Основні визначення 31
5.2. Операції над нечіткими множинами 33
5.3. Поняття нечіткої і лінгвістичної змінної 33
5.4. Нечіткі висловлювання та нечіткі предикати 34
5.4.1. Основні логічні операції з нечіткими висловлюваннями 34
5.5. Правило нечіткого логічного виводу 35
5.5.1 Композиційне правило нечіткого виводу Заде 35
Запитання для самоперевірки: 35
Тема 6 36
Модальна логіка 36
6.1. Особливості побудови модальних систем 36
6.2. Семантика Кріпке 37
6.3. Теорія “двійників” К. Льюїса 37
Запитання для самоперевірки: 38
Тема 7 38
Епістемічна логіка 38
7.1. Основні визначення 38
7.2. Оператори знання й переконання, їх властивості 39
7.3. Формальна мова епістемічної логіки 39
Запитання для самоперевірки: 39
Тема 8 39
Некласична математична логіка 40
Деонтична логіка 40
8.1. Основні визначення 40
8.2. Синтаксис та семантика деонтичної логіки 40
Запитання для самоперевірки: 40
Тема 9 40
Інтуїціоністська логіка 41
9.1. Числення висловлювань в інтуїціоністській логіці 41
9.2. Доведення формул числення висловлювань в інтуїціоністській логіці висловлювань 41
9.3. Застосування моделі Кріпке в інтуїціоністській логіці 41
Розділ 2 Елементи теорії алгоритмів 42
Тема 10 42
Основні визначення, властивості та способи задання 42
10.1. Поняття про алгоритм. Еволюція тлумачення та властивості 42
10.2. Способи задання алгоритмів 44
Задачі для самостійного розв’язання 44
Тема 11 45
Алгоритми та обчислювальні функції 45
11.1. Основні означення 45
11.2. Оператор суперпозиції 45
11.3. Оператор примітивної рекурсії 45
11.4. Оператор мінімізації 46
11.5. Гіпотеза Черча та примітивно-рекурсивні функції 46

Запитання для самоперевірки 46
Задачі для самостійного розв’язування 47
Тема 12 47
Алгоритмічні моделі 47
12.1. Основні визначення 47
12.2. Машини Тюрінга 47
12.3. Нормальні алгоритми Маркова 48
Задачі для самостійного розв’язування 49
Тема 13 49
Нумерації алгоритмів 49
13.1. Попередні відомості 49
13.1.1. Нумерація за Гьоделем 50
13.1.2. Головні універсальні функції та множини 50
13.1.3. Канторові нумерації кортежів натуральних чисел 51
13.1.4. Нумерації Кліні та Поста 51
13.2 Нумерація машин Тюрінга та частково - рекурсивних функцій 52
13.3. S-m-n- теорема 52
13.4. M – зведення та властивості злічених множин 52
13.5 Теореми Кліні про нерухому точку 53
Задачі для самостійного розв’язання 53
Тема 14 53
Складність алгоритмів 53
14.1. Асимптотичні оцінки складності 54
14.2. Класи складності 54
Задачі для самостійного розв’язання 56
Тема 15 56
NP - повні, складні та алгоритмічно нерозв’язні проблеми 56
15.1. Розв’язні та нерозв’язні проблеми, NP- повнота, складність, зведення 56
15.1.2. Приклади NP-повних задач 57
15.2. Алгоритмічно нерозв’язні проблеми 58
Запитання для самоперевірки: 59
Задачі для самостійного розв’язання 59

- Как алгоритмы формируют наш мир

Как алгоритмы формируют наш мир

http://youtu.be/kvWkcUHnzyQ

Раздел 1

Тема 8

Просмотреть
Скачать оригинал
Скачать PDF

Раздел 2

Тема 10

- Основные определения, свойства и способы задания
  - 10.1. Понятие алгоритма. Эволюция толкования и характеристики
  - 10.2. Способы задания алгоритмов

Основные определения, свойства и способы задания

А существует ли алгоритм любви?
(риторический вопрос)

Понятие алгоритма принадлежит не только к фундаментальным научным понятиям, но и к человеческим приобретениям. Мы в окружении разнообразия алгоритмов во всех сферах жизни и деятельности. Наши действия во многом доведены до автоматизма и мы неосознаем, что они уже регламентированы тем или иным алгоритмом. Так жизнь каждого из нас состоит из выполнения повседневных алгоритмов, начиная с усвоения алгоритмов в детстве, от складывания пирамидки, и заканчивая поиском алгоритмов для решения научных проблем. Вообще любую человеческую деятельность можно подать в виде алгоритмического процесса. Авторы обзора основных открытий теории алгоритмов [1] утверждают: "Алгоритмические концепции играют в процессе обучения и воспитания современного человека фундаментальную роль, в сравнении лишь с ролью письменности".

На одной из научных конференций по информатике (1984 г.) академик А. А. Самарский изобразил на слайде информатику в виде красавицы, что несется по научному океану на трёх китах. Имена этих китов: Модель, Алгоритм, Программа.

Центральный кит - теория алгоритмов, соединившись с математической логикой, составляют теоретический фундамент современных компьютерных наук.

И хотя интуитивно понятие алгоритма понятно, наверно, каждому, формально определить этот объект в нетривиальной теории алгоритмов совсем не так просто, как кажется на первый взгляд.

Во-первых, настораживает чрезмерная популярность этого понятия и, как в таких случаях говорят, его расширенное разъяснение. Так, вам могли встретиться такие различные алгоритмы как: “алгоритм Эвклида“, “алгоритм шахматиста“, “алгоритм поиска информации“,

“алгоритм, как стать красавицей” и другие.

Во-вторых, алгоритмы, как сами по себе, суть объекты специфического типа, которые тяжело описать математически. Они имеют нетиповое для математических объектов свойство, а именно семантическое свойство “иметь содержание“. В этом отношении теория алгоритмов подобна математической логике. С семаническими объектами, которые несут в себе содержание, математики, еще не привыкли работать. Смысл **терма** или **формулы** в математической логике “указательный“: терм указывает на вещь (т.е.помечает), а формула - на факт. Смысл алгоритма “приказной“: алгоритм должен быть выполнен. Таким образом, теория, которая изучает алгоритмы, может трактоваться как своего рода лингвистика приказных пропозиций.

И, *в-третьих*, теория алгоритмов возникла на стыке математики и информатики. Поэтому ученые с двух сторон “тянули на себя одеяло“ при построении теории алгоритмов. Эту специфику можно заметить, изучая монографии и пособия по теории алгоритмов. В зависимости от того, кем они написаны - математиками или знатоками информатики, мы чувствуем особенности содержания материала, отображенного в них [1-5].

Сейчас слово “алгоритм“ как научный термин вышло за границы математики. Этот термин используется в самых различных областях науки и техники, понимая при этом точно сформулированное правило, назначение которого – руководить действиями для достижения необходимого результата.

10.1. Понятие алгоритма. Эволюция толкования и характеристики

Понятие “ **алгоритм** “ - концептуальная основа различных процессов в информатике. Именно наличие алгоритмов и обеспечивает мозможность автоматизации. Больше того, значительной мерою теория алгоритмов домогает проникать математическим методам во все сферы жизни и науки, даже биологию, лингвистику,экономику аж до философии природоведения [1-12].

Понятие алгоритма формировалось с давних времен и до тридцатых годов XX ст., в научных кругах сложилось интуитивное представление про этот объект.

Одним из самых давних алгоритмов, в его интуитивном понимании, как конечной последовательности элементарных действий, которые решают поставленное задание, считается алгоритм наибольшего общего делителя двух чисел (алгоритм Эвклида). Отметим, что в течении продолжительного времени, аж до начала XX столутия, термин “алгоритм“ употреблялся только в стойком словосочетании “алгоритм Эвклида“, хотя исторически установлено, что задолго до Эвклида этот алгоритм был уже известен, например, Аристотелю.

Содержательные явления, которые привели к появлению понятия “алгоритм“, наблюдаются в математике на протяжении всего периода его существования. Именно слово “алгоритм“, как утверждают историки и математики, появилось несколько столетий тому и означало не термин, а имя. Узбекский математик **Аль-Хорезми**, учений, которому математика и человечество обязаны многими открытиями, был известен европейским математикам в латинской транскрипции как Алгоризми. А полное его имя - Абу Абдулла Абу Джафар Мухаммад ибн Муса аль-Хорезми (около 780 г. — около 850 г.) - в переводе буквально - отец Абдулли, Мухаммед, син Муси, уродженець Хорезми. Его книга - “Хисаб аль-джабр у аль-мукабаля“ . Именно из трактата Аль-Хорезми из арифметики началось знакомство Европы с алгоритмами - строгими процедурами для выполнения арифметичних операций. То есть алгоритм, точнее алгоризм, понимали как руководство для выполнения заданий.

Дальнейшее развитие математики утвердило мысль, что решение какой-либо проблемы должно быть алгоритмичным. Р. Декарт, Г. Ф. Лейбниц, Д. Гильберт стимулировали алгоритмические исследования. В 1900 на Втором межнародном конгрессе математиков (Париж) Давид Гильберт сформулировал 23 важные математические проблемы, находжение алгоритма решения которых, по его мнению, способствовало бы дальнейшему развитию математики.

Именно в то время были распространены две точки зрения :

1. Все проблемы являются алгоритмично решимыми. Просто еще не существует знаний для построения алгоритма.
2. Существуют классы заданий, для решения которых вообще не существует алгоритмов. Это очень сильное утверждение, потому что оно распространяется на все будущее.

Для доказательства не существования алгоритма необходимо было иметь его точное математическое определение, поэтому после того, как сформулировали понятие алгоритма как новой и отдельной сущности, первоочередной стала проблема нахождение адекватных формальных моделей алгоритма и исследование их характеристик. Формальные модели были предложены как для первичного понятия алгоритма, так и для производного понятия алгоритмично вычислительной функции. Впервые алгоритм как термин появился в работах Э. Бореля (1912) и Г. Вейля (1921).

Начальной точкой отсчета современной теории алгоритмов можно считать работу немецкого математика Курта Гьоделя (1931 год - теорема про неполноту символьних логик), в которой было показано, что для любой непротиворечивой системе аксиом существует утверждение, которое в рамках принятой аксиоматичной системы не может быть ни доказано, ни опровергнуто. Возникшее в связи с этой теоремой предположение про невозможность алгоритмического решения многих математических проблем (в том числе, проблемы выведения в вычислении предикатов) вызвало необходимость уточнения понятия алгоритма.

Первые фундаментальные работы по теории алгоритмов были опубликованы независимо в 1935-37 годах А. Тюрингом, А. Черчем, С. Клини, К. Гьоделем и Е. Постом. Предложенные ими машина Тюринга, машина Поста, частично рекурсивные функции и лямбда-вычисления Черча були эквивалентными формализмами алгоритма. Сформулированные ими тезисы (Поста и Черча-Тюринга) постулировали эквивалентность предложенных ими формальных систем и интуитивного понятия алгоритма. Важным развитием этих работ стало формулирование и доказательство алгоритмически нерешаемых проблем. В 1950-ые - годы существенных вклад в теорию алгоритмов сделали работы

А. М. Колмогорова и А. А. Маркова (младшего), а в 1970-х годах в теории алгоритмов плодотворно работал Ю. В. Матиясевич, в частности над решением проблем Гильберта.

Предложенное А. Марковым понятие “нормального алгоритма“ является величичайшим вкладом в мировую науку, а А. Колмогорову принадлежит приоритет в определении общего понятия алгоритма и созданияя теории сложностиі конструктивных объектов.

В 1960-70-ые годы сформировались такие направления в теории алгоритмов:

- **Класическая теория алгоритмов** (изложение задания в терминах формальных языков, понятие задачи решения, введение классов сложности, формулирование в 1965 году Эдмондсом проблемы P=NP, открытие класса NP-полных задач и их исследование)[1].
- **Теория ассимптотического анализа алгоритмов** (понятие сложности и трудоемкости алгоритма, критерии оценки алгоритмов, методы получения ассимптотических оценок, а именно для рекурсивных алгоритмов, ассимптотический анализ трудоемкости или время выполнения), в розвитии которой сделали существенный вклад Кнут, Ахо, Хопкрофт, Ульман [2,4].
- **Теория практичного анализа вычислительных алгоритмов** (получение явных функций трудоемкости, практические критерии качества алгоритмов, методика выбора рациональних алгоритмов), основополагающими работами в этом направлении, очевидно, следует считать фундаментальные труды [1,2].

С тех пор многие ученые приложились к развитию теории алгоритмов. Необходимо отметить также весомый вклад в теорию алгоритмов, сделанный Д. Кнутом, А. Ахо и Дж. Ульманом. Необходимо вспомнить и другие издания книги “Алгоритмы: построение и анализ“ Томаса Х. Кормена, и труды Чарльза І. Лейзерсона, Рональда Л. Ривеста, Клиффорда Штайна.

Ученые, которые упорядочили теорию алгоритмов, рано или поздно сталкивались с тем, что необходимо определить объект исследований. Оказалось, что все разнообразные определения, поданные ими относительно алгоритма, являются эквивалентными понятиями, и чудесным научным результатом есть доказательство эквивалентности этих формальных определений в содержании их равнозначности. Приводя все определения, как говорят математики, “к общему знаменателю“, можно дать общее определение алгоритма:

Алгоритм – способ превращения информации по правилам, сформулированным определенным языком.

Несмотря на усилия исследователей, отсутствует единственно исчерпывающее строгое определение понятия алгоритма. Все они являются интуитивными, точно также и те, котрые приведены выше. Интуитивно говоря, чтобы получить решение какой-либо задачи необходимо выполнить действия по строго определенному формальному предписанию. Это формальное предписание и есть алгоритмом.

В общем случае понятно, что алгоритм – это детерминированная процедура, которую можно применить к любому элементу определенного класса символьных входов и которая для каждого такого входа выдает через конечное число действий (шагов) соответствующий результат свого действия. Пусть D - область (множество) исходных данных задачи, а R - множество возможных результатов, тогда мы можем говорить, что алгоритм A осуществляет отображение A : D → R.

Любой алгоритм должен иметь такие основные особенности или характеристики:

Определенность (однозначность) - каждая команда алгоритма однозначно определяет действия исполнителя и не допускает двойного толкования.

Дискретность – возможность разбиение алгоритма на конечное количество этапов, причем результаты предыдущего этапа есть входными для последующего.

Массовость – алгоритм может быть использован для решения целого класса задач одного типа, для которых он создан.

Понятность – алгоритм должен быть понятным конкретному исполнителю, который должен исполнить каждую команду алгоритма в строгом соответствии с предназначением.

Результативность (сходимость) – исполнение алгоритма должно оканчиваться результатом или информацией о том, что не может быть полученным результат.

Именно этим характеристикам часто дается интуитивное определение алгоритма как конечной однозначно определенной последовательности операций, формальное выполнение которой приводит к решению поставленой задачи за конечной число шагов.

Пример 10.1. Рассмотрим алгоритм Эвклида нахождение наибольшего общего делителя двух натуральных чисел *а* и *в* (НСД (а, в)). Этот алгоритм перерабатывает слова вида (*а,в*) в алфавите, который состоит из 10 цифр, " (" и")", знака пунктуации " , ", в слова того же самого алфавита (результатом будет слово из цифр). Конечная система правил, которая определяет этот алгоритм, будет складываться из правила деления, определение остатка от деления и правила определенности, когда алгоритм заканчивает работу и выдает результат.

Пример 10.2. Рассмотрим процес приготовления блюда быстрого питания : 1. Высыпать в пустую посуду содержимое пакетика. 2. Влить в емкость 200 *мл* горячей воды. 3. Тщательно перемешать. Чтоб выполнить конечную систему правил по этому "алгоритму», необходимо проделать чисто механические действия.

Хотя последний пример не является алгоритмом в том смысле, который мы определили. Поскольку он не отвечает всем требованиям его характеристик, дадим ему определение "почти" алгоритм, точно также как и медицинским рецептам. Подумайте почему? Именно такое разделение.

Понятно, что *алгоритм* как сущность пришел к фундаментальным достижениям человечества из математики, а в современности получил статус центральной оси программирования и информатики в целом. По своей природе алгоритм – это интелектуальный продукт, который является найденным его создателем последовательностью действий над входной информацией, строгое выполнение которых приведет к желаемому или не желаемому результату. Алгоритм существует тогда и только тогда (или начинает существовать), когда существует в тоже самое время какой-то объект (чаще всего математическая его модель) или проблема, для решения которой он и создан.

По назначению алгоритм с входными данными сопоставляет выходные данные и в этом он чем-то схож с позиций математики на обычную функцию, а с позиций информатики - на способ преобразования информации. Однако главной особенностью алгоритма есть то, что он содержит описание именно лействий, как это делать. Функция может буть задана неявно, а алгоритм - нет. Алгоритм описывает одним из возможных способов своего представления(см. п.10.2), что необходимо сделать с исходными данными, чтобы получить результат.

Данное выше толкование алгоритма нельзя считать строгим в научном смысле, так как не совсем понятно, что такое "последовательность действий, которое обеспечивает получение необходимого результата". Поэтому обычно формулируют несколько общих свойств(характеристик) алгоритмов, что разрешает отличать алгоритмы от других подобных инструкций (см. п.10.1). Однако даже знание этих свойств не дает возможность определить, являются ли данные инструкции алгоритмом, или нет. Наверно, одним из определяющих для решения этой проблемы, является свойство однозначности выполнения каждой команды, данной в инструкции. Если результат выполнения каждого шага инструкции происходит однозначно и независимо от исполнителя, то эти инструкции составляют именно алгоритм.

10.2. Способы задания алгоритмов

Если взять за основу, что алгоритмы с позиции математики являются строгими правилами решения какой-либо задачи, а с позиции информатики являются центральным звеном в цепи модель-алгоритм-программа (см. выше), то правильность алгоритмов есть залогом надежного функционирования компьютерных программ, достоверности результатов в решении научных и жизненных проблем и вообще адекватного мировосприятия. Короткую простую программу можно написать непосредственно исходя из постановки задания без предварительной разработки алгоритма. Однако реальную программу так же трудно создать без алгоритма, как трудно изготовить достаточно сложную деталь без чертежа.

В этом смысле, кроме правильности самих алгоритмов по сути решение самой проблемы, необходимо иметь удобное и понятное их представление для практического применения конкретному исполнителю. При задании алгоритма необходимо позаботиться про то, чтобы алгоритм воспринимался всеми исполнителями однозначно и точно, чтобы его можно было выполнить при любых допустимых исходных условиях, и чтобы необходимый результат был получен за приемлемое время. Очевидно, что предписание "Пойди туда, не знаю куда, принеси то, не знаю что" алгоритмом не является.

Существует несколько методов записи или представления алгоритмов. Выбор метода зависит от исполнителя и представленных разработчиком вариантов представления алгоритмов.

Перший способ задания алгоритмов - это словесно-формульный (описание осуществляется в словесной форме с использование математических или других формул). Он описывает выполнение действий алгоритма в определенной последовательности с помощью слов и предложений естественного языка. Форма изложения произвольная и устанавливается разработчиком. В математике наличие формул позволяет решить задачу, даже "не используя слов".

Пример 10.3. *Записать алгоритм нахождения наибольшего общего делителя (НОД) двух натуральных чисел (алгоритм Эвклида).*

Решение. Алгоритм может быть представлен таким образом:

1. Возьмите два числа.
2. Если числа однаковые, то взять любое из них как ответ и остановиться, иначе продолжить выполнение алгоритма.
3. Определить наибольшее число.
4. Заменить большее из чисел разницей большого от меншого числа.
5. Повторити алгоритм с шага 2.

Другой способ задания алгоритмов - это представление алгоритма в виде таблиць, формул, схем, рисунков и т.д. Например, всех нас учили в детстком саду правилам поведения на дороге. И некоторые дети, очевидно, лучше воспринимают этот алгоритм, в виде схематического рисунка. Смотря на них, ребенок, а потом и взрослый человек, воспроизводять ту линию поведения, которую им предлагают.

Третий способ задания алгоритмов - это запись алгоритмов с поможью блок-схемы. Этот метод был предложен в информатике для наглядности представления алгоритма с помощью набора специальных блоков. Блок-схема – это такое графическое представление алгоритма, при котором решение задачи изображается в виде разных геометрических фигур, в середине которых записаны тексты, что вместе с формой фигуры объясняет и действия, которое необходимо выполнить, и взаимосвязи между ними.

Схема алгоритма четко определяет последовательность действий, заданных этим алгоритмом. Согласно характеристике дискретности схема может представлять собой алгоритм из разных ступеней детализации. При этом важно более точно и наглядно изображать управляющие структуры алгоритма, которые показывают логику перехода от одного действия к другому. Однако требования к графическому исполнению схем вызывают предубеждение некоторых программистов, представляются основным недостатком схем и заставляют искать новые способы записи алгоритмов.

Пример 10.4. *Используя необходимые для представления блоки, можно представить, например, алгоритм "Выхода на улицу" в таком виде:*

Четвертый способ задания алгоритмов – это запись алгоритма учебным алгоритмическим языком (псевдокодом).

Псевдокодом называется система правил записи алгоритма, с использованием набора определенных конструкций, для описания управляющих действий. Псевдокод является основным способом представления алгоритмов для теоретических исследований в теории алгоритмов. Именно на этих конструкциях отрабатывается, например, нахождение асимптотических оценок алгоритмов.

Единого, или формального, определения псевдокода не существует, поэтому возможны различные псевдокоды, отличающиеся набором служебных слов и основных (базовых) конструкций. В отличии от стандартизации синтаксиса языков программирования, на синтаксис псевдокода обычно не устанавливается стандартов, поскольку последний непосредственно не компилируется в исполнительную программу. Поэтому можно сказать, что обычно автор каждой публикации применяет свой оригинальный псевдокод, заимствуя нужные ему

конструкции из любого языка программирования. Их использование можно объяснить как индивидуальными симпатиями автора, так и распространенностью на момент написания публикации. В случае русскоязычных публикаций как псевдокод, часто используется перевод ключевых слов языков программирования с англійського на русский.

В ряде случаев **псевдокодом** называють систему команд абстрактной машины, например, Р-код, псевдокод придуманной машины MIX и т. д. В отличии от псевдокода неформального характера такой псевдокод уже строго формализованый и может быть транслирован в работающую программу при наличии программы-эмулятора данной гипотетической машины.

Известны прогнозы, которые утверждают, что дальнейшее развитие языков программирования пойдет путем их сближения с псевдокодом, что на конечном этапе позволит осуществлять программирование на естественных языках.

Пример 10.5. Записать на псевдокоде алгоритм Евклида – нахождение наибольшего общего делителя двух чисел.

Решение.

```
1 функция нод(a, b)
2     если b = 0
3         верни a
4   иначе
5         верни нод(b, a mod b)
```

Вы, наверно, заметили, что этот вариант алгоритма отличается от алгоритма, представленного во время решения примера 10.1. На самом деле существует несколько вариантов написания алгоритма Евклида, в данном случае записан в псевдокоде рекурсивный вариант.

Пятый способ, максимально приближен к компьютеру, задания алгоритмов - это языки программирования. Дело в том, что наиболее часто на практике исполнителем созданного человеком алгоритма является машина, и поэтому он должен быть написан языком, понятным для компьютера, т.е. языком программирования. Язык программирования - это знаковая система, предназначена для описания процессов решения заданий и их реализации на ЭВМ. Реализация обозначает, что описания могуть быть введены в ЭВМ и однозначно ею поняты. К языкам программирования принадлежат языки команд или машинные языки и языки высокого уровня.

Вопросы для самоконтроля:

1. Сформулируйте основные свойства(характеристики) алгоритмов.
2. Укажите направления развития теории алгоритмов.
3. Какие способы представления алгоритмов существуют?
4. Что есть псевдокод?
5. Сформулируйте понятие “массовость» в теории алгоритмов.

Розділ 1 Класична математична логіка

Тема 1

- Основные понятия логики высказываний
  - 1.1. Основные определения
  - 1.2. Высказывания и логические связки
  - 1.3. Условные и эквивалентные высказывания
  - 1.4. Интерпретация формул логики высказываний
  - 1.5. Проблема решения в алгебре высказываний.
    - Функциональная полнота множества логических операций
  - 1.6. Дедуктивні висновки у логіці висловлювань

Ключові терміни:

Атом (элементарное высказывание), Высказывание, Значенние истинности, Интерпретацией высказываний, Логика высказываний, Однозначно определенным высказыванием, Правила построения формул, Проблема решения, дизъюнкцией высказываний, импликация (условное предложение), отрицание, эквиваленция

Основные понятия логики высказываний

1.1. Основные определения

Логика как самостоятельная наука возникла в IV веке до н.э. в трудах Аристотеля, который опираясь на накопленные до него знания, дополнил их собственными и создал систему **формального логического вывода**, которая заключается в том, что в рассуждениях одни предложения исходят из других через определенную связь между их формой и структурой независимо от их содержания.

Революционные научные волнения конца XIX - начала XX века затронули и логику Аристотеля, путем реализации идеи Г. В. Лейбница , предложенной им еще в конце XVII века , о применении в логике математической символики и построений логических исчислений . Эта идея реализована в работах Д. Буля , Ч.С. Пирса, Ф. Л. Г. Фреге , наряду со многими исследованиями других ученых.

Классическая математическая логика включает два основных раздела: логику высказываний и логику предикатов. Для их построения существуют два подхода (языка) , на основе которых основаны два варианта формальной логики: алгебра логики и логическое исчисление . Между основными понятиями этих языков имеет место взаимно однозначное соответствие, но строго говоря эти термины не синонимы.

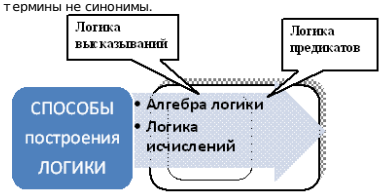


Рисунок 1.1 – Составные части классической математической логики

По своей сути логика высказываний - это наука про размышления, предпосылки и выводы которых складываются из высказываний.

Определение 1.1.1. Высказыванием называют осмысленное выражение обычного языка, которому можно приписать значения истинности.

Таким выражением может стать утверждение или повествовательное предложение, о котором можно сказать, истинное оно или ложное. Следует иметь в виду, что в логике высказываний нет средства, чтобы установить истинность или ложность простого высказывания. Если истинность и ложность нельзя установить вообще (то есть с помощью других наук), то такое высказывание не рассматривается (например: указательные высказывания, бессмысленные утверждения).

Определение 1.1.2. Значенние истинности – это абстрактный объект, которой ставится в соответствии с высказыванием: истина – когда высказывание отвечает действительности,

ложь – когда высказывание не отвечает действительности.

Обозначают : "*Истина* " – *И*, *T* (True), или 1; "*Ложь*" – Л, F (False) або 0.

Пример 1.1.1. Определить какие из данных предложений являются выражениями:

"*Днепр впадает в Черное море* "; "*Днепр впадает в Азовское море* "; "*Кто вы?*";

"*Расстояние от Земли до Сонця равняется 150 млн км* ".

Решение. Первые два предложения являются высказываниями, причем первое является истинным, а второе – ложным высказыванием. Третье предложение не является высказыванием, поскольку оно не повествовательное. Четвертое предложение также не является высказыванием. Его истинность или ложность зависит от определенной точности.

Классическая логика высказываний оперирует только двумя значениями истинности: *И* и *Л,* но не одновременно одним и другим, поэтому ее называют двузначной или бинарной логикой.

Повествовательные предложения могут быть простыми и сложными. Каждое простое предложение может быть утверждением и не может быть разбито на более мелкие предложения.

Определение 1.1.3. Атомом (элементарным высказыванием) называется такое высказывание, которое является простым повествовательным предложением, т.е. не имеет составных частей.

Для обозначения атомо используют, как символы, буквы латинского алфавита с индексами или без них.

Сложные предложения, как правило, состоят из простых предложений, соединенных союзами. То есть простые предложения, которые представляют атомы и союзы, являются элементами словаря, необходимого для формализации естественного языка с помощью логики высказываний. Значение истинности сложного высказывания определяется значениями истинности его составных частей.

Алгебра высказываний полностью абстрагируется от смыслового значения высказываний, принимая во внимание только их предметное значения, то есть денотат, которым выступают такие абстрактные объекты, как "истина" и "ложь".

Определение 1.1.4. Интерпретацией высказываний называют приписывание значений истинности атомам, из которых построены высказывания.

Если высказывания содержат *n* атомов, то можно составить 2^n интерпретаций.

Определение 1.1.5. Однозначно определенным высказыванием называют высказывания, значение истинности которого не зависит от ситуации. Например, "$3 \times 3 = 9$" = *И*. Но существуют высказывания, которые могут принимать разные значения. Например, "*Завтра будет снег*", можно придавать значение "*Истина* " и "*Ложь* " в зависимости от конкретной ситуации.

1.2. Высказывания и логические связки

Определение 1.2.1. Логика высказываний это алгебрагическая структура

$\langle \{X, I\}, \wedge, \vee, \neg, \rightarrow, \sim, X, I \rangle$ с носителем – двоичным множеством { *Л* : "*Ложь*", И : "*Истина*"}, операциями – логическими связками: " $\wedge$ " - конъюнкция, " $\vee$ " - дизъюнкция " ¬ "– отрицание, " $\rightarrow$" – импликация, " ~ " – эквивалентность и константами : *Л* – ложь и И – истина.

В обычном языке для образования сложного предложения из простых предложений используют служебные слова – связки : " *и* ", " *или* ", " *неправильно, что*" и другие. Например два предложения " *Я поеду летом к морю* " и " *Я поеду летом в горы* " можно объединить связкой " *или* " в одно сложное предложение " *Я поеду летом к морю или в горы* ". Здесь связку " *или* " нельзя присоединить ни к первому, ни ко второму простому предложению, она обслуживает одновременно оба простых предложения и поэтому называется **бинарной**. Например, в предложении "*Неправильно, что жителей в Киеве меньше, чем во Львове* " происходит противоречие

" *В Киеве меньше жителей, чем во Львове* ". Связка " *неправильно, что …* " является унарною, потому что применяется к одному предложению. Кроме рассмотренных, существуют связки : " *если* ", " *если … то* ", " *и* ", " *тогда* ", "*… тоогда и только тогда* ", " *а* ", " *нет* " и другие.

Логические связки можно рассмотреть как формальные обозначения связок, которые им соответствуют, табл. 1.2.1.

Таблиця 1.2.1

Название	Обозначения	Аналог естественного языка
отрицание	¬	" *нет*", " *неправильно, что*"
дизъюнкция	$\vee$	"или"
конъюнция	$\wedge$	"*и*"
импликация	$\rightarrow$	" *если …,то* "
эквивалентность	~ , $\leftrightarrow$, $\Leftrightarrow$	" *эквивалентно*", "*равносильно*", "*… тогда и только тогда* ", " если и только если"

Операции $\vee$, $\wedge$, $\rightarrow$, ~ являются бинарными логическими связками, а операция ¬ – унарной. Сложные высказывания, которые построены с простых высказываний, называют **формулами или молекулами**.

Определение 1.2.2. Правила построения формул в логике высказываний определяют следующим:

1. Атом есть формула.
2. Если *A* и *B* – формулы, то $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \sim B$, $\neg A$ – также формулы.
3. Никаких формул, кроме порожденных указанными выше правилами не существует.

Формулы логики высказываний, которые соответствуют сложным высказываниям, принимают значения *И* и *Л* в зависимости от значений элементарных высказываний и логических связок, из каких они построены.

Формулы логики высказываний удобно представлять таблицами истинности, которые представляют значение истинности формулы в зависимости от последовательного перебора все возможных значений истинности простых высказываний, которые составляют формулу (см. табл. 1.2.2.)

Таблиця 1.2.2

A	*B*	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \sim B$
Л	*Л*	*И*	*Л*	*Л*	*И*	*И*
Л	*И*	*И*	*Л*	*И*	*И*	*Л*
И	*Л*	*Л*	*Л*	*И*	*Л*	*Л*
И	*И*	*Л*	*И*	*И*	*И*	*И*

Из таблицы истинности следует, что отрицание $\neg A$ истинно тогда и только тогда, когда высказывание *A* ложно. Эта унарная операция отвечает отрицанию в обычном языке, которое может иметь разные синтаксические выражения, наприкмер, предложение " *Неправильно, что у Руслана есть машина* " равнозначно предложению " *У Руслана нет машины*".

Выражение $A \wedge B$ называют конъюнкцией высказывания *A* и *B*, которое истинно тогда и только тогда, когда истинны оба высказывания *A* и *B*. Эта логическая операция соответствует в естественной языке связке " *и* ", что соединяет два предложения.

Пример 1.2.1. Записать в виде формулы логики высказывания и определить истинное значение таких высказываний:

1. " 8 *делится на* 4 и 8 *больше* 6 ";
2. " 8 *делится на* 4 и *7 больше 8* ".

Решение. В данных высказываниях выделим атомы. Их три:

A – " 8 *делится на* 4 ",

B – " 8 *больше 6* ",

C – " *7 больше* 8 ".

Тогда высказывание 1 будет отвечать формуле $A \wedge B$, а высказывание 2 – формуле $A \wedge C$. Будем считать, что высказывание *C* – ложно. Используя истинное значение высказываний *A*, *B*, *C* определим значения высказываний 1 и 2.

$A \wedge B = И \wedge И = И$; $A \wedge C = И \wedge Л = Л$.

Высказывание $A \vee B$ называют дизъюнкцией высказываний *A* или *B*, которое действительно тогда и только тогда, когда истинно хотя бы одно логическое высказывание *A* или *B*. Эта логическая операция отвечает в естественном языке связке " *или* ", что соединяет два предложения.

Пример 1.2.2. Записать в виде формулы логики высказываний и определить истинное значение таких высказываний:

1. " 3 + 2 = 6 *или* 3 $\times$ 2 = 6 " ;
2. " 4 – 2 = 3 *или* 4 $\times$ 2 = 7 " .

Решение. В данных высказываниях выделим атомы:

1. *A* – " 3 + 2 = 6 " ; *B* – " 3 $\times$ 2 = 6 " ;
2. *C* – " 4 – 2 = 3 " ; *D* – " 4 $\times$ 2 = 7 "
3. Тогда высказывание 1 будет отвечать формуле $A \vee B$, а высказывание 2 – формуле $C \vee D$.

По условию высказывание B истинное, а высказывание A, C, D ложное, поэтому : $A \vee B = Л \vee И = И$; $C \vee D = Л \vee Л = Л$.

Пример 1.2.3. Пусть заданы высказывания: A – " *Николай любит играть в футбол* "; B – " *Николай любит играть в волейбол* "; C – " *Николай любит играть в теннис* ". Необходимо записать высказывание " *Николай любит играть в футбол и неправильно, что он любит играть в волейбол или теннис* " в виде формулы логики высказываний и построить соответственно ей таблицу истинности.

Решение. Высказывание " *Николай любит играть в волейбол или теннис* " можно записать в виде формулы логики высказываний как $B \vee C$. Высказывания "*Неправильно, что он любит играть в волейбол или теннис* " может быть записано в виде формулы логики высказываний $\neg(B \vee C)$, поскольку отрицание применимо ко всему высказыванию, которое следует после связки " что ... ". Исходя из этого, исходная форма сложного логического высказывания будет иметь вид $A \wedge \neg(B \vee C)$.

Таблиця истинности такого высказывания приведена в табл. 1.2.3.

Таблиця 1.2.3

A	B	C	$B \vee C$	$\neg(B \vee C)$	$A \wedge \neg(B \vee C)$
Л	Л	Л	Л	И	Л
Л	Л	И	И	Л	Л
Л	И	Л	И	Л	Л
Л	И	И	И	Л	Л
И	Л	Л	Л	И	И
И	Л	И	И	Л	Л
И	И	Л	И	Л	Л
И	И	И	И	Л	Л

Из таблицы истинности сложного высказывания следует, что в том случае данное логическое высказывание истинно, когда высказывание A – истинно, а высказывание B и C – ложны.

1.3. Условные и эквивалентные высказывания

Из талицы истинности, табл. 1.2.2, следует, что высказывание $A \to B$, что представляет собой импликацию (условное предложение), принимает значение ложности тогда и только тогда, когда A - истинно, а B - ложно. В созданном предложении $A \to B$ высказывание A называют **предпосылкой** (**условием**), **а** B – **следствием** (**выводом**). На естественном языке причинно-следственные связи между высказываниями A и B описывают такими оборотами: " *Если A, то B* ", "*A является достаточным основанием для B*" и так далее.

Пример 1.3.1. Для высказывания " *Если идет дождь, то чтобы не промокнуть я открываю зонтик над головой*" записать формулу высказываний и построить таблицу истинности.

Решение. Для данного высказывания введем атомы :

A – " *идет дождь* ", B – " *чтобы не промокнуть я открываю зонтик над головой* ".

Тогда данному высказыванию будет соответствовать формула $A \to B$, а результаты интерпретации данного высказывания приведены в таблице истинности, табл. 1.3.1.

Таблиця 1.3.1

A	B	$A \to B$	Результат
Л	Л	И	*останусь сухим*
Л	И	И	*останусь сухим*
И	Л	Л	*намокну*
И	И	И	*останусь сухим*

С помощью імплікации можно формально выразить понятия достаточного и необходимого условия. Например, если $A \to B$, то это обозначает, что "*A есть достаточным условием для B* " и одновременно, что " *B есть необходимым условием для A* ". То есть необходимость для A можно записать в форме "*B только, если A*" - $A \to B$. удверждение "*A является необходимым и достаточным условием для B*" эквивалентно двойной импликации $(A \to B) \wedge (B \to A) = A \leftrightarrow B$.

Покажем это с помощью таблицы истинности, табл. 1.3.2.

Таблиця 1.3.2

A	B	$A \to B$	$B \to A$	$(A \to B) \wedge (B \to A)$
Л	Л	И	И	И
Л	И	И	Л	Л
И	Л	Л	И	Л
И	И	И	И	И

Из таблицы истинности замечаем, что высказывание $(A \to B) \wedge (B \to A)$ истинно тогда и только тогда, когда высказывания A и B истинны или ложны одновременно, что и отвечает эквиваленции.

Но не при всех условных высказываниях бывает так.

Пример 1.3.2. В высказывании " *Если число n парное* (A), *то n делиться нацело на* 4 (B) " показать необходимость и достаточность условий и записать её истинность через знак импликации.

Решение. Поскольку ни одно непарное число на 4 не делится, то это условие A является необходимой, но в то же время есть парные числа которые не деляться нацело на 4, например 10. То есть, очевидным является отсутствие достаточности условия в заданном высказывании. Поэтому заданное высказывание по условию $A \to B$ ложно, а правильной импликацией для заданного условия будет $B \to A$.

Пример 1.3.3. В высказывании " *Если геометрическая фигура-квадрат* (A), *то она прямоугольник, у которого все стороны равны между собой* (B) " показать необходимость и достаточность условия и записать его через знак логической связки.

Решение. По условию квадрат (A) – это прямоугольник, у которого все стороны равны между собой (B), что является необходимым и достаточным условием для выполнения B и поэтому логической связью между данными высказываниями будет эквиваленция $A \leftrightarrow B$.

С условным высказыванием $A \to B$ связаны еще три высказывания: конверсия, инверсия и контрапозиция. Они определеяются таким образом:

$B \to A$ – конверсия высказывания $A \to B$;

$\neg A \to \neg B$ – инверсия высказывания $A \to B$;

$\neg B \to \neg A$ – контрапозиция высказывания $A \to B$.

Пример 1.3.4. Для высказывания " *Если он хорошо играет в футбол, то он популярный* " найти конверсию, инверсию и контрапозицию.

Решение. В соответствии с их определениями, искомые результаты будут иметь такое содержание.

Конверсия – " *Если он популярный, то он хорошо играет в футбол* ".

Инверсия – " *Если он не хорошо играет в футбол, то он не популярный* ".

Контрапозиция – " *Если он не популярный, то он не хорошо играет в футбол* ".

Определение 1.3.1. Высказывание $A \sim B$ называют эквивалентностью (эквиваленцией) тогда и только тогда, когда высказывание A и B ложны или истинны одновременно. Эта операция отвечает в естественном языке оборотам:

" *тогда и только тогда, когда* ", " *для того, чтобы* ", " *необходимо и достаточно* ".

Из таблицы истинности, табл. 1.3.1., следует, что выражение $A \sim B$ эквивалентно выражению $(A \to B) \wedge (B \to A)$. Это свидетельствует про то, что логическая эквивалентность изображает импликацию в обеих направлениях. Исходя из определения эквивалентности формула для нее имеет следующий вид($\neg$ $A \sim B = (A \wedge B) \vee A \wedge \neg$B).

Пример 1.3.5. Записать в виде формулы логики высказываний и определить истинное значение таких высказываний.

1. "*Для того чтобы* 4 : 2 = 2, *необходимо и достаточно, чтобы* 4 - 2 = 2 ".
2. " 4 : 2 = 3 *равнозначно* 4 - 2 = 1 ".

Решение. Вводим обозначение атомов:

A = 4 : 2 = 2 ; B = 4 - 2 =2 ;

C = 4 : 2 = 3 ; D = 4 - 2 = 1.

Тогда можно сказать, что высказыывание 1 отвечает формуле $A \sim B$, а высказывание 2 – формуле $C \sim D$. Если атомы A и B истинны, а атомы C и D ложны, то высказывание истинности значений сложных высказываний следующее:

$A \sim B = И \sim И = И$; $C \sim D = Л \sim Л = И$.

Чтение формул сложных высказываний может быть неозначным, если не ввести скобки, которые указывают, в каком порядке связываются между собой символы. Последовательность выполнения (приоретет операций в логике высказываний является следующей - $\neg$, $\wedge$, $\vee$, $\to$, $\sim$.

Например, такие выражения без скобок равны формулам со скобками

$A \to B \vee C = A \to (B \vee C)$; $A \to B \wedge C \sim D = (A \to (B \wedge C)) \sim D$.

В логике высказываний любую ее формулу можно поставить в соответствие какому-то

сложному высказыванию естественной формы и наоборот. Для того, чтобы щоб преобразовать сложное предложение в формулу логики высказываний необходимо выполнить следующие шаги алгоритма.

1.Путем анализа сложного предложения определить является ли оно сокращенным.

2. Если предложение сокращенное, то следует его заменить полным вариантом.

3. В полном варианте выделить простые предложения и взять их в скобки, оставив за скобками служебные слова.

4. Процесс взятия в скобки повторять до тех пор, пока полностью все сложное предложение не окажется взятым в скобки.

5. Заменить союзы и обороты естественного языка соответствующими логическими связями, а простые предложения атомарными формулами.

Пример 1.3.6. Предложение *" Поскольку я лег поздно спать, я проспал и поэтому не пошел на работу "* записать в виде формулы логики высказываний.

Решение. В этом сложном предложении выделим простые предложения и возьмем их в скобки *" Поскольку (я лег поздно спать), (я проспал) и поэтому не (пошел на работу) "*.

Все три простые предложения связаны служебными словами, которые выражают логические отношения, а перед третьим предложением стоит частица *" не "*, что отвечает логической операции *" отрицание "*. Поскольку третье предложение не является полным, то дополняем его отсутствующим подлежащим *" я "* и введем атомы:

А – " Я лег поздно спать ", *В – " Я проспал "*, *С – " Я пошел на работу "*.

Заменив простые предложения символами атомов, а служебные слова – логическими связками, получим формулу логики высказываний

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

1.4. Интерпретация формул логики высказываний

Приписывание значений И или *Л* атомарным формулам, которые входят в сложные формулы, называют **интерпретацией**.

Все формулы логики высказываний разделяются на тождественно истинные, тождественно ложные и нейтральные.

Определение 1.4.1. Формулу называют тождественно истинной (тавтологией или общезначимой), если она принимает значение *" Истина "* на всех интерпретациях (наборах значений переменных).

Например, высказывание *" Он пойдет или не пойдет в магазин "* является тавтологией, поскольку или одно высказывание, или другое обязательно состоится.

Пример 1.4.1. С помощью таблицы истинности определить истинность значения формулы логики высказывания

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

Решение. Строим таблицу истинности для заданой формулы логики высказываний, табл. 1.4.1.

Таблиця 1.4.1

A	B	$A \to B$	$A \wedge (A \to B)$	$(A \wedge (A \to B)) \to B$
Л	*Л*	*И*	*Л*	*И*
Л	*И*	*И*	*Л*	*И*
И	*Л*	*Л*	*Л*	*И*
И	*И*	*И*	*И*	*И*

Из таблицы истинности вытекает, что заданное высказывание является *" Истинным "* на всех четырех возможных наборах переменных этого высказывания, поэтому оно является тавтологией.

Пример 1.4.2. Доказать, если высказывание A и $A \to B$ тавтологии, то B тоже тавтология.

Решение. По условию A и $A \to B$ – тавтологии. Пусть при некотором распределении значений истинности для пропозиционных переменных, которые входят в A и B, B принимает значение *"Ложность"*. Но поскольку A есть тавтология, то при том же распределении значений истинности A приобретает значение *"Истинно"*. Тогда высказывание $A \to B$ получает значение *"Ложность"*, но это является противоречием, потому, что $A \to B$ есть тавтология.

Определение 1.4.2. Формулу называют тождественно ложной (противоречивой или неосуществимой), если она приобретает значение *" Ложность "* на всех интерпретациях (наборах значений переменных).

Например, высказывание *" Она двигается в направлении школы и она не двигается в направлении школы "* является противоречивым, так как невозможно одновременно делать и то и другое. То есть это высказывание является тождественно ложным.

Пример 1.4.3.С помощью таблицы истинности определить является ли тождественно ложной формула $(A \to B) \wedge (\neg A \to B) \wedge \neg B$.

Решение. Строим таблицу истинности для заданной формулы логического высказывания, табл. 1.4.2.

Таблиця 1.4.2

A	B	$A \to B$	$\neg A$	$\neg B$	$\neg A \to B$	$(A \to B) \wedge (\neg A \to B) \wedge \neg B$
Л	*Л*	*И*	*И*	*И*	*Л*	*Л*
Л	*И*	*И*	*И*	*Л*	*И*	*Л*
И	*Л*	*Л*	*Л*	*И*	*И*	*Л*
И	*И*	*И*	*Л*	*Л*	*И*	*Л*

Из таблицы истинности выплывает, что высказывание заданное формулой принимает значение *" Ложность "* на всех четырех возможных наборах переменных этого высказывания, то есть является тождественно ложным.

Определение 1.4.3. Формулу называют нейтральной (необщезначимой или непротиворечивой), если она на одних интерпритациях приобретает значение *"Истина"*, а на других *" Ложность "*.

Пример 1.4.4. С помощьютаблицы истинности определить является ли нейтральной формула $((A \to B) \to B) \to B$.

Решение. Строим таблицу истинности за данной формулой логического высказывания, табл. 1.4.3.

Таблиця 1.4.3

A	B	$A \to B$	$(A \to B) \to B$	$((A \to B) \to B) \to B$
Л	*Л*	*И*	*Л*	*И*
Л	*И*	*И*	*И*	*И*
И	*Л*	*Л*	*И*	*Л*
И	*И*	*И*	*И*	*И*

Из таблицы истинности выплывает, что высказывание, заданное логической формулой, приобретает при разных интерпритациях два значения *" Истина "* или *" Ложь "*. Поетому формула логического вісказівання является нейтральной.

Особенная роль в алгебре высказываний принадлежит тождественно истинным формулам, как способам правильных умозаключений, которые от истинных ссылок приводят к истинным высказываниям.

$\overline{\overline{A}} = A$ - закон двойного отрицания;

$A \vee \overline{A}$ - закон исключения третьего;

$A \wedge \overline{A}$ - закон отрицания противоречия;

A⇒A - закон тождественности;

$\left.\begin{array}{l} A \wedge A \Leftrightarrow A \\ A \vee A \Leftrightarrow A \end{array}\right\}$ - закон идемпотетичности;

$(A \Rightarrow B) \Leftrightarrow (\overline{B} \Rightarrow \overline{A})$ - закон контрапозиции;

$(A \Rightarrow B) \wedge (B \Rightarrow C) \Rightarrow (A \Rightarrow C)$ - закон силогизма;

$\left.\begin{array}{l} \overline{A \wedge B} \Leftrightarrow \overline{A} \vee \overline{B} \\ \overline{A \vee B} \Leftrightarrow \overline{A} \wedge \overline{B} \end{array}\right\}$ - закон де Моргана;

$A \wedge (A \Rightarrow B) \Rightarrow B$ - правило Модус поненс(modus ponens)

Для доказательства, что приведенные формулы являются тавтологией достаточно применить таблицу истинности.

Определения 1.4.4. Размышления называют правильными, если оно выражается тождественно истинной формулой.

Для проверки правильности размышления строят соответственно к нему формулу и определеяют является ли она тождественно ситинной. Истинность формулы можно проверить или с помощью таблицы истинности,где на всех интерпритациях она принимает значения *" Истина "*, или с помощью тождественных превращений, сведя их к виду одного из логических законов, где получают тоже значения *" Истина "*, что в соответствии с формулой размышлений является тавтологией.

Определения 1.4.5. Формулы алгебры высказываний $\alpha(A_1, A_2,...,A_n)$ и $\beta(A_1, A_2,...,A_n)$ называются **равносильными (логично эквивалентными)**, если значение истинности формулы α сохраняется со значением истинности формулы β.

1.5. Проблема решения в алгебре высказываний.

Функциональная полнота множества логических операций

Проблема решения в логике высказываний рассматривается как ответ на вопрос: а существует ли алгоритм, который за конечное число шагов дает возможность определить тип любой формулы алгебры высказываний. В алгебре высказываний эта проблема решается позитивно. В часности, можно предложить способы:

1. Составление таблиц истинности формул;
2. Применение метода размышлений от противного;
3. Сведение формул к нормальным формам.

Понятно, что таблица истинности для любой формулы от n высказынных переменных может быть построена за конечное число шагов и определит тип формулы. Но число строк таблицы (2^n) может оказаться слишком большим при значительном количестве атомов, которые составляют формулу. Поэтому на практике в таком случае лучше применять метод 2 или 3.

Анализ формул с использованием метода размышлений от противного базируется на таких умозаключениях:

А) Якщо припустимо, що для деякого набору значень атомів формула $\alpha(A_1, A_2, ..., A_n)$ приймає значення " Хибність ", а після аналізу формули прийдемо до протиріччя, то відносно формули α робимо висновок, що вона є тавтологією.

Б) Якщо припустимо, що для деякого набору значень атомів формула $\alpha(A_1, A_2, ..., A_n)$ приймає значення " Істина " і прийдемо до протиріччя, то є суперечною.

В) Якщо не одержимо протиріччя ні при припущенні А) , ні при припущенні Б), то робимо висновок, що формула α є нейтральною.

Приклад 1.5.1. Визначити тип формули $\alpha = ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$.

Розв'язання. Припустимо, що α не є тотожно істинною формулою. Тоді повинен існувати такий набір значень атомів, на якому вона приймає значення " Хибність ". Формула α є імплікацією. Значення " Хибність " можливо лише за умови, що $(A \Rightarrow B) \Rightarrow A$ приймає на цьому наборі значення " Істина ", а

A - " Хибність ". А тоді, слідує,що $A \Rightarrow B$ повинно приймати значення " Хибність ", що неможливо, оскільки *A* - " Хибність ". Отже, α є тавтологією.

Перед тим, як розв'язувати проблему вирішення, інколи корисно спочатку перетворити формулу логіки висловлювань за допомогою рівносильних перетворень до деякої стандартної форми. Такими формами є диз'юнктивна нормальна форма (ДНФ) та кон'юнктивна нормальна форма (КНФ).

Зафіксуємо деяку множину *X* логічних змінних.

Означення 1.5.1. Елементарною **кон'юнкцією** (**диз'юнкцією**) називається кон'юнкція (диз'юнкція) скінченного числа попарно різних логічних змінних, взятих із запереченням або без нього.

Наприклад, ABC, $\overline{A}B\overline{C}$, $A_1 A_2 \overline{A_3}$ є елементарними кон'юнкціями, $A \vee B \vee C$, $\overline{A_1} \vee A_2 \vee A_3 \vee \overline{A_4}$ є елементарними диз'юнкціями, а $A\overline{A}B$ або $A_1 \vee \overline{A_1} \vee A_2$ вже такими не будуть.

Означення 1.5.2. Елементарні кон'юнкції, що містять всі змінні з *X* , будемо називати конституентами одиниці над *X*, а елементарні диз'юнкції, які задовольняють подібну властивість – конституентами нуля над *X*.

Неважко бачити, що кожна конституента одиниці (відповідно, конституента нуля) тільки на одному, єдиному для неї, наборів атомів приймає значення "Істина" (відповідно, значення "Хибність").

Означення 1.5.3. Диз'юнктивною (кон'юнктивною) нормальною формою називається диз'юнкція (кон'юнкція) скінченого числа попарно різних елементарних кон'юнкцій (диз'юнкцій).

Наприклад, $AB \vee \overline{A} \vee B\overline{C} \vee BCD$ є ДНФ, $(A \vee B)(\overline{A} \vee C)(B \vee C)$ - КНФ. Надалі елементарні кон'юнкції в ДНФ будемо називати доданками, а елементарні диз'юнкції в КНФ – множниками.

Означення 1.5.4. ДНФ (КНФ) називається досконалою, якщо всі її доданки (множники) являють собою конституенти одиниці (нуля) над множиною всіх її атомів.

Досконалі форми будемо відповідно позначати ДДНФ та ДКНФ.

Довільну формулу алгебри висловлювань можна перетворити в одну з нормальних форм. Для цього необхідно виконати ряд кроків:

1. Усунути логічні зв'язки "Імплікація" та "Еквіваленція" за формулами $A \Rightarrow B = \overline{A} \vee B$, $A \sim B = (A \Rightarrow B) \wedge (B \Rightarrow A)$.
2. Використати закон подвійного заперечення та закони де Моргана для перенесення знаку заперечення безпосередньо до атомів.
3. Використати відповідні закони дистрибутивності.

Приклад 1.5.2. Шляхом перетворень одержати для формули $\alpha = A \sim B$ рівносильні їй ДНФ та КНФ.

Рішення. $\alpha = (A \Rightarrow B)(B \Rightarrow A) = (\overline{A} \vee B)(\overline{B} \vee A)$ - *КНФ.*

$\alpha = (\overline{A} \vee B)(\overline{B} \vee A) = \overline{A}\overline{B} \vee B\overline{B} \vee A\overline{A} \vee BA = AB \vee \overline{A}\overline{B}$ - *ДНФ.*

Інший шлях перетворення будь якої формули алгебри висловлювань до нормальних форм – це побудова спочатку для даної формули таблиці істинності, а потім за таблицею складаємо ДДНФ або ДКНФ.

Означення 1.5.5. Деяка множина операцій алгебри висловлювань називається функціонально повною, якщо довільна формула алгебри висловлювань рівносильна формулі, в яку входять лише операції з цієї системи.

Теорема 1.5.1. Кожна з множин операцій $\{\neg, \wedge, \vee\}$, $\{\neg, \wedge\}$, $\{\neg, \vee\}$, $\{\neg, \Rightarrow\}$ є функціонально повною.

Доведення. Оскільки для довільних формул α і β алгебри висловлювань мають місце рівносильності:

$\alpha \vee \beta = \overline{\overline{\alpha} \wedge \overline{\beta}}$, $\alpha \wedge \beta = \overline{\overline{\alpha} \vee \overline{\beta}}$, $\alpha \vee \beta = \overline{\alpha} \Rightarrow \beta$,

то достатньо довести, що перша з означених в теоремі 1.5.1 множин є функціонально повною - $\{\neg, \wedge, \vee\}$, а всі інші автоматично стають функціонально повними за рівносильностями.

Формально це вже доведено, бо за алгоритмом ми зможемо перетворити будь-яку формулу алгебри висловлювань в ДНФ чи КНФ (див. приклад 1.5.2), для сполучення атомів в яких застосовуються тільки операції з множини $\{\neg, \wedge, \vee\}$. Застосуємо інше доведення, використовуючи таблиці істинності і тим самим заодно доведемо ще одну теорему.

Теорема 1.5.2. Кожна функція логіки висловлювань є функцією істинності деякої ДНФ (КНФ).

Доведення. Нехай задана функція логіки висловлювань представима таблицею істинності з 2^n рядками, де кожен рядок містить деякий розподіл значень істинності для набора змінних . В кожному рядку $i = 1, 2, ..., 2^n$ таблиці істинності атоми і сама f_i може набирати значення *X* або *I*. На наборах атомів в *i-му* рядку, що надають функції значення *I* складемо елементарну кон'юнкцію $X_{i1} \wedge X_{i2} \wedge ... \wedge X_{in}$, де $X_{ij} = A_{ij}$, якщо $A_{ij} \in I$, $X_{ij} = \overline{A_{ij}}$, якщо $A_{ij} \in X$. З'єднаємо всі елементарні кон'юнкції операцією диз'юнкція.

Покажемо, що одержана конструкція є шуканою ДНФ.

За означенням 1.5.2. кожна конституента одиниці тільки на одному, єдиному для неї, наборів атомів приймає значення *I*, тобто тільки для наборів *i-го рядку*, а на всіх інших рядках вона прийме значення *X*. Тому кожний кон'юнктивний одночлен в побудованій ДНФ надасть значення *I* тільки в своєму рядку. Отже, побудована таким чином ДНФ є рівносильною до функції представленою таблицею істинності і теорему 1.5.2 доведено, а одночасно доведено і теорему 1.5.1. Для побудови КНФ дії аналогічні.

Приклад 1.5.3. Утворити ДНФ та КНФ для функції логіки висловлювань, яка представлена таблицею істинності.

A_1	A_2	f
X	*X*	*I*
X	*I*	*I*
I	*X*	*I*
I	*I*	*X*

Розв'язання. Для рядків таблиці зі значенням *I* для *f* записуємо елементарні кон'юнкції $\neg A_1 \wedge \neg A_2$, $\neg A_1 \wedge A_2$, $A_1 \wedge \neg A_2$, які зв'язуємо між собою диз'юнкцією. Одержимо ДНФ:

$f = (\neg A_1 \wedge A_2) \vee (A_1 \wedge \neg A_2) \vee (\neg A_1 \wedge \neg A_2)$.

КНФ для даної функції - $f = \neg A_1 \vee \neg A_2$.

1.6. Дедуктивні висновки у логіці висловлювань

У логіці висловлювань правила висновку використовують, щоб виводити одні істинні речення з

інших істинних речень. У коректному дедуктивному виведенні висновок з необхідністю випливає з посилок.

Означення 1.6.1. Логічним наслідком висловлювання A є висловлювання B, якщо формула $A \to B$ є тотожно істинною. Це визначення може бути узагальнено для випадку $A_1, A_2, \ldots, A_n$ висловлювань, якщо $A_1 \wedge A_2 \wedge \ldots \wedge A_n \to B$ – тотожно істинна формула.

Приклад 1.6.1. Показати, що висловлювання $(A \wedge B) \vee \neg C$ є логічним наслідком висловлювання $A \wedge [?][?][\urcorner?] C$.

Розв'язання. Доведемо, що формула $A \wedge [\urcorner?] C((A \wedge B) \vee [\urcorner?] C)$ є загальнозначущою. Для її доведення використаємо тотожності логіки висловлювань та еквівалентні перетворення

$$(A \wedge \neg C) \to ((A \wedge B) \vee \neg C) = \neg(A \wedge \neg C) \vee (A \wedge B) \vee \neg C = \neg A \vee C \vee (A \wedge B) \vee \neg C = \neg A \vee C \vee \neg C \vee (A \wedge B) = \neg A \vee (A \wedge B) \vee 1 = 1.$$

Отже формула $A \wedge \neg C \to ((A \wedge B) \vee \neg C)$ є загальнозначущою, а за означенням 1.6.1 $A \wedge \neg C$ - логічний наслідок.

Означення 1.6.1. Дедуктивним висловлюванням називають висновок формули B з формули A, заснований на тому, що B є логічним наслідком A.

Приклад 1.6.2. Довести правильність міркування за дедукцією - *" Постанова кабінету Міністрів ухвалюється, якщо за неї голосує більшість міністрів. За постанову не проголосувала більшість міністрів, тому постанова не ухвалюється "*.

Розв'язання. До речень висловлювання введемо такі атоми :

A – " за постанову проголосувала більшість міністрів ";

B – " постанова ухвалюється "; $\neg A$ *– " за постанову не проголосувала більшість міністрів ";*

$\neg B$ *– " постанова не ухвалюється ";*

Тоді засновки і висновки зазначимо відповідно через $A \sim B$, $\neg A$, $\neg B$ і приєднавши за допомогою імплікації до кон'юнкції засновків $(A \sim B) \wedge \neg A$ висновок $\neg B$ одержимо $((A \sim B) \wedge \neg A) \to \neg B$.

Перевіримо за допомогою таблиці істинності, табл. 1.6.1., що задана імплікація $((A \sim B) \wedge \neg A) \to \neg B$ є логічним наслідком.

Таблиця 1.6.1

A	B	$A \sim B$	$\neg A$	$\neg B$	$(A \sim B) \wedge \neg A$	$((A \sim B) \wedge \neg A) \to \neg B$
X	*X*	*I*	*I*	*I*	*I*	*I*
X	*I*	*X*	*I*	*X*	*X*	*I*
I	*X*	*X*	*X*	*I*	*X*	*I*
I	*I*	*I*	*X*	*X*	*X*	*I*

Із таблиці істинності слідує, що отримано тотожно істинне висловлювання. Отже, виходячи із законів, задане по умові міркування задовольняє визначення дедуктивного висновку. Таким чином, істинність висновку в дедуктивному висновку гарантується істинністю засновків.

Твердження 1.6.1. Висловлювання B є логічним наслідком висловлювання A, якщо висловлювання $A \to \neg B$ є тотожно хибним.

Твердження 1.6.2. Висловлювання B є логічним наслідком висловлювання A, якщо на всіх інтерпретаціях, на яких A істинне, B теж істинне.

Тотожна істинність або хибність засновків імплікації дозволяє зробити висновок про істинність або хибність наслідку.

Твердження 1.6.3 Якщо висловлювання B є логічним наслідком висловлювання A, а висловлювання A – тотожно істинне висловлювання, то висловлювання B також – тотожно істинне.

Твердження 1.6.4 Якщо висловлювання A є тотожно хибним, то для будь-якого висловлювання B правильно, що $A \to B$.

Правила для дедуктивного висновку будують на підставі загальнозначущих формул логіки висловлень виду $A \to B$.

Для наочного зображення правила умовиводів схематично записують за допомогою риски, над якою пишуть посилки, а під нею – висновок. Якщо посилок дві й більше, їх записують одну під одною (табл. 1.6.2.)

Таблиця 1.6.2

Правило дедуктивного висновку	Тавтологія	Назва правила
$\dfrac{A \to B \quad A}{B}$	$((A \to B) \wedge A) \to B$	Правило відділення (Modus Ponens)
$\dfrac{A \to B \quad B \to C}{A \to C}$	$((A \to B) \wedge (B \to C)) \to (A \to C)$	Гіпотетичний силогізм
$\dfrac{A \to B \quad \neg B}{\neg A}$	$(A \to B) \vee \neg B \to \neg A$	Від'ємна форма правила відділення (Modus Tollens)
$\dfrac{A}{A \vee B}$	$A \to (A \vee B)$	Правило введення диз'юнції (правило розширення)
$\dfrac{A \quad B}{A \wedge B}$	$((A) \wedge (B)) \to (A \wedge B)$	Правило введення кон'юнції
$\dfrac{A \vee B \quad \neg A}{B}$	$(A \vee B) \wedge \neg A \to B$	Правило видалення диз'юнції (диз'юнктивний силогізм)
$\dfrac{A \wedge B}{A}$	$(A \wedge B) \to A$	Правило видалення кон'юнції
$\dfrac{A \to B}{\neg B \to \neg A}$	$(A \to B) \to (\neg B \to \neg A)$	Правило контрапозиції імплікації
$\dfrac{A \quad A \to (B \vee C) \quad B \to D \quad C \to D}{D}$	$((A \wedge (A \to (B \vee C)) \wedge (B \to D) \wedge (C \to D)) \to D$	Правило вибору
$\dfrac{A \vee B \quad A \to (C \wedge \neg C)}{B}$	$((A \vee B) \wedge (A \to (C \wedge \neg C))) \to B$	Правило виключаючого вибору
$\dfrac{\neg A \to (B \wedge \neg B)}{A}$	$(\neg A \to (B \wedge \neg B)) \to A$	Правило зведення до абсурду (Reduction ad Absurdum)

Правильність всіх перелічених в табл. 1.6.2 логічних висловлювань можна довести за допомогою таблиць істинності. З усіх правил, наведених в табл. 1.6.2, найбільш часто використовується правило відділення. Правило відділення має такий логічний сенс, якщо засновок правильний, то правильний і наслідок із нього.

Приклад 1.6.3. Дано висловлювання *" Якщо n ділиться на* 9*, то n ділиться на* 3 *"*. Також відомо, що *"n ділиться на* 9*"*. Який висновок можна зробити, виходячи з цих двох висловлювань?

Розв'язання. Введемо атомарні висловлювання :

A – " n ділиться на 9 *",*

B – " n ділиться на 3 *".*

Висловлювання *" Якщо n ділиться на* 9*, то n ділиться на* 3 *"* можна записати у вигляді формули $A \to B$. З одночасного виконання засновків $A \to B$ і A можна зробити висновок B за

правилом відділення “ *n ділиться на* 3 ”.

Питання для самоперевірки:

1. Який вид речень моделює формальна логіка?
2. Наведіть приклад речень, які не розглядають у формальній логіці.
3. Що таке висловлювання? Дайте визначення.
4. Що мають на увазі під істинним значенням висловлювання?
5. Які висловлювання називають атомом?
6. Що мають на увазі під інтерпретацією висловлювання?
7. Що мають на увазі під однозначно визначеним висловлюванням?
8. Дайте визначення логічного наслідку одного (кількох) висловлювань.
9. Наведіть формули логіки висловлювань, що містять будь-які логічні зв’язки, і відповідні до них речення природної мови.
10. Дайте визначення умовного і еквівалентного висловлювання.
11. Дайте визначення конверсії, інверсії та контрапозиції висловлювання.
12. Дайте визначення правильно побудованої формули логіки висловлювань.
13. Наведіть формули логіки висловлювань і відповідні їм речення природної мови, що містять будь-які логічні зв’язки.
14. Сформулюйте алгоритм запису складного речення природної мови у вигляді формули логіки висловлювань.
15. Дайте визначення інтерпретації формули логіки висловлювань.
16. Яку формулу логіки висловлювань називають тавтологією?
17. Яку формулу логіки висловлювань називають тотожно хибною?
18. Яку формулу логіки висловлювань називають нейтральною?
19. Яку загальну систему повноти логічних зв’язок мають логічні висловлювання?
20. На які системи повноти логічних зв'язок розпадається повна система?
21. Дайте визначення дедуктивного висновку.
22. В чому полягає різниця дедуктивного і не дедуктивного висновку.
23. Яким чином будують дедуктивний висновок?
24. Дайте характеристику основних правил дедуктивного висновку.
25. Дайте визначення логіки висловлювань.
26. Дайте визначення побудові формули в логіці висловлювань.
27. Що таке інтерпретація висловлювання?
28. Що називають однозначно визначеним висловлюванням?
29. Що називають запереченням висловлювання?
30. Що називають кон’юнцією і диз’юнкцією висловлювання?
31. Які операції в логіці висловлювань є унарні, а які бінарні, назвіть їх?
32. Що у логіці висловлювань називають логічними зв’язками?
33. Скільки зв’язків є у логіці висловлювань, назвіть їх?

Задачі для самостійного розв’язування

1. Знайдіть висловлювання серед наведених нижче речень. Укажіть їх істинне значення:
   а) Який сьогодні день?
   б) Бережись автомобіля?
   в) Юпітер є найближча до Сонця планета;
   г) Усі парні числа діляться на *2*;
   д) Якщо x=2, то $x^2=4$;
   є) Загрузіть ці пакети в машину;
   ж) Не слід зберігати компакт-диски в мікрохвильовій пічці;
   з) Це твердження не може бути істинним.
2. Задані висловлювання :
   A – “*Подорож на Місяць є дорога*”;
   B – “*Я маю кошти*”;
   C – “*Я полечу на Місяць*”.
   Записати слідуючи складні висловлювання у вигляді формули логіки висловлювань:
   а) “ *У мене не має коштів і я не полечу на Місяць* ”;
   б) “ *Невірно, що у мене є кошти і я полечу на Місяць* ”;
   в) “ *У мене не має коштів і подорож на Місяць дорога або я полечу на Місяць* ”;
   г) “ *Подорож на Місяць не є дорога і я полечу на Місяць або подорож на Місяць є дорога і я не полечу на Місяць* ”.
3. Задані висловлювання :
   A – “Мій *комп’ютер – швидкісний*”;
   B – “*Я складу іспит*”;
   C – “*Я закінчу проект вчасно*”.
   Записати такі складні висловлювання у вигляді формули логіки висловлювань :
   а) “ *Я не закінчив проект вчасно і не складу іспит* ”;
   б) “ *Неправильно, що я закінчу проект вчасно* ”;
   в) “ *У мене не швидкісний комп’ютер або я закінчу проект вчасно*”;
   г) “ *У мене швидкісний комп’ютер або я не закінчу проект вчасно і складу іспит* ”.
4. Побудуйте таблиці істинності для кожного з висловлювань завдання 2 і 3.
5. Побудуйте складні висловлювання, з використанням операцій:
   а) імплікація;
   б) еквівалентність;
   в) імплікація і кон’юнкція.
6. Побудуйте складне висловлювання, еквівалентне $A \wedge B$, використовуючи операції диз’юнкції і заперечення.
7. Побудуйте два складних висловлювання, еквівалентних $A \to B$, використовуючи операції кон’юнкції і заперечення.
8. Розставте дужки у формулах :
   а) $A \vee [\urcorner?]\ B \sim [\urcorner?]\ C \vee D \sim A[?][?]$;
   б) $[\urcorner?]\ A \to [\urcorner?]\ B \wedge C \sim [\urcorner?]\ D$;
   в) $[\urcorner?]\ A \wedge B \vee [\urcorner?]\ D \to [\urcorner?]\ C \sim [\urcorner?]\ D$;
   г) $A \sim B \vee [\urcorner?]\ A \to C \wedge A \sim [\urcorner?]\ D$.
9. Для висловлювання “ *Якщо я голосую, то я гарний громадянин* ” сформулюйте :
   а) конверсію цього висловлювання;
   б) інверсію цього висловлювання;
   в) контрапозицію цього висловлювання.
10. Виключити якомога більше дужок у формулах :
    а) $(([\urcorner?]\ (A) \vee (B)) \wedge [\urcorner?]\ (C)) \to ((A \to [\urcorner?]\ (D))$;
    б) $((((A) \to (B)) \vee (C) \wedge (D)) \to [\urcorner?]\ (B))$;
    в) $[\urcorner?]\ (([\urcorner?]\ (A) \wedge ([\urcorner?]\ B) \to (C) \vee (D))$;
    г $[\urcorner?]\ ((B) \to A) \vee ([\urcorner?]\ A \to (B) \wedge D) \vee [\urcorner?]\ (A \wedge B))$.
11. Серед заданих формул логіки висловлювань знайти загальнозначущі, суперечливі або несуперечливі формули :
    а) $[\urcorner?]\ ([\urcorner?]\ B) \to B$;
    б) $[\urcorner?]\ (A \to B) \wedge B$;
    в) $(A \to B) \wedge (A \to [\urcorner?]\ B)$;
    г) $[\urcorner?]\ ((A \to B) \to B) \to A)$;
    д) $(A \wedge (B \to A)) \to B$;
    є) $(A \to B) \to (B \to A) \wedge [\urcorner?]\ A$;
    ж) $(A \sim [\urcorner?]\ B) \to ([\urcorner?]\ A \sim B)$.
12. За допомогою таблиць істинності функції $f(x_1,x_2)$ визначити повноту (пари) зв'язок логічних висловлювань.

x_1	x_2	$f(x_1,x_2)$
X	*X*	*I*
X	*I*	*I*
I	*X*	*X*
I	*I*	*I*

а)

x_1	x_2	$f(x_1,x_2)$
X	*X*	*I*
X	*I*	*I*
I	*X*	*I*
I	*I*	*X*

б)

x_1	x_2	$f(x_1,x_2)$
X	*X*	*I*
X	*I*	*X*
I	*X*	*X*
I	*I*	*I*

в)

13. “*Якщо фірма відмовляється виконати умови страйкарів, то страйк не буде закінчено, якщо він не триває більше року і президент фірми не йде у відставку. Чи закінчиться страйк, якщо фірма відмовляється діяти і страйк тільки розпочався?*” Побудуйте логічний висновок і отримайте відповідь.
14. Побудуйте дедуктивний висновок такого логічного висловлювання “ *Якщо студент не вивчив теорію, то він не виконає завдання. Студент не вивчив теорію. Отже, студент не виконає завдання* ”.
15. Визначте тип правила дедуктивного висновку, яке може бути використано у такому міркуванні “ *Йде сніг і температура повітря - 10С^0. Отже температура повітря - 10С^0.*

Коментарі. Основні відомості, щодо поняття логіки висловлювань і необхідних для них логічних зв’язок, викладені в [1, 7], умовні і еквівалентні висловлювання взяті з [7, 22], повнота

систем логічних зв'язок - із [7, 18], а дедуктивні висновки у логіці висловлювань випливають із [14].

Тема 2

- Числення висловлювань
  - 2.1. Формальна аксіоматична теорія L
  - 2.2. Теорема дедукції
  - 2.3. Побудова доведень у логіці висловлювань
  - 2.4. Аксіоматичний метод
  - 2.5. Конструктивний метод
  - 2.6. Метод доведення від супротивного
  - 2.7. Метод резолюцій

Числення висловлювань

При побудові алгебри висловлювань (розділ 1) в основу покладено поняття висловлювання, як об'єкту приймаючому логічні значення "Істина" або "Хибність" (закон виключеного третього). Ці поняття в багатьох випадках суб'єктивні і не відносяться до математичних.

В цьому розділі розглядається формалізована аксіоматична теорія – числення висловлювань, яка вільна від указаної вище вимоги і представляє одну з можливих аксіоматизацій алгебри висловлювань.

Довільну формулу F числення висловлювань можна змістовно інтерпретувати як складене висловлювання, істинність або хибність якого залежить від істинності елементарних висловлювань, що до нього входять. Таким чином, кожній формулі F числення висловлювань можна аналогічно тому, як це було зроблено в алгебрі висловлювань, поставити у відповідність функцію істинності f.

При побудові числення висловлювань можуть бути вибрані різні системи аксіом і правила виводу, проте при будь-якому виборі множина формул числення висловлювань збігається з множиною тотожно істинних формул алгебри висловлювань.

2.1. Формальна аксіоматична теорія L

Означення 2.1.1. Формальна аксіоматична теорія ***L*** є визначеною, якщо виконані такі умови.

1. 1. Задано деякий злічений алфавіт – символи теорії L. Скінченні послідовності символів теорії L називають виразами теорії ***L***.
2. 2. Задано підмножину виразів теорії *L*, яку називають множиною формул теорії ***L*** (часто існує процедура, за допомогою якої можна завжди визначити, чи є даний вираз формулою).
3. 3. Задано деяку множину формул, елементи якої називають аксіомами теорії ***L*** (якщо є можливість перевірити, чи є дана формула аксіомою, то теорію *L* називають ефективно аксіоматизованою, або аксіоматичною теорією).
4. 4. Задано скінченну множину $R_1, R_2,..., R_n$ відношень між формулами, які називають правилами виведення.

Для будь-якого R_i є ціле додатне j, таке, що для множини формул $A_1, A_2,..., A_j$ і будь-якої формули A ефективно вирішується питання, чи знаходяться ці формули $A_1, A_2,..., A_j$ у відношенні R_i з формулою A, і якщо це так, то A називають безпосереднім наслідком формул $A_1, A_2,..., A_j$ за правилом $R_i(A_1, A_2,..., A_j)$.

Означення 2.1.2. Виведенням у теорії ***L*** називають таку послідовність формул $A_1, A_2,..., A_n$, коли будь-яка з формул A_i є або аксіомою, або безпосереднім наслідком деяких попередніх формул за одним із правил виведення.

Означення 2.1.3. Теоремою теорії ***L*** називають таку формулу A, коли існує виведення в теорії *L* формули A, в якому останнім елементом є формула A. Це виведення називають виведенням формули ***A*** в теорії *L*.

Якщо є алгоритм для перевірки, чи *A* є теоремою теорії *L*, то теорію *L* називають розв'язною теорією, інакше – теорією, що не є розв'язною.

Означення 2.1.4. Формулу ***A*** називають наслідком у теорії ***L*** множини формул *Г* тоді і тільки тоді, коли існує така послідовність формул $A_1, A_2,..., A_n$, що $A_n = A$, і будь-яка з формул A_i є або аксіомою, або елементом *Г*, або безпосереднім наслідком деяких попередніх формул за одним із правил виведення.

Послідовність $A_1, A_2,..., A_n$ називають виведенням *A* із *Г*, а елементи множини *Г* – гіпотезами виведення.

Запис $\Gamma \vdash A$ означає, що *A* є наслідком множини формул *Г*. Якщо формула *Г* скінченна $\Gamma = \{B_1, B_2,..., B_n\} \vdash A$, то пишуть $B_1, B_2,..., B_n \vdash A$. Якщо *Г* – порожня множина $\varnothing$, то запис $\Gamma \vdash A$ має місце тоді і тільки тоді, коли *A* є теорема. Замість $\varnothing \vdash A$ прийнято писати $\vdash A$. Таким чином, $\vdash A$ є скороченим підтвердженням, що *A* – теорема.

Наведемо декілька простих властивостей поняття виведення із посилок.

1. Якщо $\Gamma \subseteq \Delta$ і $\Gamma \vdash A$, то $\Delta \vdash A$.
2. $\Gamma \vdash A$ тоді і тільки тоді, коли в *Г* є скінченна підмножина Δ, для якої $\Delta \vdash A$.
3. Якщо $\Delta \vdash A$ і $\Gamma \vdash A$ для будь-якого *B* із множини Δ, то $\Gamma \vdash A$.

Властивість 1 показує, що якщо *A* виведене із множини посилок *Г*, то вона залишиться виведеною, якщо додати до *Г* нові посилки. Частина "*тоді*" твердження 2 випливає із 1. Частина "*тільки тоді*" - це твердження є очевидним, якщо будь-яке виведення *A* із *Г* використовує скінченне число посилок із *Г*. Твердження 3 означає, якщо *A* вивідне із Δ і кожна формула, яка є в Δ, також вивідна з *Г*, то *A* виводимо із *Г*.

Формальна аксіоматична теорія *L* для числення висловлювань задається таким способом.

1. 1. Символами теорії *L* є $\neg, \rightarrow, (,)$ і букви A_j з цілими додатними числами якості індекси: $A_1, A_2, A_3,...$. Символи $\neg, \rightarrow, (,)$ називають примітивними зв'язками, а A_j – пропозиційними змінними, або пропозиційними буквами.
2. 2. Усі пропозиційні букви A_j є формулами. Якщо *A* і *B* - формули, то $\neg A$, $A \rightarrow B$ теж формули теорії *L*.
3. 3. Для будь-яких формул *A*, *B*, *C* теорії *L* формули:

A1) $A \rightarrow (B \rightarrow A)$;

A2) $((A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$;

A3) $(\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow B) \rightarrow B)$

є аксіомами теорії *L*.

4. Єдиним правилом виведення служить правило ***modus ponens*** (***MP***), за допомогою якого із формули *A* і $A \rightarrow B$ виводиться *B*:

$A, A \rightarrow B \vdash B$.

Необхідно зазначити, що нескінченна множина аксіом теорії *L* задана за допомогою тільки трьох схем аксіом *A1*, *A2*, *A3*, кожна з яких породжує нескінченну множину аксіом. Для будь-якою формули логіки висловлювань легко перевірити, є вона аксіомою чи ні, а звідси випливає, що теорія *L* є ефективно аксіоматизована теорія.

Інші зв'язки в аксіоматичній теорії *L* можна ввести за допомогою вже відомих формул

$A \& B \Leftrightarrow \neg(A \rightarrow \neg B)$; $A \vee B \Leftrightarrow \neg A \rightarrow B$; $A \Leftrightarrow B \Leftrightarrow (A \rightarrow \neg B) \& (B \rightarrow A)$

Теорема 2.1.1. Довести, що для будь-якої формули *A* виконується умова

$\vdash_L A \rightarrow A$. Знак $\vdash_L$ - означає, що мова йде про теорію *L*.

Доведення. Побудову виведення формули $A \rightarrow A$ виконаємо в теорії *L*.

1. Підставимо формулу $A \rightarrow A$ у схему аксіоми *A2*, після чого отримаємо

$(A \rightarrow ((A \rightarrow A) \rightarrow A)) \rightarrow ((A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A))$.

2. Але формула $(A\to((A\to A)\to A))$ - це є схема аксіоми *A1*.
3. Використовуючи 1 і 2 за правилом *MP*, отримаємо

$$(A\to(A\to A))\to(A\to A).$$

4. Використовуючи схему аксіоми *A1*, отримаємо

$$A\to(A\to A).$$

5. Використовуючи 3 і 4 за правилом *MP*, отримаємо

$$A\to A.$$

Необхідно зазначити, що кожна з аксіом *A1,…,A3* є незалежна.

Існують інші аксіоматизації числення висловлювань, наприклад, Гільберта-Аккермана, Расселла, Кліні.

Логічними зв'язками аксіоматизації Гільберта-Аккермана є $\neg$, $\vee$, а $A\to B$ служить скороченням для формули $\neg A\vee B$. Аксіоми є такими:

ГА1) $A\vee A\to A$;
ГА2) $A\to A\vee B$;
ГА3) $A\vee B\to B\vee A$;
ГА4) $(B\to C)\to((A\vee B)\to(A\vee C))$.
Правилом виведення є *MP*.

Логічними зв'язками **аксіоматизації Расселла** є $\wedge$, $\neg$, а $A\to B$ – скорочення для формули $\neg(A\wedge\neg B)$. Його аксіоми мають такий вигляд:

РА1) $A\to(A\wedge A)$;
РА2) $(A\wedge A)\to A$;
РА3) $(A\to B)\to(\neg(B\wedge C)\to\neg(C\wedge A))$.
Правилом виведення є *MP*.

Логічними зв'язками **аксіоматизації Кліні** є $\neg,\wedge,\vee,\to$, аксіоми мають такий вигляд

КА1) $A\to(B\to A)$;
КА2) $(A\to(B\to C))\to((A\to B)\to(A\to C))$;
КА3) $A\wedge B\to A$;
КА4) $A\to(B\to(A\wedge B))$;
КА5) $A\to(A\vee B)$;
КА6) $B\to(A\vee B)$;
КА7) $(A\to C)\to((B\to C)\to((A\vee B)\to C))$;
КА8) $(A\to B)\to((A\to\neg B)\to\neg A)$;
КА9) $\neg\neg A\to A$.
Правилом виведення є *MP*.

2.2. Теорема дедукції

Теорема дедукції дає нове правило виведення у логіці висловлювань, і тому вона є важливою при доведенні теорем і різних висловлювань.

Теорема дедукції 2.2.1. Якщо *Г* – множина формул, *A* і *B* – формули і $\Gamma, A\vdash B$, то $\Gamma\vdash A\to B$. Зокрема, якщо $A\vdash B$, то $\vdash A\to B$.

Доведення. Нехай $B_1, B_2,\ldots, B_n$ – виведення із $\Gamma\cup\{A\}$ формули *B*, тобто $B_n = B$. Доведення будемо проводити індукцією по *i* – довжині виведення $(1\le i\le n)$ формули *B*.

При *i=1* B_1 повинно бути або а) елементом *Г*; б) аксіомою;
в) самою формулою *A*.

У випадках "а" і "б" теорема випливає з того, що за схемою *A1* маємо $B_1\to(A\to B_1)$. Звідси за *MP* отримаємо, що $\Gamma\to(A\to B_1)$.

У випадку "б", тобто коли $A = B_1$, із доведеного вище маємо, що $A\vdash A$ – теорема. Отже, $\Gamma\vdash A\to A$ і випадок *i=1* цим вичерпано.

Нехай тепер $\Gamma\vdash A\to B_k$ справедливе для всіх $k<i$. Покажемо, що $\Gamma\vdash A\to B_i$. У цьому випадку можливі такі варіанти

а) B_i є аксіомою; б) B_i є елементом *Г*;

в) $B_i = A$; г) B_i є наслідком за правилом *MP* деяких формул B_j і $B_m(j<m<i)$, і B_m має вигляд $B_j\to B_i$.

Доведення випадків "а" - "в" нічим не відрізняється від доведення цих випадків для *i=1*.

У випадку "г", використовуючи принцип індукції, маємо $\Gamma\vdash A\to B_j$ і $\Gamma\vdash A\to(B_j\to B_i)$.

За схемою аксіом *A2* маємо $(A\to(B_j\to B_i))\to((A\to B_j)\to(A\to B_i))$. Отже, за правилом *MP* отримаємо $\Gamma\vdash(A\to B_j)\to(A\to B_i)$ і знову за правилом *MP* – $\Gamma\to A\to B_i$, що й потрібно було довести.

Теорема 2.2.2. Довести, що $A\to B,\ B\to C\vdash A\to C$.
Доведення. За означенням теорії *L*

1) $A\to B$ – гіпотеза;
2) $B\to C$ – гіпотеза;
3) *A* – гіпотеза;
4) Використовуючи *MP* із 1 і 3, отримаємо *B*;
5) Використовуючи *MP* із 2 і 4, отримаємо *C*.

Отже, $A\to B,\ B\to C,\ A\vdash C$ і за теоремою дедукції маємо $A\to B,\ B\to C\vdash A\to C$,що й потрібно було довести.

Теорема 2.2.3. Довести, що $A\to(B\to C), B\vdash A\to C$.
Доведення. За означенням теорії *L*

1. $A\to(B\to C)$ – гіпотеза;
2. *B* – гіпотеза;
3. *A* – гіпотеза;
4. використовуючи *MP* із 1 і 3, отримаємо $B\to C$;
5. використовуючи *MP* із 2 і 4, отримаємо *C*.

Отже, $A\to(B\to C), B, A\vdash C$, і за теоремою дедукції маємо $A\to(B\to C), B\vdash A\to C$, що і потрібно було довести.

Теорема 2.2.4. Для будь-яких формул *A* і *B* такі формули є теоремами формальної теорії *L*:
а) $\neg\neg B\to B$**; д)** $(A\to B)\to(\neg B\to\neg A)$**;**
б) $B\to\neg\neg B$**; є)** $A\to(\neg B\to\neg(A\to B))$**;**
в) $\neg A\to(A\to B)$**; ж)** $(A\to B)\to((\neg A\to B)\to B)$**.**
г) $(\neg B\to\neg A)\to(A\to B)$;

а) $\vdash\neg\neg B\to B$
Доведення

1. 1.$(\neg B\to\neg\neg B)\to((\neg B\to\neg B)\to B)$ – схема аксіом *A3*;
2. 2.$\neg B\to\neg B$ – теорема 2.1.1;
3. 3.$(\neg B\to\neg\neg B)\to B$ – теорема 2.2.3;
4. 4.$\neg\neg B\to(\neg B\to\neg\neg B)$ – схема аксіом *A1*;
5. 5.$\neg\neg B\to B$ –

б) $B\to\neg\neg B$.
Доведення

1. 1.$(\neg\neg\neg B\to\neg B)\to((\neg\neg\neg B\to B)\to\neg\neg B)$ – схема аксіом *A3*;
2. 2.$\neg\neg\neg B\to\neg B$ – теорема 2.2.4а;
3. 3.$(\neg\neg\neg B\to B)\to\neg\neg B$ – за *MP* із 1 і 2;
4. 4.$B\to(\neg\neg\neg B\to B)$ – схема аксіом *A1*;
5. 5.$B\to\neg\neg B$ – теорема 2.2.2 і за *MP* із 3 і 4.

Зіставляючи теореми 2.2.4а і 2.2.4б, отримаємо теорему

$B \Leftrightarrow \neg\neg B$,

яка виражає відомий нам із алгебри логіки закон подвійного заперечення.

в) $\vdash \neg A \to (A \to B)$.

Доведення

1. 1. $\neg A$ – гіпотеза;
2. 2. A – гіпотеза;
3. 3. $A \to (\neg B \to A)$ – схема аксіом *A1*;
4. 4. $\neg A \to (\neg B \to \neg A)$ – схема аксіом *A1*;
5. 5. $\neg B \to A$ – за *MP* із 2, 3;
6. 6. $\neg B \to \neg A$ – за *MP* із 1, 4;
7. 7. $(\neg B \to \neg A) \to ((\neg B \to A) \to B)$ – схема аксіом *A3*;
8. 8. $(\neg B \to A) \to B$ – за *MP* із 6, 7;
9. 9. B – за *MP* із 5, 8.

Отже, використовуючи 1 – 9, маємо, що $\neg A, A \vdash B$ і за теоремою дедукції отримаємо $\vdash \neg A \to (A \to B)$.

г) $\vdash (\neg B \to \neg A) \to (A \to B)$.

Доведення

1. 1. $\neg B \to \neg A$ – гіпотеза;
2. 2. A – гіпотеза;
3. 3. $(\neg B \to \neg A) \to ((\neg B \to A) \to B)$ – схема аксіом *A3*;
4. 4. $A \to (\neg B \to A)$ – схема аксіом *A1*;
5. 5. $(\neg B \to A) \to B$ – за *MP* із 1, 3;
6. 6. $A \to B$ – теорема 2.2.2;
7. 7. B – за *MP* із 2, 6.

Отже, використовуючи 1–7 маємо, що $\neg B \to \neg A, A \vdash B$ і, двічі застосувавши дедукції, отримаємо $\vdash (\neg B \to \neg A) \to (A \to B)$.

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

Доведення

1. 1. $A \to B$ – гіпотеза;
2. 2. $\neg\neg A \to A$ – теорема 2.2.4а;
3. 3. $\neg\neg A \to B$ – теорема 2.2.2;
4. 4. $B \to \neg\neg B$ – теорема 2.2.4б;
5. 5. $\neg\neg A \to \neg\neg B$ – теорема 2.2.2;
6. 6. $(\neg\neg A \to \neg\neg B) \to (\neg B \to \neg A)$ – теорема 2.2.4д;
7. 7. $\neg B \to \neg A$ – за *MP* із 5, 6.

Отже, використовуючи 1–7 маємо $A \to B \vdash \neg B \to \neg A$ і за теоремою отримаємо $\vdash (A \to B) \to (\neg B \to \neg A)$.

Порівнюючи теореми 2.2.1г і 2.2.1д, отримаємо теорему

$A \to B \Leftrightarrow \neg B \to \neg A$,

на якій ґрунтується метод доведення тверджень, відомий під назвою метод доведення від супротивного.

е) $\vdash A \to (\neg B \to (A \to B))$

Доведення. Оскільки A, $A \to B \vdash B$, то за теоремою отримаємо $A \to ((A \to B) \to B)$.
За теоремою 2.2.4д маємо
$\vdash ((A \to B) \to B) \to (\neg B \to \neg (A \to B))$.
Використовуючи теорему 2.2.2, отримаємо
$\vdash A \to (\neg B \to (A \to B))$.

є) $(A \to B) \to ((\neg A \to B) \to B)$

Доведення.

1. 1. $A \to B$ – гіпотеза.
2. 2. $\neg A \to B$ – гіпотеза.
3. 3. $(A \to B) \to (\neg B \to \neg A)$ – теорема 2.2.4 д.
4. 4. $\neg B \to \neg A$ – за *MP* із 1,3.
5. 5. $(\neg A \to B) \to (\neg B \to \neg\neg A)$ – теорема 2.2.4.д..
6. 6. $\neg B \to \neg\neg A$ – за *MP* із 2,5;
7. 7. $((\neg B \to \neg\neg A) \to (\neg B \to \neg A) \to B)$ – схема аксіом *A3*.
8. 8. $(\neg B \to \neg A) \to B$ – за *MP* із 6,7.
9. 9. B – за *MP* із 4,8.

Отже, $A \to B, \neg A \to B \vdash B$, і застосувавши двічі теорему дедукції, отримаємо
$\vdash (A \to B) \to ((\neg A \to B) \to B)$.

2.3. Побудова доведень у логіці висловлювань

Логіка – це наука про способи доведення. У звичайній логіці всі доведення будуються на відношенні еквівалентності, а в логіці висловлювань – на відношенні порядку, тобто на відношенні, яке є між причиною і наслідком. При цьому дуже важливо в логіці висловлювань відрізняти мови і мета мови, об'єктні та суб'єктні висловлювання. При нехтуванні цією різницею є ризик потрапити в протиріччя, яке називають логічним парадоксом. Розглянемо дію такого парадоксу на практиці: "*Парадокс брехуна*". "*Я брехун*", – сказав брехун. Але оскільки брехун каже про себе, що він брехун, то він виступає у своїй протилежній якості, а саме небрехуна. Тому наведене висловлювання необхідно розуміти інакше "*Я брехун*", – сказав небрехун. Але тепер із цього висловлювання випливає, що правдивий чоловік повідомляє про себе, що він брехун. Правдивому чоловікові потрібно вірити. Тому друге висловлювання необхідно розуміти все-таки так, я все це відтворено в першому висловлюванні. Таким чином, виникає невизначеність, яка полягає в тому, що незрозуміло, як кваліфікувати того, хто говорить, – як брехуна чи як не брехуна, а звідси ідентифікувати як істинне чи як хибне.

Логічний парадокс тут виник тому, що в наведених висловлюваннях не робиться розділення між двома принципово різними логічними рівнями. Тут, крім "*брехуна*" чи "*небрехуна*", у цій логічній ситуації бере участь суб'єкт "*метаспостерігач*". Якщо провести чітке синтаксичне відділення смислового змісту, яке повинно стосуватися до нас як метаспостерігачів, від іншої семантики об'єктних персонажів, то логічну суперечність буде знято. Тоді ситуацію з брехуном можна подати таким чином.

"*Я брехун*", – сказав брехун.
"*Це істина*", – сказав метаспостерігач.
"*Я брехун*", – сказав небрехун.
"*Це хибно*", – сказав метаспостерігач.
"*Я небрехун*", – сказав брехун.
"*Це хибно*", – сказав метаспостерігач.
"*Я небрехун*", – сказав небрехун.
"*Це істина*", – сказав метаспостерігач.

Для розпізнавання мови логіки висловлювань і метамови дослідника між посилками і наслідком замість об'єктивного символу імплікації "$\to$" будемо ставити суб'єктивний символ метаімплікації "$\Rightarrow$", а між посилками замість об'єктивних символів кон'юнкції "$\wedge$" і диз'юнкції "$\vee$", ставити суб'єктивні символи метакон'юнкції "$,$" і метадиз'юнкції "$;$".

Тоді твердження, яке необхідно довести, в логіці висловлювань можна оформити у вигляді такого причинно-наслідкового відношення:

$P_1, P_2, ..., P_{n+1}, P_n \Rightarrow C$, (2.3.1)

де P_i – посилка (причина); C – висновок (наслідок).

Читається: "*Якщо посилки* $P_1, P_2, ..., P_{n+1}, P_n$ *істинні, то і висновок* C *також істинний*", або "*Якщо причини* $P_1, P_2, ..., P_{n+1}, P_n$ *мали місце, то матиме місце і наслідок*".

Щоб не переплутати об'єктивне висловлювання із суб'єктивним висловлюванням, справедливість якого необхідно встановити, домовимося речення 2.3.1. називати клаузою

(clause).

Означення 2.3.1. Клаузою називають **метапропозицію**, в якій виконується відношення порядку, оформлене через символ метаімплікації "$\Rightarrow$".

Як і відношення еквівалентності, **відношення порядку задовольняє три закони**:

- **рефлективності** – $A \Rightarrow A$;
- **антисиметричності** – якщо $A \Rightarrow B$, то $\neg B = \neg A$;
- **транзитивності** – якщо $A \Rightarrow B$ і $B \Rightarrow C$, то $A = C$.

На відміну від еквівалентності відношення порядку припускає виконання закону **антисиметричності**, який можна записати так:

якщо $A \Rightarrow B$ і $B \Rightarrow A$, то $A = B$.

Замість букв у клаузі можна підставляти об'єктивні висловлювання, і тоді вона наповнюється конкретним змістом, який іменується семантикою, або легендою. Наприклад, нехай задана клауза

$A \to B$, $A \Rightarrow B$.

Якщо прийняти, що $A =$ "*блиснула блискавка*", а $B =$ "*грянув грім*", то можна скласти таку легенду: "*Відомо, що якщо блиснула блискавка, то після цього гряне грім. Блиснула блискавка. Тому повинен грянути грім*".

Над суб'єктом, який формулює метапропозиції, може стояти другий суб'єкт, для якого уже пропозиції першого суб'єкта будуть об'єктивні. Тоді клаузу 2.3.1 другий суб'єкт, або метасуб'єкт, запише для себе таким логічним виразом: $(P_1 \wedge P_2 \wedge ... \wedge P_{n+1} \wedge P_n) \to C$.

Перетворивши цей вираз у диз'юнкцію, отримаємо

$\neg P_1 \vee \neg P_2 \vee ... \vee \neg P_{n+1} \vee \neg P_n \vee C$.

Звідси знаходимо:

$(P_1 \wedge P_2 \wedge ... \wedge P_{n+1}) \to \neg P_n \vee C$.

(2.3.2)

Клауза 2.3.2. може бути подана в другій еквівалентній формі

$P_1, P_2, ..., P_{n+1} \Rightarrow \neg P_n, C$

(2.3.3)

Унаслідок комутативності кон'юнкції на місці посилки P_n може бути будь-яка посилка і причому не одна. Наприклад, клауза

$P_1, P_2, P_3, P_4 \Rightarrow C_1; C_2; C_3$.

може бути перетворена в іншу еквівалентну форму:

$P_1, \neg C_2, P_4, \neg C_1 \Rightarrow \neg P_2; C_3; \neg P_3$

(2.3.4)

Однак клауза 2.3.1 порівняно з 2.3.3 й іншими подібними формами типу 2.3.4 має певні переваги, які дають можливість використати її в мові логічного програмування ПРОЛОГ. Її називають хорнівською. Задовільну клаузу завжди можна звести шляхом еквівалентних перетворень до хорнівського вигляду.

Якщо символ метаімплікації "$\Rightarrow$" клаузи 2.3.1 перемістити в крайнє ліве положення, то вона перетвориться в тавтологію, а якщо в крайнє праве положення, то в суперечність: $1 \Rightarrow \neg P_1; \neg P_2; ...; \neg P_{n+1}; \neg P_n; C$ – тавтологія, $P_1, P_2, ..., P_{n+1}, P_n, \neg C \Rightarrow 0$ – протиріччя.

2.4. Аксіоматичний метод

Аксіоматичний метод перевірки тотожної істинності логічних висловлювань ґрунтується на тому, щоб серед нескінченного числа істинних

клауз знайти незалежну систему аксіом, за допомогою якої можна було

установити справедливість будь-яких клауз.

Оскільки доведення в логіці висловлювань будуються на відношенні порядку, то логіка висловлювань є розширенням логіки Буля. Тому всі істинні тотожності логіки Буля автоматично стають справедливими клаузами логіки висловлювань. Наприклад, закон склеювання логіки Буля $(A \vee B) \wedge (A \vee \neg B) = A$ можна подати такими клаузами:

$(A \vee B)$, $(A \vee \neg B) \Rightarrow A$, $A \Rightarrow (A \vee B) \wedge (A \vee \neg B)$, $1 \Rightarrow ((A \vee B) \wedge (A \vee \neg B)) \sim A$,

$A \vee B \Rightarrow (A \vee \neg B) \to A$.

Таким чином, незалежна система аксіом логіки Буля, яка складається з чотирьох законів – комутативності, асоціативності, дистрибутивності, нуля і одиниці, – автоматично стає системою аксіом і логіки висловлювань. Для визначення відношення порядку необхідне будь-яке одне елементарне висловлювання, до якого можна було зводити всі інші найбільш складні висловлювання. Таким висловлюванням може бути очевидна думка: "*Істину може висловлювати кожний*".

Використовуючи формальну мову логіки висловлювань, цю думку можна подати такою кляузою: $A \Rightarrow B \to A$, яка означає "*якщо A істинне, то джерелом цієї істинності може бути що завгодно, наприклад B*". Якщо зробити еквівалентне перетворення, то отримаємо клаузу

$A, B \Rightarrow A$. (2.4.1)

Семантика цієї клаузи теж змінилася, і матиме таке висловлювання: "*якщо раніше було встановлено, що A істинне, то істинність B не може проявитися так, щоб A стало хибним*", або "*істинність одного висловлювання B не може впливати на істинність другого висловлювання A*".

Отже, як основну аксіому логіки висловлювань, що виражає відношення порядку, можна використовувати клаузу 2.4.1.

Приклад 2.4.1. Довести аксіоматичним методом справедливість клаузи

A, $A \to B \Rightarrow B$ (2.4.2)

правилом відділення, "*Modus Ponens*".

Розв'язання. Використовуючи тотожності логіки Буля, виконаємо еквівалентні перетворення заданої клаузи $A \wedge (\neg A \vee B) \Rightarrow B$, отримаємо B, $A \Rightarrow B$, що згідно з 2.4.1 підтверджує справедливість клаузи A, $A \to B \Rightarrow B$. Тобто якщо в процесі доведення істинності будь-якої складної клаузи вдалося звести її до клаузи 2.4.2, то необхідно вважати, що доведення відбулося.

Приклад 2.4.2. Довести аксіоматичним методом справедливість гіпотетичного силогізму, який заданий такою кляузою:

$A \to B$, $B \to C \Rightarrow A \to C$. (2.4.3)

Розв'язання. Перенесемо A вліво за символ метаімплікації і отримаємо таку клаузу: A, $A \to B$, $B \to C \Rightarrow C$. Скориставшись правилом відділення 2.4.2 , отримаємо B, $B \to C \Rightarrow C$. Скориставшись другий раз правилом відділення 2.4.2 , отримаємо B, $C \Rightarrow C$, що підтверджує справедливість гіпотетичного силогізму, оскільки його виведення відповідає основній аксіомі логіки висловлювань 2.4.1.

Приклад 2.4.3. Довести аксіоматичним методом істинність така тавтології (аксіома *A1*): $1 \Rightarrow A \to (B \to A)$.

Розв'язання. Використовуючи істинні тотожності логіки Буля, виконаємо еквівалентне перетворення заданої тавтології $1 \Rightarrow \neg A \vee \neg B \vee A$ й отримаємо відношення порядку A, $B \Rightarrow A$, яке наведене в 5.4.1, що і підтверджує справедливість тавтології $1 \Rightarrow A \to (B \to A)$ – аксіома *A1*.

Приклад 2.4.4. Довести аксіоматичним методом істинність такої тавтології (аксіома *A2*): $1 \Rightarrow (A \to B \to C)) \to ((A \to B) \to (A \to C))$.

Розв'язання. Використовуючи істинні тотожності логіки Буля, виконаємо еквівалентні перетворення заданої тавтології: $A \to (B \to C)$, $A \to B$, $A \Rightarrow C$.

Користуючись правилом відділення $(B \to C)$, $B \Rightarrow C$ і ще раз правилом відділення, отримаємо відношення порядку B, $C \Rightarrow C$, що і підтверджує справедливість тавтології

$1 \Rightarrow (A \to (B \to C)) \to ((A \to B) \to (A \to C))$– аксіома *A3*.

Приклад 2.4.5. Довести аксіоматичним методом істинність такої тавтології (аксіома *A3*): $1 \Rightarrow (\neg B \to \neg A) \to ((\neg B \to B) \to B))$.

Розв'язання. Використовуючи істинні тотожності логіки Буля, виконаємо еквівалентні перетворення заданої тавтології $\neg B \to \neg A$, $\neg B \to B \Rightarrow B$, що еквівалентно клаузі $A \to B$, $B \Rightarrow B$, відповідає 2.4.2 правилу відділення, і підтверджує справедливість тавтології $1 \Rightarrow (\neg B \to \neg A) \to ((\neg B \to B) \to B))$ та аксіоми *A3*.

Приклад 2.4.6. Користуючись аксіоматичним методом, довести істинність такої клаузи: A, $B \to D$, $C \to D$, $A \to (B \vee C) \Rightarrow D$.

Розв'язання. Задану клаузу можна подати як клаузу

$\neg B \to D$, $C \to D$, $B \vee C \Rightarrow D$. Використовуючи істинні тотожності Буля, виконаємо еквівалентні перетворення останньої клаузи і отримаємо $\neg B \vee D$, $\neg C \vee D$, $B \vee C \Rightarrow D$. Застосувавши логічний закон дистрибутивності, отримаємо $(\neg B \wedge \neg C) \vee D$, $B \vee C \Rightarrow D$.

Після логічного перетворення клауза матиме такий вигляд:

$(B \vee C) \to D$, $(B \vee C) \Rightarrow D$, що відповідає правилу відділення і підтверджує істинність клаузи A, $B \to D$, $C \to D$, $A \to (B \vee C) \Rightarrow D$.

Приклад 2.4.7. Користуючись аксіоматичним методом, довести істинність клаузи A, $B \vee D \Rightarrow A \wedge C$; $B \wedge \neg C$.

Розв'язання. Виконуючи еквівалентні логічні перетворення, отримаємо A, $B \vee C$, $\neg(A \vee C)$, $\neg(B \vee \neg C) \Rightarrow 0$; A, $B \vee C$, $\neg A \vee \neg C$, $\neg B \vee C \Rightarrow 0$.

Застосовуючи закон дистрибутивності, маємо A, $C \vee (B \wedge \neg B)$, $\neg A \vee \neg C \Rightarrow 0$; A, $\neg A \vee \neg C \Rightarrow \neg C$; A, $A \to \neg C \Rightarrow \neg C$, що відповідає правилу відділення і підтверджує істинність клаузи A, $B \vee C \Rightarrow A \wedge C$; $B \wedge \neg C$

Для зручності та наглядності доведення логічних висловлювань аксіоматичним методом посилки логічного висловлювання записують у стовпчик спеціальної таблиці, де зазначаються перетворювана форма даного висловлювання і метод її отримання.

Приклад 2.4.8. Довести істинність логічного висловлювання $A \to B$, $\neg C \to \neg B$, $\neg C \Rightarrow \neg A$. аксіоматичним методом.

Розв'язання. Посилки логічного висловлювання заносимо в табл. 2.4.1 і робимо над ними перетворення, використовуючи відповідні правила і закони.

Таблиця 2.4.1

Посилки, наслідок	Тотожна формула	Як отримана
1. $A \to B$		
2. $\neg C \to \neg B$		
3. $\neg C$		
4. $\neg B$	$\neg C \to \neg B$, $\neg C \Rightarrow \neg B$	2, 3 і *MP*
5. $\neg B \to \neg A$	$A \to B \Leftrightarrow \neg B \to \neg A$	із закону контрапозиції
6. $\neg A$	$\neg B \to \neg A$, $\neg B \Rightarrow \neg A$	4, 5 і *MP*

Із табл. 2.4.1 випливає, що доведення істинності наслідку $\neg A$ логічного висловлювання отримано шляхом перетворення логічної посилки $A \to B$ на еквівалентну $\neg B \to \neg A$ через закон контрапозиції, а також застосування правила *"Modus ponens"* для відповідних посилок заданого висловлювання.

Приклад 2.4.9. Довести істинність логічного висловлювання $\neg A \to \neg B$, $(A \vee \neg B) \to C$, $\neg D \to \neg C$, $D \to (\neg E \vee \neg F) \Rightarrow \neg E \vee \neg F$ аксіоматичним методом.

Розв'язання. Посилки логічного висловлювання заносимо в табл. 2.4.2 і робимо над ними перетворення, використовуючи відповідні правила виведення.

Таблиця 2.4.2

Посилки, наслідок	Тотожна формула	Як отримана
1. $\neg A \to \neg B$	$A \vee \neg B$	
2. $(A \vee \neg B) \to C$		
3. $\neg D \to \neg C$	$C \to D$	
4. $D \to (\neg E \to \neg F)$		
5. C	$(A \vee \neg B)$, $(A \vee \neg B) \to C \Rightarrow C$	1, 2 і *MP*
6. D	C, $C \to D \Rightarrow D$	3, 5 і *MP*
7. $\neg E \vee \neg F$	D, $D \to (\neg E \to \neg F) \Rightarrow \neg E \vee \neg F$	4, 6 і *MP*

Із табл. 2.4.2 випливає, що доведення істинності наслідку $\neg E \vee \neg F$ логічного висловлювання отримано шляхом застосування правила *"Modus ponens"* для відповідних посилок заданого висловлювання.

Приклад 2.4.10. Довести істинність логічного висловлювання $\neg(\neg A \vee B)$, $\neg C \to \neg D$, $(A \wedge \neg B) \to D$, $\neg C \vee K \Rightarrow K$ аксіоматичним методом.

Розв'язання. Посилки логічного висловлювання заносимо в табл. 2.4.3 і робимо над ними відповідні перетворення.

Таблиця 2.4.3

Посилки, наслідок	Тотожна формула	Як отримана
1. $\neg(\neg A \vee B)$	$A \wedge \neg B$	
2. $\neg C \to \neg D$	$D \to C$	
3. $(A \wedge \neg B) \to D$		
4. $\neg C \vee K$	$C \to K$	
5. D	$(A \wedge \neg B)$, $(A \wedge \neg B) \to D \Rightarrow D$	1, 3 і *MP*
6. C	D, $D \to C \Rightarrow C$	2, 5 і *MP*
7. K	C, $C \to K \Rightarrow K$	4, 6 і *MP*

Із табл. 2.4.3 випливає, що доведення істинності наслідку *K* логічного висловлювання отримано шляхом застосування правила "*Modus ponens*" для відповідних посилок заданого висловлювання.

2.5. Конструктивний метод

Протилежним до аксіоматичного є конструктивний метод у логіці висловлювань, який ґрунтується на таблицях істинності. Для його розуміння побудуємо таблицю істинності для будь-якого одного прикладу. Нехай задана така легенда: "*Експедитор сказав, що він бачив шофера в кімнаті відпочинку*". Ця кімната знаходиться поряд зі складом. Стріляли у складі. Шофер заявив, що він ніякої стрілянини не чув. Висновок: якщо експедитор каже правду, то шофер веде слідство до помилки; не можуть експедитор і шофер одночасно говорити правду.

Для висловлювань з легенди введемо такі позначення:

A = "*Експедитор сказав правду*".
B = "*Шофер відпочивав у кімнаті відпочинку*".
C = "*Кімната відпочинку знаходиться поряд зі складом*".
D = "*Шофер чув стрілянину*".
E = "*Шофер сказав правду*".

Посилки слідчого мають такі наповнення:

"*Якщо експедитор сказав правду, то шофер знаходився в кімнаті відпочинку*" $P_1 = A \to B$.

"*Якщо шофер знаходився в кімнаті відпочинку, то він повинен був чути все, що робиться у складі*" $P_2 = B \to C$.

"*Якщо шофер мав можливість чути, що робиться у складі, то він чув і стрілянину*" $P_3 = C \to D$.

"*Якщо вірити шоферу, то він не чув стрілянини*" $P_4 = E \to \neg D$.

Висновки слідчого мають такі наповнення:

"*Шофер мене обманює за умови, що експедитор каже правду*" $C_1 = A \to \neg E$.

"*Експедитор і шофер одночасно кажуть правду*" $C_2 = A \wedge E$.

"*Шофер обманює, він знаходився в кімнаті відпочинку, яка дійсно розташована поряд зі складом, – це все так, але за умови, що експедитор сказав правду або що шофер знаходився у кімнаті відпочинку*":

$C_3 = (A \vee D) \to (\neg E \wedge B \wedge C)$.

"*Шофер обманює, він чув стрілянину, а кімната відпочинку дійсно розташована поряд зі складом – це все так, але за умові, що експедитор сказав правду або що шофер знаходився в кімнаті відпочинку*": $C_4 = (A \vee B) \to (\neg E \wedge D \wedge C)$.

Будуємо таблицю істинності, табл. 2.5.1, в якій під *P* розуміють спільну причину всіх посилок P_1, P_2, P_3, P_4, тобто їх кон'юнкцію: $P = P_1 \wedge P_2 \wedge P_3 \wedge P_4$.

Таблиця 2.5.1

№	A	B	C	D	E	P_1	P_2	P_3	P_4	P	C_1	C_2	C_3	C_4
0	X	X	X	X	X	I	I	I	I	I	I	X	I	I
1	I	X	X	X	X	X	I	I	I	X	I	X	X	X
2	X	I	X	X	X	I	X	I	I	X	I	X	I	X
3	I	I	X	X	X	I	X	I	I	X	I	X	X	X
4	X	X	I	X	X	I	I	X	I	X	I	X	I	I
5	I	X	I	X	X	X	I	X	I	X	I	X	X	X
6	X	I	I	X	X	I	I	X	I	X	I	X	I	X
7	I	I	I	X	X	I	I	X	I	X	I	X	I	X
8	X	X	X	I	X	I	I	I	I	I	I	X	X	I
9	I	X	X	I	X	X	I	I	I	X	I	X	X	X
10	X	I	X	I	X	I	X	I	I	X	I	X	X	X
11	I	I	X	I	X	I	X	I	I	X	I	X	X	X
12	X	X	I	I	X	I	I	I	I	I	I	X	X	I
13	I	X	I	I	X	X	I	I	I	X	I	X	X	I
14	X	I	I	I	X	I	I	I	I	I	I	X	I	I
15	I	I	I	I	X	I	I	I	I	I	I	X	I	I
16	X	X	X	X	I	I	I	I	I	I	I	X	I	I
17	I	X	X	X	I	X	I	I	I	X	X	I	X	X
18	X	I	X	X	I	I	X	I	I	X	I	X	I	X
19	I	I	X	X	I	I	X	I	I	X	X	I	X	X
20	X	X	I	X	I	I	I	X	I	X	I	X	I	I
21	I	X	I	X	I	X	I	X	I	X	X	I	X	X
22	X	I	I	X	I	I	I	X	I	X	I	X	I	X
23	I	I	I	X	I	I	I	X	I	X	X	I	X	X
24	X	X	X	I	I	I	I	I	X	X	I	X	X	I
25	I	X	X	I	I	X	I	I	X	X	X	I	X	X
26	X	I	X	I	I	I	X	I	X	X	I	X	X	X
27	I	I	X	I	I	I	X	I	X	X	X	I	X	X
28	X	X	I	I	I	I	I	I	X	X	I	X	X	I
29	I	X	I	I	I	X	I	I	X	X	X	I	X	X
30	X	I	I	I	I	I	I	I	X	X	I	X	X	X
31	I	I	I	I	I	I	I	I	X	X	X	I	X	X

Клауза є істиною, якщо істинні значення слідства C покривають всі істинні значення спільної причини P, тобто істинні значення спільної причини утворюють підмножину істинних значень слідства. Ця необхідність виконується для наслідку C_1, оскільки $P=\{0, 8, 12, 14, 15, 16\}\subset\{0, ..., 31\}=C_1$.

Оскільки $P_1,\ P_2,\ P_3,\ P_4 \Rightarrow C_1$, то тавтологія, складена з цих посилок, буде дорівнювати $1 \Rightarrow \neg P_1,\ \neg P_2,\ \neg P_3,\ \neg P_4,\ C_1$, а суперечність матиме вигляд

$P_1,\ P_2,\ P_3,\ P_4,\ \neg C_1 \Rightarrow 0.$

Якщо наслідок C_1 замінити на C_2, то у всіх зазначених випадках причинно-наслідкові відношення порушуються і клауза перетворюється в хибне метависловлювання.

Для наслідку C_3 в рядках 8 і 12 із множини рядків $\{0, 8, 12, 14, 15, 16\}$, стоїть хибне значення, тому умовні причинно-наслідкові відношення не виконуються і C_3 є хибним наслідком

$P=\{0, 8, 12, 14, 15, 16\}\subset\{0, 2, 4, 6, 7, 14, 15, 16, 18, 20, 22\}=C_3$

Для наслідку C_4 всі його позначення істинності покривають істинності спільної причини P, тобто наслідок C_4 є істинним висновком:

$P=\{0, 8, 12, 14, 15, 16\}\subset\{0, 4, 8, 12, 13, 14, 15, 16, 20, 24, 28\}=C_4.$

У загальному випадку побудуємо всі спільні рядки подій. У нашому випадку таких рядків 6, вони відповідають 0, 8, 12, 14, 15, 16. Їх об'єднання дає граничний випадок умови виконання причинно-наслідкових відношень

$\neg A, \neg B, \neg C, \neg D, \neg E;\ \neg A, \neg B, \neg C, D, \neg E;\ \neg A, \neg B, C, D, \neg E;$
$\neg A, B, C, D, \neg E;\ A, B, C, D, \neg E;\ \neg A, \neg B, \neg C, \neg D, E.$

Але це є ДДНФ, яка відповідає конкретній причині P. Всі можливі покриття шести значень істинності дає множина істиннісних наслідків. Так, висновок $C_1 = \neg A;\ \neg E,\ \ C_4 = \neg A,\ \neg B;\ C, D, \neg E \div$ покривають усі шість умов виконання причинно-наслідкових відношень і тому вони є істинні. Що стосується двох інших висновків $C_2 = A, E,$ і $C_3 = \neg A, \neg D; C, B, \neg E$, то вони не покривають усі шість умов виконання причинно-наслідкових відношень, і тому є хибними наслідками.

Для знаходження істинних наслідків із заданих причин знайдемо мінімальну нормальну форму (МНФ), мінімальне і трансверсальне покриття.

Для знаходженням МНФ за відомими ДДНФ використаємо метод Вейча або Карно, внаслідок чого отримаємо таку МНФ:

$\neg A, \neg B, \neg C, \neg D;\ \neg A, \neg B, D, E;\ B, C, D, \neg E.$

Мінімальне покриття – це покриття з найменшим числом термів, що є висновком C_1. До нього входять **два вирішальних** висловлювання, пов'язаних із правдивістю касира A і правдивістю експедитора E. Всі інші твердження B, C, D є другорядними і можуть виступати в результуючому висновку спільно з A і E.

2.6. Метод доведення від супротивного

Цей метод доведення істинності висловлювання полягає в такому: допускають, що істинним є заперечення того висловлювання, яке необхідно довести (наслідок); потім через причини (посилки) намагаються дійти до суперечності. Якщо це відбулося, то досліджуване логічне висловлювання істинне, якщо ні – хибне.

Приклад 2.6.1. Довести істинність (хибність) логічного висловлювання $A \vee B,\ B \to C,\ A \to D \Rightarrow C \vee D$ методом від супротивного.

Розв'язання. Для зручності доведення логічного висловлювання представимо його в такому вигляді:

$A \vee B$
$B \to C$
$\underline{A \to D}$
$\Delta\ C \vee D,$

де символ Δ позначає " наслідок ".

Припустимо, що посилки істинні, а наслідок - хибний. Якщо наслідок хибний, тоді $C \vee D$- хибне. Але якщо $C \vee D$- хибне, то C і D також хибне. Якщо C і D хибні, а посилки $A \to D$і $B \to C$ істинні, то A і B також хибні. Але якщо A і B хибні, а посилка $A \vee B$ за умовою істинна, то ми дійшли до суперечності, з якої випливає, що дане висловлювання істинне.

Приклад 2.6.2. Довести істинність (хибність) логічного висловлювання $A \to B,\ B \to C,\ C \Rightarrow A$ методом від супротивного.

Розв'язання. Для зручності доведення логічного висловлювання представимо його в такому вигляді:

$A \to B$
$B \to C$
$\underline{C}$
$\Delta\ A$

Припустимо, що посилки істинні, а наслідок – хибний. Тоді C – істинне, а A – хибне. Але якщо A – хибне, а C – істинне, то перші дві посилки $A \to B$ і $B \to C$будуть також істинні, і це не веде до суперечності. Тому дане логічне висловлювання – хибне.

Приклад 2.6.3. Довести істинність (хибність) логічного висловлювання $\neg A \vee B,\ C \vee \neg B,\ A,\ D \Rightarrow C \vee D$ методом від супротивного.

Розв'язання. Для зручності доведення логічного висловлювання представимо його в такому вигляді:

$\neg A \vee B$
$C \vee \neg B$
A
$\underline{D}$
$\Delta\ C \vee D.$

Припустимо, що посилки істинні, а наслідок – хибний. Тоді C і D – хибні, а A – істинне. Але якщо A - істинне, то і B за умовою повинне бути істинним. А якщо це так, то посилка $C \vee \neg B$буде хибна, що призводить до протиріччя, з якого випливає істинність даного висловлювання.

2.7. Метод резолюцій

У цьому методі для доведення істинності висловлювань використовують аксіоми порядку. Суть методу зводиться до того, що дві посилки диз'юнкти з протилежними термами завжди можливо склеїти в один остаточний диз'юнкт, у якому вже не буде протилежних термів, наприклад $A \vee B,\ C \vee \neg B \Rightarrow A \vee C,$

де A і C – задовільні терми або цілі диз'юнкти з будь-яким набором термів, включаючи хибність; B і $\neg B$ - задовільні терми. Якщо послідовно застосувати метод резолюцій до досліджуваної клузи, то в результаті цього отримаємо зменшення числа букв, у тому числі й до їх повного зникнення.

Метод резолюцій за своєю суттю заміняє аксіому порядку, оскільки вона сама може бути доведена в рамках методу резолюцій, наприклад:

$$A,\ B \Rightarrow A,\ A,\ B,\ \neg A \Rightarrow 0,\ 0,\ B \Rightarrow 0.$$

Необхідно відмітити, що посилка B тут не використовується. Це може бути допустимо, оскільки необов'язково використовувати всі посилки (їх число може бути збитковим) – головне в цьому отримання результату. При доведенні клузи методом резолюцій необхідно її подати в нормальній кон'юктивній формі, а необхідною умовою доведення є отримання нуля, що є підставою істинності розглядуваної клузи.

Приклад 2.7.1. Довести істинність клузи $D \to F,\ A \to (E \to D),\ C \to (B \to A) \Rightarrow E \to (\neg C \vee F) \vee \neg B$ методом резолюцій.

Розв'язання. Подамо клаузу в КНФ:

$$\neg D \vee F,\ \neg A \vee \neg E \vee D,\ \neg C \vee \neg B \vee A \Rightarrow \neg E \vee \neg C \vee F \vee \neg B.$$

Перенесемо всі посилки в ліву частину клузи

$$\neg D \vee F,\ \neg A \vee \neg E \vee D,\ \neg C \vee \neg B \vee A,\ E,\ C,\ \neg F,\ B \Rightarrow 0.$$

Випишемо посилки клузи у стовпчик і склеїмо їх по черзі:

1. $\neg D \vee F$,	8. $\neg A \vee \neg E \vee F$	(1, 2),
2. $\neg A \vee \neg E \vee D$,	9. $\neg C \vee A$	(3, 7),
3. $\neg C \vee \neg B \vee A$,	10. $\neg C \vee \neg E \vee F$	(8, 9),
4. E,	11. $\neg C \vee F$	(4, 10),
5. C,	12. F	(5, 11),
6. $\neg F$,	13. 0	(6, 12).
7. B,		

Із черги склеювання посилок випливає, що ця клауза істинна.

Приклад 2.7.2. Довести істинність клузи $A \to B,\ C \to D,\ A \vee C,\ A \vee \neg D,\ C \to \neg B \Rightarrow E \to (A \vee B) \to (A \wedge B)$ методом резолюцій.

Розв'язання. Подамо клаузу в КНФ:

$$\neg A \vee B,\ \neg C \vee D,\ A \vee C,\ \neg A \vee \neg D,\ \neg C \vee \neg B \Rightarrow \neg (A \vee B) \vee A \wedge B.$$

Перенесемо всі посилки в лівий бік клузи:

$$\neg A \vee B,\ \neg C \vee D,\ A \vee C,\ \neg A \vee \neg D,\ \neg C \vee \neg B,\ A \vee B,\ \neg A \vee \neg B \Rightarrow 0.$$

Випишемо всі по порядку посилки клузи і склеїмо їх по черзі:

1. $\neg A \vee B$,	8. $A \vee D$	(2, 3),	15. 0 (11, 14).
2. $\neg C \vee D$,	9. $B \vee D$	(1, 8),	
3. $A \vee C$,	10. $B \vee \neg D$	(4, 6),	
4. $\neg A \vee \neg D$,	11. B	(9, 10),	
5. $\neg C \vee \neg B$,	12. $\neg C$	(5, 11),	
6. $A \vee B$,	13. A	(3, 12),	
7. $\neg A \vee \neg B$,	14. $\neg B$	(7, 13),	

Із черги склеювання посилок випливає, що ця клауза істинна.

Питання для самоперевірки:

1. Які умови повинні бути виконані під час визначення формальної аксіоматичної теорії *L*?
2. Що називають аксіомами теорії *L*?
3. Що називають гіпотезами виведення у теорії *L*?
4. Що називають наслідком у теорії *L* і як його позначають?
5. Що називають теоремою в теорії *L*?
6. Яким способом задають формальну аксіоматичну теорію *L*?
7. Скільки правил виведення використовують у теорії *L*?
8. Які аксіоми використовують у теорії *L* для доведення будь-яких формул у логіці висловлювань? Запишіть їх.
9. Які ви знаєте інші аксіоматичні числення висловлювань? Запишіть їх.
10. Сформулюйте теорему дедукції.
11. Для яких цілей застосовується теорема дедукції?
12. Яке основне правило виведення використовують у теоремі дедукції?
13. Чи використовують гіпотези, аксіоми та самі формули при доведенні теореми дедукції?
16. На чому ґрунтується аксіоматичний метод?
17. Яка аксіома використовується як основна при аксіоматичному методі?
18. Яке правило найчастіше використовується в аксіоматичному методі?
19. Назвіть тотожності та закони алгебри Буля, які використовуються в аксіоматичному методі.
20. Сформулюйте означення конструктивного методу.
21. Яка відмінність конструктивного методу від аксіоматичного?
22. Що розуміють під мінімальним покриттям?
23. Сформулюйте метод доведення висловлювань від супротивного.
24. У чому різниця методів доведення логічних висловлювань від супротивного і аксіоматичного?
25. Сформулюйте кроки алгоритму, які покладені в основу методу резолюцій.
26. Яка аксіома використовується в методі резолюцій?
27. Чим відрізняється метод резолюцій від аксіоматичного і конструктивного методів?

Задачі для самостійного розв'язування

1. Довести, що для будь-якої формули A в теорії L виконується умова $\vdash_L (\neg A \to A) \to A$.
2. Довести, що для будь-якої формул A, B і C у теорії L виконується умова $A \to B,\ B \to C \vdash_L A \to C$.
3. Довести, що для будь-яких формул A, B у теорії L виконується умова $\vdash_L (\neg B \to \neg A) \to (A \to B)$.
4. Довести, що для будь-яких формул A, B, C у теорії L виконується умова $A \to (B \to C) \vdash_L B \to (A \to C)$.
5. Довести, що такі формули є теоремами теорії L:
   а) $((A \to B) \to A) \to A$;
   б) $A \to (B \to (A \wedge B))$;
   в) $(A \to B) \to ((B \to C) \to (A \to C))$.
6. Довести, що для будь-яких формул A, B і C такі формули є теоремами теорії L:
   а) $((A \vee B) \wedge (A \to C) \wedge (B \to C)) \to C$;
   б) $A \to (B \to C) \sim (A \wedge B) \to C$.
7. У логічному висловлюванні "*Будь-яке висловлювання є хибним*" зняти суперечність.
8. У логічному висловлюванні "*В одному селі жив перукар. Він голив тих жителів села, хто не голився сам*". Виразити семантику цієї суперечності формальною мовою і записати його за допомогою двох метависловлювань.
9. Довести такі клузи аксіоматичним методом:
   а) $(A \to C) \to (\neg A \wedge B) \Rightarrow A \vee B$;
   б) $(A \wedge B) \to C \Rightarrow A \to (B \to C)$;
   в) $\neg C, A \vee B \Rightarrow (B \to C) \to A$;
   г) $A \to (C \to B),\ D \to A,\ C \Rightarrow D \to B$;
   д) $A \vee B,\ A \vee B,\ B \to D,\ D \to C \Rightarrow C$;
   е) $A \to C,\ A \vee B,\ B \to D,\ D \to C \Rightarrow C$;
   є) $A \to B,\ B \vee C,\ C \to A,\ B \to C \Rightarrow A \wedge B$;
   ж) $A \to (B \to \neg C),\ \neg A \to B,\ \neg A \to (\neg B \to C) \Rightarrow C;\ B$.
10. Для заданих легенд побудувати клузи і довести їх істинність за допомогою конструктивного методу:

а) "*Якщо в одному місці щось вибуває, то в іншому щось прибуває і навпаки.*" Якщо в космічному просторі є "*чорна дірка*", то в неї все провалюється, тобто в її оточенні щось вибуває. Якщо є "*біла дірка*", то із неї в навколишній простір повинна прибувати речовина. Якщо є "*чорна дірка*", то її неможливо побачити, оскільки вона не випромінює світло. Астроном подивився і нічого не побачив. Отже, "*біла дірка*" є. Це хибний умовивід. Істинним заключенням є, наприклад, таке: "*Якщо є "чорна дірка", то десь у космічному просторі повинна неодмінно з'явитися речовина.*"

б) "*Усе живе повинне бути чутливим. Будь-яке матеріальне тіло займає деякий об'єм. Якщо щось займає об'ємний простір і може бути чутливим, то це щось є не що інше, як живий організм. Нехай є щось живе, але не є організмом. Тоді випливає наслідок, що це щось нематеріальне.*"

11. Довести істинність (хибність) логічних висловлювань у таких клаузах методом від супротивного:

а) $A\to B$, $A\to C$, $C\vee B$ / A	б) $A\to B$, $\neg B\to\neg C$, $C\to D$, $D\vee B$ / $A\vee C$	в) $\neg A\vee B$, $\neg B\to C$, $\neg C$ / $\neg A$	г) $A\vee B$, $\neg B\to\neg C$, $\neg C$ / A
д) $A\to B$, $A\to C$, $\neg(\bar{A}\vee B)$ / $\neg A$	є) $\neg A\vee\neg B$, $C\vee\neg B$, $\neg A$ / $C\vee\neg A$	ж) $A\vee B$, $B\to C$, $A\to D$ / $C\vee D$	з) $A\to B$, $B\to C$ / C

12. Кожну із наведених клауз довести методом резолюцій:

а) $(A\to C)\to(\neg A\wedge B)\Rightarrow A\vee B$;

б) $A\to(B\to\neg C),\ \neg A\to B,\ \neg A\to(\neg B\to C)\Rightarrow C;B$;

в) $A\to C,\ D\to F,\ B\to E,\ \neg D\to\neg C,\ A\to B\Rightarrow A\to(E\wedge F)$;

г) $E\to F,\ C\to(D\to E),\ (A\to B)\to C\Rightarrow D\to(A\vee F)$;

д) $A\to B,\ B\vee C,\ C\to A,\ B\to C\Rightarrow A\wedge B$;

Коментарі. У цьому розділі формальна теорія L і теорема дедукції викладені відповідно до [14]. Більш детальне викладення цього матеріалу можна знайти в класичних підручниках, наприклад [12]. У цьому розділі побудова доведень у логіці висловлювань викладена більш удосконалено і наочно порівняно з [8,27], що підтверджується значною кількістю різноманітних прикладів.

Тема 3

- Логіка предикатів першого порядку
  - 3.1. Основні поняття логіки предикатів
  - 3.2. Квантори
  - 3.3. Формули логіки предикатів
  - 3.4. Рівносильність формул логіки предикатів
  - 3.5. Закони і тотожності логіки предикатів
  - 3.6. Властивості числення предикатів першого порядку

Логіка предикатів першого порядку

3.1. Основні поняття логіки предикатів

В розділу 1 формалізація мови у логіці висловлювань здійснюється шляхом розбиття мовних повідомлень на окремі неподільні речення (атоми), а їх змістовне об'єднання – за допомогою символів ~ , → , $\wedge$, $\vee$, ¬ . Внутрішня структура речень при цьому не враховується. Наприклад, для умовиводу *"Деякі люди геніальні. Сократ – людина. Отже, він геніальний"*, інтуїтивно зазначений логічний висновок стосується деяких індивідуумів, до яких Сократ міг і не належати. Тобто, в цьому висловлюванні не врахована внутрішня змістова особливість узагальнення *" деякі "*. А якщо розглянути умовивід *„Кожна людина смертна. Оскільки Сократ – людина, то він смертний"*, то інтуїтивно зазначений логічний висновок є коректним.

Розглянемо цей умовивід більш детально. Для цього введемо атоми: *A – "кожна людина смертна"; B – "Сократ - людина" ; C – "Сократ смертний"*.

Тоді умовивід буде відповідати формулі логіки висловлювань $A\wedge B\to C$, або перетворивши яку в нормальну форму, отримаємо

$A\wedge B\to C=\neg(A\wedge B)\vee C=\neg A\vee\neg B\vee C$.

Остання формула на інтерпретації (*I, I, X*) хибна (*X*), що свідчить про те, що вона не є загальнозначущою. А це означає, що у рамках логіки висловлювань *C* не є логічним наслідком *A* і *B*. Така обмеженість можливостей цієї формалізації пов'язана з тим, що в атомі *A* не врахована внутрішня змістова особливість узагальнення *"кожний"*.

Теорія предикатів враховує внутрішню структуру речень і ґрунтується на тому, що в цих реченнях представлені об'єкти мають певні властивості, або знаходяться між собою у певному відношенні. Наприклад, в висловленні *" Сократ – людина "*, підмет *" Сократ "* є об'єктом, а присудок *" людина "* виражає деяку його властивість. Головним для логіки предикатів є саме друга складова речення, яка фіксується, а значення об'єкту пропонується позначити деякою змінною величиною. Таким чином, можна розглянути речення *" x – людина "*, яке не являється висловлюванням, а є висловлювальною формою, підстановка в яку замість параметра *x* об'єктів (значень) з деякої множини M перетворює форму в висловлювання.

Нехай M – непорожня підмножина декартового добутку $M_1xM_2x\ldots xM_n$

множин (n $\geq$ 1).

Означення 3.1.1. n – місним предикатом, заданим на множині M, називається речення, що містить n змінних $x_1,x_2,\ldots,x_n$ і стає висловленням при кожній заміні їх елементами з відповідних множин $a_1,a_2,\ldots,a_n$.

n – місний предикат будемо позначати $P(x_1, x_2,\ldots, x_n)$, змінні $x_1,x_2,\ldots,x_n$ будемо називати *предметними змінними*, а елементи множин, які ці змінні пробігають $((a_1,a_2,\ldots,a_n)\in M)$ – *предметними константами.* Наприклад, над множиною натуральних чисел речення *" x – просте число "* є одномісний предикат, а речення *" x кохає y "* є двомісний предикат на множині людей.

Таким чином, будь-який n – місний предикат можна ототожнити з логічною функцією n аргументів, яка приймає значення із множини *{I, X}*, тобто

$P(x_1,x_2,\ldots,x_n): M_1xM_2x\ldots xM_n\to\{I,X\}$.

Це пояснюється тим, що в математичній логіці нас менше цікавить змістовна суть предиката, а більш важливо знати яке значення істинності ставиться у відповідність за допомогою даного предиката тій чи іншій послідовності елементів.

Таким чином, предикат $P(x_1,x_2,\ldots,x_n)$ буде визначений, якщо: задана деяка множина *M*, яку називають областю визначення предиката (предметна область); задана область значень (фіксована множина {I,X }); зазначено правило, за допомогою якого кожному елементу, що взятий у предикатній області, ставиться у відповідальність один із двох елементів із області значень. Предикат, що не має предметних змінних (n=0) є висловлюванням або нульмісним предикатом; якщо *n*=1, то предикат відповідає властивості; якщо *n*=2, то предикат є бінарним відношенням; якщо *n*=3, то предикат – тернарне відношення і т.д.

Приклад 3.1.1. Представити предикатами речення: *"x – ціле число"; "x ділиться на y"*.

Розв'язання. Дії або властивості цих речень оберемо як назву предикатів: ЦІЛЕ, ДІЛИТЬСЯ. Тоді задані висловлювання можна записати у вигляді предикатів таким чином: ЦІЛЕ(*x*), ДІЛИТЬСЯ(*x, y*). Перший предикат є одномісним і виражає деяку властивість чисел, а другий двомісним і виражає бінарне відношення подільності на множині чисел.

Над предикатами можна виконувати звичайні логічні операції. Результатом цих операцій будуть нові предикати. Наприклад, нехай *P(x)* позначає предикат *"x ділиться на 2"*, а *Q (x)* – предикат *"x ділиться на 3"*. Тоді вираз

P(x) $\wedge$*Q (x)* позначає предикат *"x ділиться на 2 та x ділиться на 3"*, тобто позначає предикат *"x ділиться на 6"*.

За допомогою логічних операцій можна будувати як завгодно складні предикати. Для побудови формул логіки предикатів застосовуються:

- *предметні змінні* $x_1,x_2,\ldots,x_n$;
- *предметні константи* $a_1,a_2,\ldots,a_n$;
- *функціональні символи* f_i^n , де i =1,2,... вказує порядковий номер символу, а n =1, 2, ... вказує на кількість аргументів;
- *предикатні символи* P_i^n, де i=1,2,...вказує порядковий номер символу, а n =0, 1, 2, ... вказує на кількість аргументів;
- *знаки логічних операцій* ~ , → , $\wedge$, $\vee$, ¬, ↔;
- *квантори* $\forall$, $\exists$;
- *знаки пунктуації* – ліва і права дужки та кома.

Серед слів, записаних за допомогою вказаних вище символів, виділяють терми і формули, що визначаються індуктивним чином.

Означення 3.1.2. 1) будь-яка предметна змінна або предметна константа є термом; 2) якщо *f* - функціональний символ, а $t_1, t_2, \ldots, t_n$ – терми, то $f(t_1, t_2, \ldots, t_n)$ є терм. 3) Ніяких інших термів, крім породжених за допомогою зазначених вище правил, не існує.

Приклад 3.1.2. Записати у вигляді предикатів такі речення: *"Сестра Миколи"*, *"Студенти складають іспит"*, *"Число x більше числа x - 1"*.

Розв'язання 1. Речення *"Сестра Миколи"* не може бути *"істинним"* або *"хибним"*, тому його не можна зобразити у вигляді предиката. Дане речення є деяким елементом предметної області, яке відповідає множині людей.

2. Речення *"Студенти складають іспит"* може набувати значення *"істина"* або *"хибність"*, тому його можна записати у вигляді предиката. У структурі речення можна виділити присудок *"складають"*, підмет *"студенти"* і додаток *"іспит"*. Підмет і додаток можна розглядати як предметні константи, а присудок – як нульмісний предикат. Тому речення *"Студенти складають іспит"* можна записати у вигляді нульмісного предиката таким чином: СКЛАДАТИ (студенти, іспит).

3. Присудком у цьому реченні є слово *"більше"*. Підмет *"x"* і додаток *"x - 1"* зобразимо у вигляді термів. Терм *"x - 1"* має внутрішню структуру, оскільки в ньому присутній функціональний символ " мінус ". Виходячи із цього речення набере вигляду двомісного предиката: БІЛЬШЕ((*x,1*), мінус(*x,1*)), де *x* – предметна змінна, а *1* – константа.

Приклад 3.1.3. Предикат "ДОРІВНЮВАТИ (*x* ,7)" записати природною мовою.

Розв'язання. У предикаті 7 є константа, а *x* – предметна змінна, тому він відповідає твердженню природної мови - *x дорівнює* 7.

3.2. Квантори

Окрім застосування до предикатів " звичайних " логічних операцій (~ , → , ∧ , ∨ , □, ↔) та операції підстановки замість предметних змінних їх конкретних значень, в логіці предикатів використовують так звані операції зв'язування квантором (операції квантифікації). Квантори вперше були введені саме в рамках класичної математичної логіки.

Означення 3.2.1. Квантором загальності називають знак $\forall$, під впливом якого предикат *P (x)* , визначений на множині *M*, набуває істинного значення для всіх $x \in M$, і позначають це як $\forall x\, P(x)$.

Означення 3.2.2. Квантором існування називають знак $\exists$, під впливом якого предикат *P (x)*, визначений на множині *M*, набуває істинного значення для деяких $x \in M$, і позначають це як $\exists x\, P(x)$.

Знак квантора загальності $\forall$ є перевернутою буквою "*A*", що є першою буквою англійського слова "*All*", що означає " *всі* ", а знак квантора існування $\exists$ є перевернутою літерою *"E"*, що є першою буквою англійського слова " *Exist* " і означає " *існування* ". Квантор $\forall$ у природній мові читається як " всі ", "кожен", "всякий", "який би не був". Квантор $\exists$ - "існує (бодай один) ", "існують", "знайдеться (бодай один) ", "знайдуться", "деякі".

Якщо квантор застосовується до предикату, то кажуть, що він *навішується* на формулу. Наприклад, $\forall x_1\, P(x_1, x_2, \ldots, x_n)$ – на n – місний предикат навішений квантор загальності, при цьому він зв'язує змінну x_1 , а на інші змінні його дія на розповсюджується. Змінна x_1 в цьому випадку називається **пов'язаною**, всі інші змінні є **вільними**. В результаті операції квантифікації місність предикату зміниться (в даному випадку предикат стане n-1 – місним).

Важливу роль в логіці предикатів відіграє поняття **області дії квантора**. Область дії квантора – це предикат, на який даний квантор навішується. Як правило, вона виділяється дужками.

Наприклад, у предиката $\forall$ x $\exists$ y (x<y ∧ z>0)↔(x=z) змінна *z* вільна, змінна *x* частково вільна, бо має два зв'язаних і одне вільне входження, а змінна *y* пов'язана.

Якщо предикат P(x) не містить інших змінних, окрім □, вирази $\exists$ x P(x) та $\forall$ x P(x) є реченнями, що виражають істинні або хибні висловлювання.

Приклад 3.2.1. Висловлювання: " *Всі студенти складають іспити* " і " *Деякі студенти складають іспити на відмінно* " представити засобом логіки предикатів.

Розв'язання. Введемо предикати: *P(x)* – " *x – складає іспити* ", *Q(x)* – " x – *складає іспити на відмінно* ", x□ M, де M – множина студентів. Тоді шукані представлення матимуть вигляд $\forall x\, P(x)$ і $\exists x\, Q(x)$.

Приклад 3.2.2. Для предметної області множини дійсних чисел записати засобом логіки предикатів такі твердження: "*Існує число, квадрат якого дорівнює 25*"; „ *Для всіх x є правильним, що* $(x+2)^2 = x^2 + 4x + 4$ ".

Розв'язання. Введемо предикат P(*x, y*) – " x=y ", який є істинним тоді, коли значення змінної *x* дорівнює значенню *y*. У цьому випадку, використовуючи квантори, можна записати:

$\exists x\, P(x^2, 25)$; $\forall x\, P((x+2)^2, x^2 + 4x + 4)$.

3.3. Формули логіки предикатів

Означення 3.3.1. Формулою у логіці предикатів називають формулу, яка задовольняє такі індуктивні визначення.

1. Якщо *P* - символ *n-місного* предиката, а $t_1, t_2, \ldots, t_n$ – терми, то

$P(t_1, t_2, \ldots, t_n)$ – формула, яку називають **атомарною** (елементарною). Всі предметні змінні атомарних формул – вільні, пов'язаних змінних немає.

1. 2. Якщо *P* – формула, то ¬ *P* – також формула. Вільні й пов'язані змінні формули ¬ *P* – це відповідно вільні та пов'язані змінні формули *P* .
2. 3. Якщо F і G – формули і немає таких предметних змінних, які були б пов'язаними в одній формулі, але вільними в іншій, тоді $(F \vee G)$, $(F \wedge G)$, $(F \to G)$, $(F \sim G)$ є теж формулами, в яких статус змінних зберігається.
3. 4. Якщо *P* – формула, яка містить вільну змінну *x*, то $\forall x\, P$ і $\exists x\, P$ також є формулами. Змінна *x* у цих формулах стає пов'язаною. Статус інших змінних зберігається.
4. 5. Ніяких інших формул, крім породжених зазначеними вище правилами, не існує.

Приклад 3.3.1. Визначити, які із виразів є формулами логіки предикатів:

- $P(x_1, x_3, x_5)$;
- $\forall x_1 \exists x_2\, P(x_1, x_2, x_3) \to \forall x_1 P(x_1, x_4)$;.
- $\exists x_1\ \forall x_3\, (P(x_1, x_3) \wedge P(x_3, x_4)$

Розв'язання. Вираз а) є атомарною формулою, в якій x_1, x_3, x_5 - вільні змінні. Вираз б) є формулою, в якій x_1, x_2 - пов'язані змінні, а x_3 , x_4 – вільні змінні.

Вираз в) не є формулою, оскільки порушено правило 3 означення 3.3.1.

Приклад 3.3.2. Визначити, які змінні є пов'язаними, а які вільними у таких формулах:

1. $P(x,y,z)$,; $\forall y\, (P(x) \to \exists x\, Q(x,y))$; $\exists y\, (P(y) \to \forall y\, Q(x,y))$.

Розв'язання. Усі три змінні у формулі а) є вільними. У формулі б) змінна *y* є пов'язаною, а змінна *x* – і пов'язаною, і вільною. У формулі в) змінна *x* є вільною, а змінна *y* пов'язаною.

Означення 3.3.3. Інтерпретацією формули логіки предикатів називається система, яка складається з непорожньої множини M ⊂ $(M_1 x M_2 x \ldots x M_n)$, яку називають областю інтерпретації, і відповідності, яка кожному предикатному символу P_i^n зіставляє певний n – місний предикат, заданий на множині M, кожному функціональному символу f_i^n – деяку n – арну алгебраїчну операцію, кожній константі α_i– деякий конкретний елемент із M.

Для кожної інтерпретації на області D формула *F* може набувати істинного *I*, або хибного *X* значення згідно з такими правилами.

1. 1. Якщо задані значення формул *F* і *G*, то значень істинності формул $(F \vee G)$, $(F \wedge G)$, $(F \to G)$, $(F \sim G)$ можна визначити за допомогою таблиць істинності відповідних логічних операцій.
2. 2. Формула $\forall x\, F$ набуває значення *I*, якщо *F* набуває значення *I* для кожного *x* із предикатної області D, у протилежному випадку - *X*.
3. 3. Формула $\exists x\, F$ набуває значення *I*, якщо *F* набуває значення *I* для деякого *x* із предикатної області D, у протилежному випадку - *X*.

Означення 3.3.4. Формула логіки предикатів називається :

1. істинною в даній інтерпретації, якщо вона виконується на будь-якому наборі елементів з області і інтерпретації;
2. хибною в даній інтерпретації, якщо вона не виконується на жодному наборі елементів з

області інтерпретації;
3. виконуваною в даній інтерпретації, якщо вона виконується хоча б на одному наборі елементів з області інтерпретації;
4. спростовною в даній інтерпретації, якщо вона не виконується хоча б на одному наборі елементів з області інтерпретації.

Приклад 3.3.3. Побудувати інтерпретацію формул: $P(x_1, x_2)$;

$\forall x_2 P(x_1, x_2)$; $\exists x_2 \forall x_1 P(x_1, x_2)$.

Розв'язання. Введемо область інтерпретації – множину цілих додатних чисел Z_+, а замість $P(x_1, x_2)$ введемо предикат "$x_1 \geq x_2$". Перша формула – це саме предикат, побудований на Z_+. Вона є виконуваною і спростовною. Друга формула буде виражати одномісний предикат, вона є хибною. Третя формула – це істинне висловлювання яке стверджує про існування найменшого цілого додатного числа.

3.4. Рівносильність формул логіки предикатів

Нехай формули F і G мають одну й ту саму множину вільних змінних (зокрема порожню).

Означення 3.4.1. Формули F і G є рівносильними в заданій інтерпретації, якщо на будь-якому наборі значень вільних змінних вони набувають однакових значень. Формули F і G є рівносильними на множині M, якщо вони рівносильні у всіх інтерпретаціях, заданих на множині M. Формули F і G є рівносильними в логіці предикатів, якщо вони рівносильні на всіх множинах, тоді $F \equiv G$.

Приклад 3.4.1. Визначити рівносильність формул

$F = P(x_1, x_2) \wedge P(x_1, x_3)$ і $G = P(x_1, x_2) \wedge P(x_2, x_3)$, які задані на множині $M = \{a,b\}$ предикатами $Q_1(x,y)$ та $Q_2(x,y)$, наведеними в табл. 3.4.1 і 3.4.2.

Таблиця 3.4.1

x	y	$Q_2(x,y)$
a	a	I
a	b	X
b	a	X
b	b	I

Таблиця 3.4.2

x	y	$Q_2(x,y)$
a	a	I
a	b	I
b	a	X
b	b	X

Розв'язання. Якщо областю інтерпретації є множина M, а в формулах F і G предикатному символу P зіставляється предикат $Q_1(x,y)$, то формули F і G рівносильні в заданій інтерпретації тому, що вони набувають значення I тільки на двох наборах змінних (a,a,a) та $(b,b,b,)$, на інших наборах – X.

Якщо областю інтерпретації є множина M, а P інтерпретується як предикат $Q_2(x,y)$, то формули F і G – нерівносильні в заданій інтерпретації тому, що формула F в наборі (a,b,b) набуває значення I, а формула G – значення X.

Необхідно виділити, що для логіки предикатів зберігаються всі рівносильності та правила рівносильних перетворень логіки висловлювань.

3.5. Закони і тотожності логіки предикатів

Усі закони і тотожності, які справедливі у логіці висловлювань, залишаються справедливими й у логіці предикатів. Але у логіці предикатів існують додаткові закони, які використовують для еквівалентного перетворення формул, що містять квантори та змінні.

1. *Заміна зв'язаної змінної.*

Уведення нового позначення зв'язаної змінної не змінює змісту формули логіки предикатів за умови, якщо ніяка вільна змінна у будь-якій частині формули не буде після перейменування зв'язаною змінною, наприклад:

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

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

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

2. *Комутативні властивості кванторів.*

У логіці предикатів можна змінювати місцями тільки однойменні квантори, наприклад:

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

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

$\forall x \exists y P(x,y) \neq \exists y \forall x P(x,y)$.

Приклад 3.5.1. Для предиката $\forall x \exists y$ ДОРІВНЮЄ$(x+1, y)$ показати, що зміна місцями кванторів приводить до невідповідності між висловлюваннями до і після зміни місцями кванторів.

Розв'язання. Заданий предикат є висловлюванням і означає, що для будь-якого числа x існує число y, яке більше його на одиницю. Наведене висловлювання є істинним. Але якщо поміняти порядок розташування кванторів на протилежний, то отримаємо таке висловлювання: $\exists y \forall x$ ДОРІВНЮЄ$(x+1, y)$. Одержане висловлювання означає, що існує таке число y (одне), яке на одиницю більше будь-якого числа x. Це висловлювання не відповідає попередньому і є хибним, що підтверджує закон комутативної властивості кванторів.

3. *Дистрибутивні властивості кванторів.*

Нехай формула $P(x)$ містить пов'язану змінну x, а формула Q не містить пов'язаної змінної x й обидві вони задовольняють п.3 означення 3.3.1, тоді

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

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

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

$\forall x (P(x) \vee Q) = \forall x P(x) \vee Q$.

Доведемо першу з цих рівностей (інші доводять аналогічно). Нехай $x^{i_1}, x^{i_2} \ldots, x^{i_\kappa}$ – усі вільні змінні формули $\exists x (P(x) \wedge Q)$. Тоді вони будуть усіма вільними змінними формули $\exists x P(x) \wedge Q$.

Розглянемо довільну інтерпретацію на множині M. Нехай $(a_1, a_2, \ldots, a_\kappa)$ – довільний набір значень вільних змінних $x^{i_1}, x^{i_2} \ldots, x^{i_\kappa}$, де $a^i \in M$. Оскільки формула Q не містить пов'язаної змінної x, то можна визначити значення цієї формули на наборі $(a_1, a_2, \ldots, a_\kappa)$ у частині, що стосується вільних змінних формули Q. Якщо формула Q на наборі $(a_1, a_2, \ldots, a_\kappa)$ набула значення X, то $(\exists x P(x) \wedge Q)$ на цьому наборі теж X і для будь-якого елемента $a^i \in M$ на наборі $(a_1, a_2, \ldots, a_\kappa)$ своїх вільних змінних $x^{i_1}, x^{i_2} \ldots, x^{i_\kappa}$ формула $P(x) \wedge Q$ набуває значення X. Звідси випливає, що $\exists x P(x) \wedge Q$ на цьому наборі теж дорівнює X. Якщо формула Q на наборі $(a_1, a_2, \ldots, a_\kappa)$ набула значення I, то для будь-якого елемента $a^i \in M$ на наборі $(a_1, a_2, \ldots, a_\kappa)$ формула $P(x) \wedge Q$ й $P(x)$ набувають однакових значень істинності. Звідси випливає, що $\exists x (P(x) \wedge Q)$ на наборі $(a_1, a_2, \ldots, a_\kappa)$ тотожна $\exists x P(x) \wedge Q$.

Якщо не вимагати, щоб формула Q не містила пов'язаної змінної x, то справджуватимуться тільки дві рівносильності:

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

$\exists x (P(x) \vee Q(x)) = \exists x P(x) \vee \exists x Q(x)$.

Сформульований дистрибутивний закон справедливий тільки для квантора загальності $\forall$ при кон'юнкції і квантора існування $\exists$ при диз'юнкції, оскільки інші комбінації приводять до нерівностей

$\forall x (P(x) \vee Q(x)) \neq \forall x P(x) \vee \forall x Q(x)$;

$\exists x (P(x) \wedge Q(x)) \neq \exists x P(x) \wedge \exists x Q x)$.

Для подолання цього обмеження дистрибутивного закону, слід використовувати заміну зв'язаної змінної

$\forall x P(x) \vee Q(x) = \forall x P(x) \vee \forall y Q(y) = \forall x \forall y (P(x) \vee Q(y))$;

$\exists x\, P(x) \wedge \exists x\, Q(x) = \exists x\, P(x) \wedge \exists y\, Q(y) = \exists x \exists y\, (P(x) \wedge Q(y))$.

4. *Закон де Моргана для кванторів.*

Нехай $P(x)$ – формула, що містить пов'язані й вільні змінні. Тоді справджуються рівносильності

$\neg \forall x P(x) = \exists x \neg P(x)$; (3.5.1)

$\neg \exists x\, P(x) = \forall x \neg P(x)$. (3.5.2)

Доведемо спочатку рівносильність 3.5.1. Нехай $x^{i_1}, x^{i_2}, \ldots, x^{i_n}$ – множина всіх вільних змінних формули P, відмінних від пов'язаних змінних x. Покажемо, що на будь-якому наборі значень вільних змінних $(a^1, a^2, \ldots, a^n)$, $a^i \in M$, формули $\neg \forall x P(x)$ і $\exists x \neg P(x)$ набувають однакових значень істинності. При цьому можливі два випадки.

1. 1. Для будь-якого елемента $a \in M$ $P(x) = I$ на наборі $(a, a^1, a^2, \ldots, a^n)$.
2. 2. Для деякого елемента $a^0 \in M$ $P(x) = X$ на наборі $(a^0, a^1, a^2, \ldots, a^n)$.
3. У першому випадку для будь-якого елемента $a \in M$ маємо $\neg P(x) = X$ на наборі $(a^1, a^2, \ldots, a^n)$. Звідси за означенням $\exists x \neg P(x) = X$ на наборі $(a^1, a^2, \ldots, a^n)$, з іншого боку, $\forall x P(x) = I$, а $\neg \forall x P(x) = X$ на наборі $(a^1, , a^2, \ldots, a^n)$.

У другому випадку для елемента $a^0 \in M$ маємо $\neg P(x) = I$ на наборі $(a^0, a^1, a^2, \ldots, a^n)$. Звідси $\exists x \neg P(x) = I$ на наборі $(a^1, a^2, \ldots, a^n)$. З іншого боку, $\forall x P(x) = X$, а $\neg \forall x P(x) = I$ на наборі $(a^1, a^2, \ldots, a^n)$, що і потрібно було довести для рівносильності формули 3.5.1. Доведення рівносильності формули 3.5.2 аналогічне описаному для формули 3.5.1.

3.6. Властивості числення предикатів першого порядку

У цьому параграфі розглядаються основні властивості числення предикатів першого порядку.

Теорема 3.6.1 (теорема про повноту). Кожна виведна в численні предикатів першого порядку формула є загальнозначущою.

Доведення. Достатньо впевнитися в тому, що аксіоми *АП1– АП5* – загальнозначущі формули, і показати, що застосування правил виведення *АП1– АП3* до загальнозначущих формул веде до одержання тільки загальнозначущих формул.

Загальнозначущість аксіом *АП1– АП3* (вони ж *А1– А3*) було розглянуто в 2.1, 2.2. Покажемо загальнозначущість формули $\forall x\, A(x) \to A(y)$ (аксіома *АП4*). Припустимо, що ця формула загальнозначуща, тоді одержимо, що у задовільній предметній області є такий набір значень предметних змінних, при якому $\forall x\, A(x) = I$, а $A(y) = X$. Але оскільки є значення предметних змінних, при яких формула A набуває хибного значення X, тоді згідно з квантором загальності формула $\forall x\, A(x)$ також повинна набувати значення X. Виникла суперечність і доводить загальнозначущість формули $\forall x\, A(x) \to A(y)$, звідки випливає і загальнозначущість аксіоми *АП4*. При доведенні аксіоми *АП5* припустимо, що формула $\forall xi\, (A \to B) \to (A \to \forall xi\, B)$ незагальнозначуща. Тоді при деякій інтерпретації формула $A \to \forall xi\, B$ набуде значення X, а $\forall xi\, (A \to B)$ – значення I. Якщо $A \to \forall xi\, B = X$, то $A = I$, а $\forall xi\, B = X$. Оскільки $\forall xi\, (A \to B) = I$, то для будь-якого xi при $B = X$ повинне виконуватися $A = I$. Із протиріччя для істинного значення формули A випливає справедливість загальнозначущості аксіоми *АП5*, що і необхідно було довести.

Теорема 3.6.2. Числення предикатів першого порядку ***ПЛ*** є **несуперечною теорією**.

Доведення. Припустимо, що для деякої формули A в теорії ***ПЛ*** є дві теореми A і $\neg A$. Із факту загальнозначущості будь-якої із цих теорем отримаємо, що $A = I$ і $\neg A = I$, але одночасно цього не може бути, що й потрібно було довести.

Теорема 3.6.3. Якщо формула B не залежить від формули A при доведенні Γ, $A \vdash B$, то $\Gamma \vdash B$.

Доведення. Доведення виконаємо індукцією за довжиною виведення

$B_1, B_2, \ldots, B_n = B$. При $n = 1$ маємо $B_1 = B$ і B або належить $\Gamma \cup \{A\}$, або є аксіомою. Але B не може збігатися з A, внаслідок незалежності B від A. Отже, $\Gamma \vdash B$.

Нехай теорема справедлива для всіх формул B, довжина виведення яких менша n. Якщо $B \in \Gamma$ або B – аксіома, то $\Gamma \vdash B$. Якщо ж B – безпосередній наслідок деяких формул із $B_1, B_2, \ldots, B_{n-1}$ (однієї або двох) і оскільки B не залежить від A, то від A не залежить жодна із цих формул. Але тоді за припущенням індукції, із Γ виводяться формули (одна або дві), а разом з ними – і формула B, що й потрібно було довести.

Теорема 3.6.4 (теорема дедукції). Нехай $\Gamma, A \vdash B$ і при цьому нехай існує таке виведення B із Γ, A, в якому жодне застосування правила узагальнення до формул, які залежать від A в цьому виведенні, не зв'язується квантором загальності формули A. Тоді $\Gamma \vdash A \to B$.

Доведення. Доведення проводиться індукцією за довжиною виведення

$B_1, B_2, \ldots, B_n = B$. При $n = 1$ існують дві можливості:

а) $B_i \in \Gamma$ або є аксіомою;

б) $B_1 = A$.

У першому випадку а) $A \to B$, оскільки це випливає з аксіоми *АП1*, $B_1 \to (A \to B_i)$ за правилом МР.

У випадку б), коли $A = B_i$, то $\Gamma \vdash A_i \to B_j$, оскільки $A \to A$ – тавтологія.

Нехай тепер теорема справедлива для всіх $i < n$. Припустимо, що існують індекси k і j, менші i, і що $B_k = B_j \to B_i$. Тоді за припущенням індукції $\Gamma \vdash A \to B_j$ і $\Gamma \vdash A \to (B_j \to B_i)$. Але за аксіомою *АП2* маємо $A \to (B_j \to B_i) \to \to ((A \to B_j) \to (A \to B_i))$ і за правилом МР отримаємо, що $\Gamma \vdash A \to B_i$.

Припустимо, нарешті, що існує такий індекс $j < i$, що $B_i = \forall xk\, B$. Тоді за припущенням індукції $\Gamma \vdash A \to B_i$ і або B_j не залежить від A, або xk не є вільною змінною формули A. Якщо B_j не залежить від A, то за теоремою 7.2.1

$\Gamma \vdash B_j$ і застосовуючи правило узагальнення отримаємо $\Gamma \vdash \forall xk\, B_j$. За схемою аксіоми *АП1* знаходимо, що $B_j \to (A \to B_j)$ і $A \to B_i$ – за правилом МР. За припущенням індукції $\Gamma \vdash A \to B_j$, за правилом узагальнення $\Gamma \vdash \forall xk$, $A \to B$ і, нарешті, за правилом МР отримаємо $\Gamma \vdash A \to \forall xk\, B_j$, тобто $\Gamma \vdash A \to B_i$, що й потрібно було довести.

Наслідок 3.6.1. Якщо $\Gamma, A \vdash B$ та існує виведення B без застосування правила узагальнення до вільних змінних формули A, то $\Gamma \vdash A \to B$.

Наслідок 3.6.2. Якщо формула A замкнена і $\Gamma, A \vdash B$, то $\Gamma \vdash A \to B$.

Приклад 3.6.1. Довести, що $\vdash \forall x\, \forall y\, A \to \forall x\, \forall y\, A$.

Розв'язання. Застосуємо предикатні аксіоми, правило МР та узагальнення:

а) $\forall x\, \forall y\, A$ – гіпотеза ;

б) $\forall x\, \forall y\, A \to \forall y\, A$ – аксіома *АП4* ;

в) $\forall y\, A$ – МР із а), б) ;

г) $\forall y\, A \to A$ – аксіома *АП4* ;

д) A – МР із в), г) ;

е) $\forall x\, A$ – правило узагальнення із д) ;

є) $\forall y\, \forall x\, A$ – правило узагальнення із е) .

Теорема 3.6.5 . Якщо формули $A(x)$ і $A(y)$ подібні, то $\vdash \forall x\, A(x) \Leftrightarrow \forall y\, A(y)$.

Доведення. Використовуючи аксіому *АП4*, маємо $\vdash \forall x\, A(x) \to A(y)$. Виходячи з умови теореми 3.6.5 і скориставшись правилом узагальнення, отримаємо

$\vdash \forall y\, (\forall x\, A(x) \to A(y))$. За схемою аксіоми *АП5* маємо $\vdash \forall x\, A(x) \to \forall y\, A(y)$. Аналогічно доводять і формулу $\vdash \forall y\, A(y) \to \forall x\, A(x)$. Для цього необхідно скористатися тавтологією $A \to (\neg\neg B \to \neg (A \to \neg B)) \Leftrightarrow A \to (B \to A \wedge B)$, у результаті чого отримаємо $\forall x\, A(x) \Leftrightarrow \forall y\, A(y)$.

Теорема 3.6.6 (теорема еквівалентності). Якщо формула B є підформулою A і формула A^1 отримана з формули A заміною яких-небудь (можливо, жодного) входжень B формулою C і якщо будь-яка змінна формули B або формули C, що є одночасно пов'язаною змінною формули A, зустрічаються в списку $y_1, y_2, \ldots, y_k$, то $\vdash \forall y1\, \forall y2 \ldots \forall yk\, (B \Leftrightarrow C) \to (A \Leftrightarrow A')$.

Доведення. Доведення виконаємо індукцією за числом зв'язок і кванторів у A. Зауважимо, що якщо жодне входження B насправді не змінюється, то A збігається з A' і формула, яку потрібно вивести, є окремим випадком тавтології $B \to (A \Leftrightarrow A)$. Якщо B збігається з A і це єдине входження замінюється на C, то формула, яку необхідно вивести, виводиться із аксіоми *АП4*. Отже, необхідно вважати, що B є власна підформула формули A і принаймні одне входження B замінюється.

Тепер припустимо, що теорема справедлива для будь-якої формули з меншим числом кванторів і логічних зв'язків, ніж у A.

Випадок 1. A – елементарна формула. Тоді B не може бути власною підформулою формули A.

Випадок 2. A має вигляд $\neg D$. Тоді нехай A' є $\neg D'$. За припущенням індукції $\vdash \forall y1\, \forall y2 \ldots \forall$

yk (B $\Leftrightarrow$ *C) → (D* $\Leftrightarrow$ *D')*. Звідси за допомогою тавтології $A \Leftrightarrow B \to (\neg A \to \neg B)$ отримаємо

$\vdash \forall y1\ \forall y2 \ldots \forall yk\ (B \Leftrightarrow C) \to (A \Leftrightarrow A')$.

Випадок 3. *A* має вигляд $D \to E$. Нехай $A' = D' \to E'$. За припущенням індукції

$\vdash \forall y1\ \forall y2 \ldots \forall yk\ (B \Leftrightarrow C) \to (D \Leftrightarrow D')$ і $\vdash \forall y1\ \forall y2 \ldots \forall yk\ (B \Leftrightarrow C) \to (E \Leftrightarrow E')$. Застосовуючи тавтологію $((A \Leftrightarrow B) \wedge (C \Leftrightarrow D)) \to ((A \to C) \Leftrightarrow (B \to D))$,

отримаємо $\vdash \forall y_1\ \forall y_2 \ldots \forall y_k\ (B \Leftrightarrow C) \to (A \Leftrightarrow A')$.

Випадок 4. *A* має вигляд $\forall x\ D$. Тоді $A' \in \forall x\ D'$. За припущенням індукції $\vdash \forall y1\ \forall y2 \ldots \forall yk\ (B \Leftrightarrow C) \to (D \Leftrightarrow D')$. Змінна *x* не є вільною змінною формули $\vdash \forall y1\ \forall y2 \ldots \forall yk\ (B \Leftrightarrow C)$. Якщо ж припустимо, що це так, то *x* входила б вільно в *B* або *C*, і оскільки *x* пов'язана в *A*, то *x* входила б до переліку $y_1, y_2, \ldots, y_k$ і була б пов'язаною змінною в $\forall y1\ \forall y2 \ldots \forall yk\ (B \Leftrightarrow C)$, а це суперечить умові. Застосовуючи тепер аксіому *АП5*, отримаємо $\forall y1\ \forall y2 \ldots \forall yk\ (B \Leftrightarrow C) \to (D \Leftrightarrow D')$. Використовуючи $\forall x\ (A \Leftrightarrow D) \to (\forall xA \Leftrightarrow \forall x\ D)$, маємо $\forall x\ (D \Leftrightarrow D') \to (\forall xD \Leftrightarrow \forall xD')$. Користуючись транзитивністю імплікації, остаточно отримаємо

$\forall y1\ \forall y2 \ldots \forall yk\ (B \Leftrightarrow C) \to (\forall xD \Leftrightarrow \forall xD')$ або $\forall y1 \forall y2 \ldots \forall yk\ (B \Leftrightarrow C) \to (A \Leftrightarrow A')$, що й потрібно було довести.

Теорема 3.6.7 (теорема про заміну). Нехай формули *A*, *A'*, *B* і *C* задовольняють умови теореми еквівалентності. Тоді якщо $\vdash B \Leftrightarrow C$, то $\vdash A \Leftrightarrow A'$, а якщо $\vdash B \Leftrightarrow C$ і $\vdash A$, то $\vdash A'$. Дана теорема є простим наслідком теореми еквівалентності.

Теорема 3.6.8 (теорема перейменування пов'язаних змінних). Якщо $\forall x\ B(x)$ є підформулою формули *A*, формула *B (y)* подібна формулі *B(x)* і A' отримана з *A* заміною хоча б одного входження $\forall x\ B(x)$ у *A* на $\forall y\ B(y)$, то $\vdash A \Leftrightarrow A'$.

Доведення. Доведення випливає з теорем 3.6.5 і 3.6.7.

Запитання для самоперевірки:

1. Які додаткові нові логічні поняття введені в логіку предикатів і з якою метою?
2. Дайте визначення поняття «предикат».
3. Що називають *n*-місним предикатом?
4. Що називають порядком предиката?
5. Що називають нульмісним предикатом?
6. Наведіть приклади функціональних символів.
7. Який функціональний символ називають *n*-місним?
8. Дайте визначення поняття «терма».
9. Що розуміють під предметною областю?
10. Дайте визначення понять «предметна константа» і «предметна змінна».
11. Дайте визначення квантора загальності.
12. Що розуміють під квантором існування?
13. Як позначають квантори загальності та існування?
14. Які змінні називають зв'язаними, а які – вільними?
15. Як залежить значення предиката від зв'язаних або вільних змінних?
16. Поясніть призначення зв'язаних змінних.
17. До яких наслідків призводить застосування квантора за однією із змінних тримісного предиката.
18. Коли предикат $\forall y\ P(x,y)$ набуває значення "*Істина*"?
19. Коли предикат $\exists x\ P(x,y)$ набуває значення "*Істина*"?
20. Що називають формулою у логіці предикатів?
21. Яку формулу називають атомарною?
22. Що розуміють під областю дії квантора?
23. Що таке інтерпретація формули логіки предикатів?
24. Запишіть правила, згідно з якими формула логіки предикатів може набути істинного або хибного значення.
25. Порівняйте поняття інтерпретації у логіці висловлювань і логіці предикатів.
26. Що таке рівносильність формул логіки предикатів?
27. Як впливає область інтерпретації на рівносильність формул логіки предикатів?
28. Як впливають на рівносильність вільні й зв'язані змінні у формулах логіки предикатів?
29. До якого наслідку призводить перенесення квантора на початок формули?
30. Які додаткові закони і тотожності існують у логіці предикатів порівняно з логікою висловлювань?
31. Поясніть суть заміни зв'язаної змінної.
32. Сформулюйте комутативні властивості кванторів.
33. Сформулюйте дистрибутивні властивості кванторів.
34. Яким чином можна обійти обмеження дистрибутивних властивостей кванторів?
35. Запишіть формули закону де Моргана для кванторів.
36. Яке числення називають численням предикатів першого порядку?
37. Які властивості числення предикатів першого порядку ви знаєте?
38. Довести, що будь-яке числення предикатів першого порядку є несуперечною теорією.
39. Сформулюйте умови теореми еквівалентності формул у численні предикатів першого порядку.

Задачі для самостійного розв'язування

1. Напишіть, використовуючи мову логіки предикатів такі речення:
   а) "*число x більше 2*";
   б) "*x сестра y*";
   в) "*діагоналі ромба перпендикулярні*";
   г) "*кожне явище має свою причину*".
2. Перекладіть природною мовою такі висловлювання логіки предикатів:
   а) ДОРІВНЮВАТИ *(x , 10)*;
   б) ЛЮБИТИ (мама (Каті), гімнастика) ;
   в) БІЛЬШЕ (плюс *(x , 1) , 1)*.
3. Змінні функції "$x > y$" набувають значень на множині *{ 2,3,4 }*; P_1 , P_2 - предикати, що задаються цією функцією відповідно при алфавітному і зворотному йому порядках. Необхідно встановити:
   а) область визначення предикатів P_1 і P_2;
   б) значення істинності $P_1(3,4)$ і $P_2(3,4)$.
4. Визначте еквівалентність таких предикатів:
   а) $x^2 = x$ і $x = 1$;
   б) $x^2 = 1$ і $x = 1$;
   в) $x^3 = 1$ і $x = 1$.
5. Запишіть такі висловлювання, використовуючи логіку предикатів:
   а) будь-яке парне додатне число ;
   б) існує таке число *x*, що $x^2 + 1 = 10$;
   в) для будь-якого *x* завжди $x + 0 = x$.
6. Вкажіть вільні та пов'язані входження кожної зі змінних у таких формулах:
   а) $\exists x\ P(x,y) \vee \exists x\ Q(x,y)$;
   б) $\forall y\ Q(x,y) \wedge \forall x\ P(x,y)$;
   в) $\forall x\ [(P(x,y) \to Q(y)) \vee \exists y\ P(x,y)]$.
7. Наведені висловлювання для предиката *P(x,y)*, де "*x сестра y*", сформулювати природною мовою, визначивши їх значення істинності:
   а) $\exists y\ \forall x\ P(x,y)$;
   б) $\exists x\ \forall y\ P(x,y)$;
   в) $\exists x\ \exists y\ P(x,y)$;
   г) $\forall y\ \exists x\ P(x,y)$;
   д) $\forall x\ \exists y\ P(x,y)$;
   е) $\forall x\ \forall y\ P(x,y)$.
8. Предикат $P(x,y)$ задано на предметній області $D=\{a, в\}$ такою матрицею:

x	*а*	*а*	*в*	*в*
y	*а*	*в*	*а*	*в*
P(x,y)	*Х*	*І*	*І*	*І*

Яка з наведених формул визначає цей предикат?

а) $\forall x\ \forall y\ P(x,y)$;
б) $\exists y\ \forall x\ P(x,y)$;
в) $\forall x\ P(x,a)$;
г) $\forall y\ \exists x\ P(x,y)$;
д) $\forall y\ P(a;y)$;
є) $\exists x\ \exists y\ P(x,y)$;

9. Визначте вільні та пов'язані входження змінних у такі формули, що містять предикати *P* і φ:

1.
   1. $\forall x_2\ P(x_1, x_2)$;
   2. $\exists x_1 (\forall x_3\ P(x_1, x_2) \to P(x_1, x_2))$;
   3. $\forall x3 (\forall x_1 P(x_1, x_2) \longleftarrow P(x_1, x_2))$;
   4. $\forall x_1 \exists x_2 P(x_1, x_2) \to Q(x_1)$;
   5. $\forall x_2 P(x_1, x_2) \to \exists x_1 P(x_1, x_2)$.

10. Знайдіть значення істинності таких предикатних виразів:

1. $P(a, f(a)) \wedge P(b, f(b))$;
2. $\forall x \forall y\ P(x,y) \to P(f(x), f(y))$

на предикатній області $D = \{3,4\}$ при значенні функціональних символів:
$f(a) = 4$, $f(b) = 3$; предикатів $P(3,3,) = I$, $P(3,4,) = I$, $P(4,3,) = I$, $P(4,4,) = X$.

11. Оцінити формулу $\exists x\ P(x) \to Q(f(x), a)$ на інтерпретації при: $D = \{1,2\}$;

a = 1; f (1) = 2; f (2) = 1;P (1) = X; P (2) = I; Q (1,1) = I; Q (1,2) = I;
Q (2,1) = X; Q (2,2) = I.

12. Визначити рівносильність формул $\exists\ x_1 P\ (x_1)$, $\forall\ x_1 P\ (x_1)$ на двоелементній множині *M = { a,b }*.

13. Визначити рівносильність формул *F = P (x1,x2)* $\vee$ *P (x1,x3)* і *G = P (x1,x2)* $\vee$ *P (x2,x3)*, які задані на множині *M = { a,b }* предикатами Q_1 *(x,y)* та Q_2*(x,y)*, наведеними в таблицях

x	y	$Q_1(x,y)$	x	y	$Q_1(x,y)$
a	a	*X*	a	a	*x*
a	b	*I*	a	b	*I*
b	a	*I*	b	a	*X*
b	b	*X*	b	b	*I*

14. Для заданих предикатів $\exists\ x\ (P\ (x) \wedge [\urcorner?]\ Q\ (x))$ і $[\urcorner?]\ \forall\ z\ (P\ (z) \to Q\ (y))$ встановити їх еквівалентність.

15. Винести за дужки квантори:
а) $\exists\ x\ P\ (x,y) \vee (\forall\ x\ Q\ (x) \vee \forall\ y\ Q\ (y))$;
б) $\exists\ x\ \forall\ y\ P\ (x,y) \wedge \exists\ x\ \exists\ y\ (Q\ (x, z) \wedge \exists\ y\ P\ (x,y))$;
в) $\exists\ x\ P\ (x,y) \wedge (\exists\ y\ P\ (y) \vee Q)$.

16. Довести загальнозначущість такої формули: $x\ P\ (x) \vee [/TEX;tkfntkmyj]</i><span\ style="text-indent:\ 20px;">[TEX]\exists\ y \neg P\ (y)$.

17. Довести суперечливість формули $\forall\ x\ P\ (x) \wedge \exists\ y \neg P\ (y)$.

18. Показати , що комутативні властивості для виразу $\forall\ x\ \exists\ y\ P\ (x,y) \neq \exists\ y\ \forall\ x\ P\ (x,y)$ не виконуються.

19. Довести, що:
а) $\vdash \forall\ x\ (A \to B) \to (\forall\ x\ A \to \forall\ x\ B)$;
б) $A, \forall\ x\ A \to B \vdash \forall\ x\ B$;
в) $\vdash \forall\ x_1\ \forall\ x_2 \ldots \forall\ x_m\ A \to A$;
г) $\forall\ x\ (A \to B) \Leftrightarrow (A \to \forall\ x\ B)$;
д) $\forall\ x\ (B \to A) \Leftrightarrow (\exists\ x\ B \to A)$;
е) $\forall\ x\ (A \Leftrightarrow D) \to (\forall\ x\ A \Leftrightarrow \forall\ x\ B)$.

Коментарі. Основні відомості, з логіки предикатів, кванторів і формул логіки предикатів, викладені в цьому розділі, узяті з [9,20], рівносильність формул логіки предикатів і логічні висновки – з [1,7,21] , а закони, тотожності логіки предикатів і випереджені нормальні форми випливають із[20,28] .

Тема 4

- Аксіоматичні системи логік
  - 4.1. Система аксіом і правил виведення
  - 4.2. Випереджені нормальні форми
  - 4.3. Побудова доведень в аксіоматичній системі
  - 4.4. Метод ідентифікації
  - 4.5. Метод резолюцій

Аксіоматичні системи логік

Аналогічно числення висловлювань в аксіоматичній системі логік існує формальна система – числення предикатів, яка займається конструюванням формул і доведенням їх загальнозначущості. Числення предикатів, тобто формальна теорія предикатів будується за вищенаведеною класичною схемою побудови формальних теорій.

Аксіоматична система логік має ідентичну численню висловлювань структуру: мову, систему аксіом і правила виведення. Мова аксіоматичної системи логік складається з правильно побудованих формул логіки предикатів першого порядку, а опис аксіом і правил виведення наведений у наступному параграфі.

4.1. Система аксіом і правил виведення

Числення предикатів першого порядку визначають аксіоматичним шляхом. Аксіоми поділяють на два класи: логічні аксіоми і власні аксіоми, або нелогічні аксіоми. До логічних аксіом відносять аксіоми, які визначаються для будь-яких *A, B, C* схемами:

АП1) A→ (B → A);
АП2) (A → (B → C)) → ((A → B) → (A → C));
АП3) (¬ B → A) → ((¬ B → A) → B);
АП4) $\forall$ *x1A(x1) → A(t), де t – терм, вільний для змінної x1 у формулі A(x1);*
АП5) $\forall$ *x1 (A → B) → (A →* $\forall$ *x1B), якщо формула A не містить вільних входжень змінної x1.*

Власні аксіоми, будучи визначеними, фіксують деяку конкретну теорію першого порядку, тому для спільного описання всіх теорій першого порядку вони не можуть бути сформульовані. Якщо власних аксіом немає, то таку теорію називають **численням предикатів першого порядку** і позначають ***ПЛ***.

У численні предикатів першого порядку визначено два основні правила виведення:

П1) **modus ponens** (МР): із *A і A→B* випливає *B*;

П2) правило узагальнення: із *A* випливає $\forall$ *xi A*.

Поняття виведення із множини формул у теорії ***ПЛ*** визначають аналогічно численню висловлювань, але при цьому необхідно внести додаткове уточнення. Воно полягає в тому, що у виведенні формули *B* із множини формул *Г* змінна *xi* є фіксована, якщо в цьому виведенні ні разу не застосовувалося правило *П2* узагальнення по змінній x_i. У протилежному разі кажуть, що змінна x_i у даному виведенні варіюється.

Для позначення факту наявності виведення формули *B* із множини формул *Г*, в якому всі змінні, *xj,…xjz* фіксовані, використовують запис $Г \vdash x_j,\ldots x_{jz}\ B$. Інакше кажучи, у цьому записі указують, що правило узагальнення використовують тільки по змінних $x_j,\ldots x_{jz}$.

Приклад 4.1.1. Вивести формулу *A(a0, x2)* із формули *A(x1, x2)*.

Розв'язання. Для виведення формули застосуємо правило узагальнення й із *A(x1, x2)* одержимо $\forall\ x_1\ A(x_1, x_2)$. Згідно з аксіомою *АП4* запишемо аксіому

$$\forall\ x_1\ A(x_1, x_2) \to A(x_0, x_2). \qquad (4.1.1)$$

Одержана формула $\forall\ x_1\ A(x_1, x_2)$ разом з 4.1.1 по *МР* дозволяє вивести *A (a0,x)*. У побудованому виведенні змінна *x1* варіюється. Тоді виведення формули можна записати у такому вигляді $A(x_1, x_2) \vdash x_1\ A_1\ (a_0, x_2)$.

Як зазначалося вище, в логіці предикатів першого порядку визначені два основні правила виведення: *П1), П2),* але з розділів 1 і 2 можна зробити висновок, що для можливості скорочення довжини виведення необхідно розширити множину правил виведення. Розглянемо деякі з них, які можуть бути застосовані при виведенні формул у логіці предикатів першого порядку.

Їх можна розділити на дві групи. До першої відносять такі структурні правила виведення.

1. Закон тотожності
$A \vdash A$

2. Правило введення, яке формулюється так: якщо $Г \vdash A$, то $Г, B \vdash A$. Для наглядності його, а в подальшому й усі інші, будемо записувати у вигляді
$\frac{Г \vdash A}{Г, B \vdash A}$

3. Правило перестановки
$\frac{Г, B, C, Δ \vdash A}{Г, C, B, Δ \vdash A}$

4. Правило скорочення
$\frac{Г, B, B, Δ \vdash A}{Г, B, Δ \vdash A}$

5. Правило перетину
$\frac{Г \vdash Δ, A \vdash B}{Г, Δ \vdash C}$

До другої групи відносять логічні правила виведення, які, у свою чергу, можна розбити на групи, для яких є логічні зв'язки і квантори зі своєю групою правил. Крім того, усередині кожної групи правила ділять на два види: **правила введення**, які показують доведення формули з даним логічним символом, і **правила вилучення**, які показують використання формули з даним логічним символом для доведення інших формул.

1. Правило диз'юнкції:

введення

$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}$, $\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$.

вилучення

$\frac{\Gamma, A \vdash C;\ \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C}$.

2. Правило кон'юкції:

введення

$\frac{\Gamma \vdash A;\ \Gamma \vdash B}{\Gamma \vdash A \wedge B}$.

вилучення

$\frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}$.

3. Правило імплікації:

введення

$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$.

вилучення

$\frac{\Gamma \vdash A;\ \Gamma \vdash A \to B}{\Gamma \vdash B}$.

4. Правило еквівалентності:

введення

$\frac{\Gamma, A \vdash B;\ \Gamma, B \vdash A}{\Gamma \vdash A \sim B}$.

вилучення

$\frac{\Gamma \vdash;\ \Gamma \vdash A \sim B}{\Gamma \vdash B}$.

$\frac{\Gamma \vdash B;\ \Gamma \vdash A \sim B}{\Gamma \vdash A}$.

5. Правило заперечення:

введення

$\frac{\Gamma, A \vdash B;\ \Gamma, A \vdash \neg B}{\Gamma \vdash \neg A}$.

вилучення

$\frac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}$.

6. Правило узагальнення:

введення

$\frac{\Gamma \vdash A(y)}{\Gamma \vdash \forall x\, A(x)}$.

вилучення

$\frac{\Gamma \vdash \forall x\, A(x)}{\Gamma \vdash A(y)}$.

У формулі введення квантора загальності *A (y)* означає для довільного $y \in F$, а саме правило стверджує істинність $\forall x\, A(x)$, якщо доведена істинність *A (y)* для будь-якого *y*, тобто для всіх елементів *y* з розглянутої предметності області *F*.

У формулі **видалення квантора загальності** *A (y)* означає для довільного $y \in F$, а саме правило використовується для доведення істинності *A (y)*, де *y* – довільно обраний елемент предметної області *F*, у якій справедливе $\forall x\, A(x)$.

7. Правило існування:

введення

$\frac{\Gamma \vdash A(y)}{\exists x\, A(x)}$.

вилучення

$\frac{\Gamma, \exists x\, A(x) \vdash C}{\Gamma, A(y) \vdash C}$.

У формулі **вилучення квантора існування** *A (y)* означає для деякого $y \in F$, а саме правило дозволяє вирішити, що $\exists x\, A(x)$ є істинним тоді і тільки тоді, коли відомий деякий елемент *y*, для якого істинне *A (y)*.

У формулі **вилучення квантора існування** *A (y)* означає для деякого $y \in F$, а саме правило полягає в позначенні імені елемента *y*, для якого *A (y)* є істинне.

Усі розглянуті вище правила відповідають звичайним прийомам математичного міркування. Наприклад, у правилі існування вилучення квантора $\exists$ відповідає правилу одиничного вибору. Припустимо, що є $\exists x\, A(x)$ і необхідно вивести *C*. Оскільки є *x* таке, що *A (x)*, то можна розглянути (вибрати) одне із таких *x*. Позначимо його через *y*. А це означає, що для цього *y* дійсне *A (y)*. Таким чином, достатньо вивести формулу *C* із *A (y)*. Аналогічно можна розглянути й інші правила.

8. Правило перейменування вільних змінних

У логіці предикатів у виведенні формули *A (x)*, що містить вільні входження *x*, жодне з яких не знаходиться в області дії квантора за *y*, виходить вивідність *A (x)*.

Приклад 4.1.1. Довести справедливість правила перейменування вільних змінних.

Розв'язання.

а) *A (x)* – гіпотеза;

б) $A(x) \to (B \to A(x))$ – аксіома *АП1:* тут як *B* можна обрати будь-яку довідну формулу, яка не містить вільних входжень змінних *x*;

в) $B \to A(x)$ – МР, із а), б) ;

г) $B \to \forall x\, A(x)$ – правило узагальнення із в) ;

д) $\forall x\, A(x)$ – наслідок з кроку г), оскільки *B* – істинна формула ;

у) *A (y)* – аксіома *АП4* із д) .

9. Правило перейменування пов'язаних змінних

У логіці предикатів у виведенні $\forall x\, A(x)$ виходить вивідність $\forall y\, A(y)$, а у виведенні $\exists x\, A(x)$ – вивідність $\exists y\, A(y)$ за умови, що *A (x)* не містить вільних входжень *y*, а містить вільні входження *x*, жодне з яких не входить до області дії квантора за *y*.

Приклад 4.1.2 Довести правило перейменування пов'язаних змінних для квантора загальності.

Розв'язання.

а) $\forall x\, A(x)$ – гіпотеза ;

б) $\forall x\, A(x) \to A(y)$ – аксіома *АП4* ;

в) $\forall x\, A(x) \to \forall y\, A(y)$ – правило узагальнення із б) ;

г) $\forall x\, A(x)$ – МР із а), в) .

Приклад 4.1.3 Довести правило перейменування пов'язаних змінних для квантора існування.

Розв'язання

а) *A (y)* – гіпотеза;

б) $A(y) \to \exists x\, A(x)$ – правило 7;

в) $\exists y\, A(y) \to \exists x\, A(x)$ – правило існування із а) і б);

г) $\exists x\, A(x)$ – МР із а) і в).

4.2. Випереджені нормальні форми

У логіці висловлювань було введено дві нормальні форми: кон'юнктивна і диз'юнктивна. В аксіоматичній системі логік вводиться третя нормальна форма, яку називають **випередженою нормальною формою.**

Означення 4.2.1. Формула *Ф* у логіці предикатів знаходиться у випередженій нормальній формі (ВНФ) тоді і тільки тоді, коли вона може бути зображена у вигляді $(Q_1 x_1) \ldots (Q_n x_n)\ (A)$, де $(Q_i x_i)$ є або $(\forall x)$, або $(\exists x)$ і всі x_i різні для різних $i = 1, \ldots, n$, а *A* – формула, яка не містить квантора. Приставку $(Q_1 x_1)$ називають префіксом, а *A* – матрицею формули *Ф*. Якщо формула *A* залежить від вільної змінної *x*, то це записується як *A (x)*, якщо ні, то – *A*.

Теорема 4.2.1. Для будь-яких формул *F, G, H* теорії числення першого порядку справедливі такі еквівалентності:

а) $\neg \forall x\, F(x) \Leftrightarrow \exists x\, (\neg F(x))$;

б) $\neg \exists x\, F(x) \Leftrightarrow \forall x\, (\neg F(x))$;

в) $\forall x\, F(x) \vee G \Leftrightarrow \forall x\, (F(x) \vee G)$;

$\forall x\, F(x) \wedge G \Leftrightarrow \forall x\, (F(x) \wedge G)$;

г) $\exists x\, F(x) \vee G \Leftrightarrow \exists x\, (F(x) \vee G)$;

$\exists x\, F(x) \wedge G \Leftrightarrow \exists x\, (F(x) \wedge G)$;

д) $\forall x\, F(x) \wedge \forall x\, H(x) \Leftrightarrow \forall x\, (F(x) \wedge H(x))$;

є) $\exists x\, F(x) \vee \exists x\, H(x) \Leftrightarrow \exists x\, (F(x) \vee H(x))$;

ж) $Q_1 x\, F(x) \wedge Q_2 x\, H(x) \Leftrightarrow Q_1 x\, Q_2 y\, (F(x) \wedge H(y))$;

з) $Q_1 x\, F(x) \vee Q_2 x\, H(x) \Leftrightarrow Q_1 x\, Q_2 y\, (F(x) \vee H(y))$.

Доведення . Виконаємо доведення еквівалентності а) $\neg \forall x\, F(x) \Leftrightarrow \exists x\, (\neg F(x))$. Нехай *r* – довільна інтерпретація на деякій області *R*. Якщо $\neg (\forall x\, F(x))$ істинна в інтерпретації *r*, то $\forall x\, F(x)$ хибна в *r*. Це означає, що існує такий елемент $k(x) \in R$, для якого $F(k(x))$ хибна, або те саме, що $\neg F(k(x))$ істинна в *r*. Отже, $\exists x\, (\neg F(x))$ істинна в *r*.

Якщо $\neg (\forall x\, F(x))$ хибна в *r*, то $\forall x\, F(x)$ істинна в *r*. Це означає, що *F(x)* виконується на всіх послідовностях із *R* або що $\neg F(x)$ не виконується на жодній такій послідовності з *R*. Отже, $\exists x\, (\neg F(x))$ хибна в *r*. Таким чином,

$\neg (\forall x\, F(x)) \Leftrightarrow \exists x\, (\neg F(x))$.

Доведення еквівалентності б) аналогічне доведенню еквівалентності а).

Доведемо еквівалентність в) $\forall x\, F(x) \vee G \Leftrightarrow \forall x\, (F(x) \vee G)$. Нехай маємо $\exists x\, F(x) \vee G \Leftrightarrow \neg (\forall x\, F(x)) \to G \Leftrightarrow \exists x\, (\neg F(x)) \to G$.

Отже,

1) $\exists x\, (\neg F(x)) \to G$ – гіпотеза;

2) $\neg F(x) \to \exists x\, (\neg F(x))$ – правило 7;

3) $\neg F(x) \to G$ – транзитивність із 1) і 2) ;

4) $\forall x\,(\neg F(x) \to G) \Leftrightarrow \forall x\,(F(x) \vee G)$ – загальнозначність із 3).

І навпаки:

1) $\forall x\,(F(x) \vee G) \Leftrightarrow \forall x\,(\neg F(x) \to G)$ – гіпотеза ;

2) $\neg F(x) \to G$ – правило 6 із 1;

3) $(\neg F(x) \to G) \to (\neg G \to F(x))$ – тавтологія ;

4) $\neg G \to F(x)$ – МР із 2) і 3) ;

5) $\forall x\,(\neg G \to F(x))$ – узагальнення із 4 ;

6) $\forall x\,(\neg G \to F(x)) \to (\neg G \to \forall x\,F(x))$ – схема аксіом *АП5* ;

7) $\neg G \to \forall x\,F(x)$ – МР із 5 і 6 ;

8) $(\neg G \to \forall x\,F(x)) \to (\neg \forall x\,F(x) \to G)$ – тавтологія ;

9) $\neg \forall x\,F(x) \to G \Leftrightarrow \forall x\,F(x) \vee G$ – МР із 7) і 8).

Доведемо спочатку першу еквівалентність г).

1) $\exists x\,F(x) \vee G \Leftrightarrow \forall x\,(\neg F(x)) \to G$ – гіпотеза ;

2) $(\neg F(x)) \to G$ – правило 6 ;

3) $(\neg F(x)) \to G \to \exists x\,(\neg F(x) \to G)$ – правило 7 ;

4) $\exists x\,(\neg F(x) \vee G) \Leftrightarrow \exists x\,(F(x) \vee G)$ – МР із 2) і 3).

І навпаки:

1) $\exists x\,(\neg F(x) \to G)$ – гіпотеза;

2) $\exists x\,(\neg F(x) \to G) \to (\neg F(x) \to G)$ – правило 7 ;

3) $\neg F(x) \to G$ – МР із 1) і 2) ;

4) $\forall x\,(\neg F(x)) \to \neg F(x)$ – правило 6 ;

5) $\forall x\,(\neg F(x)) \to G \Leftrightarrow \neg \forall x\,(\neg F(x)) \vee G \Leftrightarrow \exists x\,(F(x)) \vee G$ – транзитивність із 3) і 4).

Доведення інших еквівалентностей із в) і г) тепер легко випливає із законів де Моргана.

$\forall x\,F(x) \vee G \Leftrightarrow \neg\,(\neg \forall x\,F(x) \wedge \neg G) \Leftrightarrow ((\exists x \neg F(x)) \wedge \neg G \Leftrightarrow$

$\Leftrightarrow \neg(\exists x\,(\neg F(x) \wedge \neg G) \Leftrightarrow \neg\,(\exists x \neg\,(F(x) \vee G) \Leftrightarrow \forall x\,(F(x)) \vee G)$.

$\forall x\,F(x) \wedge G \Leftrightarrow \neg\,(\neg \forall x\,F(x) \vee \neg G) \Leftrightarrow (\exists x\,(\neg F(x)) \vee \neg G) \Leftrightarrow$

$\Leftrightarrow \neg(\exists x\,(\neg F(x) \vee \neg G) \Leftrightarrow \neg\,(\exists x \neg\,(F(x) \wedge G) \Leftrightarrow \forall x\,(F(x) \wedge G)$.

Аналогічно доводяться й інші еквівалентності із г).

Доведення решти еквівалентностей пропонуються як вправи.

Теорема доведена.

Із теореми 4.2.1 випливає такий метод побудови ВНФ. Для зведення формули Φ до випередженої нормальної форми використовують такі дії:

а) вилучення логічних зв'язок " $\to$" і " $\Leftrightarrow$" :

$A \Leftrightarrow B = (A \to B) \wedge (B \to A)$;

$A \to B = \neg A \vee B$;

б) вилучення і перенесення знака " $\neg$" всередину формул:

$\neg(\forall x\,F(x)) = \exists x\,(\neg F(x))$;

$\neg(\exists x\,F(x)) = \forall x\,(\neg F(x))$;

$\neg\neg F = F$;

в) перенесення кванторів:

$Q\,x\,P(x) \vee G = Q\,x\,(F(x) \vee G)$;

$Q\,x\,P(x) \wedge G = Q\,x\,(F(x) \wedge G)$;

$\forall x\,F(x) \wedge \forall x\,H(x) = \forall x\,(F(x) \wedge H(x))$;

$\exists x\,F(x) \vee \exists x\,H(x) = \exists x\,(F(x) \vee H(x))$;

$Q1\,x\,F(x) \wedge Q_2 x\,H(x) = Q_1 x\; Q_2\; y\,(F(x) \wedge H(y))$;

$Q1\,x\,F(x) \vee Q_2 x\,H(x) = Q_1 x\; Q_2\; y\,(F(x) \vee H(y))$.

Виходячи із методу побудови ВНФ, алгоритм отримання випередженої нормальної форми матиме такі кроки.

1. Вилучити логічні зв'язки еквіваленції " $\Leftrightarrow$" та імплікації " $\to$" за допомогою правил а).

2. Перенести знак операції заперечення " $\neg$" всередину формул, користуючись правилами: $\neg(F \vee G) = \neg F \wedge \neg G$; $\neg(F \wedge G) = \neg F \vee \neg G$,

і правилами б).

3. Виконати перейменування пов'язаних змінних, якщо є така необхідність.

4. Винести квантори на початок формули, використовуючи правила в).

Обґрунтуванням цього алгоритму є твердження, яке безпосередньо випливає з теореми 4.2.1.

Теорема 4.2.2. Алгоритм випередженої нормальної форми перетворює будь-яку формулу A теорії числення предикатів першого порядку *АП* в таку формулу B, яка знаходиться у ВНФ, що $\vdash A \Leftrightarrow B$ у ***ПL***.

Приклад 4.2.1. Звести формулу $\forall x\,F(x) \to \exists x\,H(x)$ до ВНФ.

Розв'язання. Користуючись кроками 1, 2, 4 алгоритму, отримаємо

$\forall x\,F(x) \to \exists x\,H(x) = \neg(\forall x\,F(x)) \vee \exists x\,H(x) = \exists x\,(\neg F(x)) \vee \exists x\,H(x) = \exists x\,(\neg F(x) \vee H(x))$.

Приклад 4.2.2. Отримати ВНФ для формули

$\forall x\ \forall y\ \forall z\,(P(x, y) \wedge P(y, z)) \to \exists z\,R(x, y, z)$.

Розв'язання. Користуючись кроками 1, 2, 3, 4 алгоритму, отримаємо

$\forall x\ \forall y\ \forall z\,(P(x, y) \wedge P(y, z)) \to \exists z\,R(x, y, z) =$

$= (\forall z\ \forall y\,(\neg \exists z\,(P(x, y) \wedge P(y, z)) \vee \exists z\,R(x, y, z) =$

$= \forall x\ \forall y\ \exists z\,(\neg P(x, y) \vee \neg P(y, z)) \vee \exists u\,R(x, y, и) =$

$= \forall x\ \forall y\ \exists z\ \exists u\,(\neg P(x, y) \vee \neg P(y, z) \vee R(x, y, и))$.

Оскільки матриця A формули Φ не містить кванторів, то її можна подати у кон'юнктивній нормальній формі (КНФ). Формула P знаходиться у кон'юнктивній нормальній формі, якщо вона має вигляд $P = P_1 \vee P_2 \vee \ldots \vee P_n$,

де $P_i = A_{1i} \wedge A_{2i} \wedge \ldots \wedge A_{ik}$, і кожне A_{ij} – це атомарна формула або її заперечення.

Алгоритм побудови КНФ має такі кроки.

1. 1. Вилучити зв'язки " $\Leftrightarrow$" і " $\to$" з формули A за допомогою правила а), як в алгоритмі ВНФ.
2. 2. Вилучити подвійне заперечення за допомогою правила $\neg(\neg F) = F$, і перенести знак " $\neg$" до атомів, користуючись законами де Моргана, як у алгоритмі побудови ВНФ.
3. 3. Для отримання нормальної форми формули A використати дистрибутивні закони:

$F \vee (G \wedge H) = (F \vee G) \wedge (F \vee H)$;

$F \wedge (G \vee H) = (F \wedge G) \vee (F \wedge H)$.

Приклад 4.2.3. Задану формулу $(P \wedge (Q \to R)) \to S$ перетворити у КНФ.

Розв'язання. Використовуючи кроки алгоритму 1, 2 і 3, отримаємо

$(P \wedge (Q \to R)) \to S = (P \wedge (\neg Q \vee R)) \to S = \neg\,(P \wedge (\neg Q \vee R)) \vee S =$

$= (\neg P \vee \neg\,(\neg Q \vee R)) \vee S = (\neg P \vee Q) \wedge (\neg P \vee \neg R) \vee S =$

$= (\neg P \vee Q \vee S) \wedge (\neg P \vee \neg R \vee S)$.

Приклад 4.2.4. Задану формулу $(P \vee \neg Q) \to R$ перетворити в КНФ.

Розв'язання. Використовуючи кроки наведеного вище алгоритму, отримаємо $(P \vee \neg Q) \to R = \neg\,(P \vee \neg Q) \vee R = (\neg P \wedge \neg\,(\neg Q)) \vee R = (\neg P \wedge Q) \vee R$.

4.3. Побудова доведень в аксіоматичній системі

Основною задачею аксіоматичної системи логіки предикатів є установлення істинності предикатних тотожностей і клауз. Позначимо через P_i будь-який предикат із задовільним числом аргументів, а через $(Q_i\,x_i)$– відповідну йому кванторну групу. Тоді, наприклад, закон дистрибутивності матиме такий вигляд

$(Q_1 x_1)\ P_1 \vee ((Q_2 x_2)\ P_2 \wedge (Q_3 x_3)\ P_3) = ((Q_1 x_1)\ P_1 \vee (Q_2 x_2)\ P_2) \wedge ((Q_1 x_1)\ P_1 \vee (Q_3 x_3)\ P_3)$.

У тому, що він виконується для одномісних предикатів, можна переконатися за допомогою процедури конкретизації:

$x = a, b$; $Q_1 x_1 = \forall x$; $P_1 = A(x)$;

$y = c, d$; $Q_2 x_2 = \exists y$; $P_2 = B(y)$;

$z = e, f$; $Q_3 x_3 = \forall z$; $P_3 = C(z)$;

$[A(a) \wedge A(b)] \vee \{[B(c) \vee B(d)] \wedge [C(e) \wedge C(f)]\} = \{[A(a) \wedge A(b)] \vee [B(c) \vee B(d)]\} \wedge \{[A(a) \wedge A(b)] \vee [C(e) \wedge C(f)]\}$.

Від того, що у квадратних дужках з'явиться замість диз'юнкції кон'юнкція і навпаки, а також замість одномісного предиката будуть фігурувати різні багатомісні предикати, – суть

тотожності не зміниться. Вона залишиться істинною внаслідок справедливості законів логіки множин і принципу суперпозиції, який стверджує, що заміна будь-якої константи іншою константою або навіть групою констант не може вплинути на істинність тотожності.

Ті самі висновки можуть бути виконані й по відношенню до логіки висловлювань, яка відрізняється від логіки Буля аксіомою порядку

$(Q_1 x_1)\ P_1, (Q_2 x_2)\ P_2 \Rightarrow (Q_1 x_1)\ P_1$.

Процедура конкретизації зводить предикатну аксіому порядку до простого висловлювання, тому якщо клауза істинна для висловлювань, то вона буде істинною і для предикатів.

4.4. Метод ідентифікації

Цей метод базується на перетворенні предикатної клаузи на клаузу логічного висловлювання, шляхом ідентифікації ідентичних квантованих предикатів. Алгоритм методу ідентифікації має такі кроки.

1. 1. Виділити однойменні квантовані предикати, позначивши їх відповідними літерами формул логіки висловлювань.
2. 2. Перетворити предикатну клаузу на клаузу логіки висловлювань, записавши її за допомогою ідентифікованих літер формул логіки висловлювань і замінивши при цьому всі логічні зв'язки розглянутої предикатної клаузи.
3. 3. Розглянути одержану клаузу логіки висловлювань на її істинність чи хибність, використовуючи один із методів пропозиційної логіки: § 2.4, § 2.5,

§ 2.6, § 2.7.

Приклад 4.4.1. Довести істинність предикатної клаузи $\forall x\ \exists y\ P(x, y) \to \to \forall v\ P(a, v),\ \forall x\ P(a, x) \to \exists x\ \forall y\ P(y, x) \Rightarrow\ \forall u\ \exists v\ P(u, v) \to \exists v\ \forall u\ P(u, v)$ методом ідентифікації.

Розв'язання. Використовуючи перший крок алгоритму, введемо позначення:

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

$B = \forall v\ P(a, v) = \forall x\ P(a, x);$

$C = \exists x\ \forall y\ P(y, x) = \exists v\ \forall u\ P(u, v).$

Згідно з другим кроком алгоритму, робимо перетворення предикатної клаузи на клаузу логіки висловлювань:

$A \to B, B \to C \Rightarrow A \to C.$

Доведення істинності клаузи логіки висловлювань виконано аксіоматичним методом у прикладі 4.3.2. Звідси випливає, що і предикатна клауза є теж істинна.

Приклад 4.4.2. Довести істинність предикатної тотожності

$\exists x\ \forall y \neg P(x,y) \vee \exists u\ \forall v\ P(v, u) \wedge\ \forall v\ \exists u\ P(v, u) \wedge \exists y\ \forall x\ P(x, y) =$

$= \exists x\ \forall y\ P(y, x) \wedge \forall u\ \exists v\ P(u, v)$ методом ідентифікації.

Розв'язання. Згідно з першим кроком алгоритму введемо такі позначення:

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

$B = \exists u\ \forall v\ P(v, u) = \exists y\ \forall x\ P(x, y) = \exists x\ \forall y\ P(y, x).$

Використовуючи другий крок алгоритму, робимо перетворення предикатної тотожності на тотожність алгебри Буля:

$(\neg A \vee B) \wedge (A \wedge B) = B \wedge A.$

Виконуючи елементарні логічні перетворення отримаємо

$B \wedge A = B \wedge A,$

що підтверджує істинність предикатної тотожності.

Приклад 4.4.3. Довести істинність предикатної клаузи

$\forall v\ \exists u\ P(u, v) \to \forall u\ \forall v\ P(u, v),\ \exists x\ \forall y\ P(x, y) \Rightarrow\ \forall u\ P(u, u)$ методом ідентифікації.

Розв'язання. Користуючись першим кроком алгоритму, введемо такі позначення:

$A_1 = \exists x\ \forall y\ P(x, y);\ A_2 = \forall v\ \exists u\ P(u, v);$

$B_1 = \forall u\ \forall v\ P(u, v);\ B_2 = \forall u\ P(u, u).$

Використовуючи другий крок алгоритму, робимо перетворення предикатної клаузи у клаузу логіки висловлювань:

1. $\square \to B_1,\ A_1 \Rightarrow B_2$ або $\neg A_2 \vee B_1,\ A_1, \neg B_2 \Rightarrow 0.$
2. Згідно з методом резолюції, § 2.7 суперечності можливі, якщо можливі дві інші суперечності, що випливають із клаузи логіки висловлювань:
3. $\square, \neg A_2 \Rightarrow 0,\ B_1, \neg B_2 \Rightarrow 0.$

Подання їх предикатними суперечностями

$\exists x\ \forall y\ P(x, y), \neg\ \forall v\ \exists u\ P(v, u) \Rightarrow 0,$

$\forall u\ \forall v\ P(u, v), \neg\ \forall u\ P(u, u) \Rightarrow 0,$

підтверджує істинність предикатної клаузи.

Приклад 4.4.4. Довести істинність предикатної клаузи

$\forall z\ B(z, z) \to \exists x \neg \exists y\ B(x, y),\ \exists z\ \forall y \neg A(z, y) \to\ \forall u\ \forall v\ B(v, u), \neg \exists x\ A(b, x) \Rightarrow \Rightarrow \exists z$

$\exists x \neg B(x, z) \wedge \exists z\ \forall y \neg A(z, x)$ методом ідентифікації.

Розв'язання. Згідно з першим кроком алгоритму введемо такі позначення:

$A_1 = \forall z\ \exists y\ A(z, y);\ A_2 = \exists x\ A(b, x);$

$B_1 = \forall u\ \forall v\ B\ (v, u) = \forall z\ \forall x\ B(x, z);$

$B_2 = \forall z\ B\ (z, z);\ B_3 = \forall x\ \exists y\ B(x, y).$

Використовуючи другий крок алгоритму, перетворюємо предикатну клаузу у клаузу логіки висловлювань:

$B_2 \to \neg B_3,\ \neg A_1 \to B_1, \neg A_2 \Rightarrow \neg\ B_1 \wedge \neg A_1.$

Замінивши імплікацію і привівши клаузу до суперечності, отримаємо

$\neg B_2 \vee \neg B_3,\ A_1 \vee\ B_1, \neg A_2 \Rightarrow 0.$

Згідно з методом резолюцій суперечності можливе тоді і тільки тоді, коли можливі інші суперечності клаузи, які випливають із клаузи логіки висловлювань: $A_1, \neg A_2 \Rightarrow 0,\ B_1, \neg B_2 \Rightarrow 0,\ B_1, \neg\ B_3 \Rightarrow 0.$

Подання їх предикатними суперечностями

1. $\forall\ z\ \exists y\ A\ (z, z), \neg\ \exists x\ A(b, x) \Rightarrow 0;$
2. $\forall\ u\ \forall v\ B\ (v, u), \neg\ \forall z\ B(z, z) \Rightarrow 0;$
3. $\forall\ z\ \forall x\ B(x, z), \neg\ \forall x\ \exists y\ B(x, y) \Rightarrow 0$

підтверджує істинність предикатної клаузи.

4.5. Метод резолюцій

В основу цього метода покладено не лише перетворення предикатної клаузи на клаузу логічного висловлювання, а подання останньої у вигляді кон'юктивної нормальної форми (КНФ) і суперечності. Алгоритм методу резолюцій має такі кроки.

1. Виділити однойменні квантові предикати, позначивши їх відповідними літерами формул логіки висловлювань.

2. Перетворити предикатну клаузу на клаузу логіки висловлювань, записавши її за допомогою ідентифікованих літер формул логіки висловлювань.

3. Перетворити формулу логіки висловлювань на КНФ і звести її до суперечності.

4. Виконати склеювання диз'юнктивних посилок, видаливши у них тавтологічні висловлювання або їх компонування, що приводить до суперечностей.

5. Якщо на кроці 4 алгоритму буде одержано загальну суперечність, то предикатна клауза істинна, а якщо ні – то хибна.

Приклад 4.5.1. Довести істинність предикатної клаузи

$\forall x\ \exists y\ P(x, y) \to \forall v\ P(a, v),\ \forall x\ P(a, x) \to \exists x\ \forall y\ P(y, x) \Rightarrow$

$\Rightarrow \forall u\ \exists vP(u, v) \to \exists\ v\ \forall\ u\ P(u, v)$

методом резолюцій.

Розв'язання. Використовуючи перший крок алгоритму, введемо позначення

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

$B = \forall v\ P(a, v) = \forall x\ P(a, x);$

$C = \exists x\ \forall y\ P(y, x) = \exists v\ \forall u\ P(u, v).$

Згідно з другим кроком алгоритму виконаємо перетворення предикатної клаузи на клаузу логіки висловлювань: $A \to B, B \to C \Rightarrow A \to C.$

Користуючись третім кроком алгоритму, перетворимо отриману клаузу логіки висловлювань у КНФ і зведемо її до суперечності

$\neg A \vee B, \neg B \vee C, \neg (\neg A \vee C) \Rightarrow 0.$

На четвертому кроці алгоритму виконаємо склеювання першої і другої посилок, у результаті одержимо посилку $\neg A \vee C$, яку склеюємо з третьою посилкою. Результатом цього склеювання є загальна суперечність, тому предикатна клауза – істинна.

Приклад 4.5.2. Довести істинність предикатної клаузи

$\forall u P(u, u, a) \vee \forall y \forall z P(y, y, z) \Rightarrow \exists q P(q, c, q) \exists q \forall u P(b, u, q) \exists q \forall u P(b, u, q)$ методом резолюцій.

Розв'язання. Використовуючи перший крок алгоритму, введемо позначення: $A = \forall u P(u, u, a)$; $B = \forall y \forall z P(y, y, z)$;

$A' = \exists q P(q, c, q)$; $B' = \exists q \forall u P(b, u, q)$.

Користуючись другим кроком алгоритму, виконаємо перетворення предикатної клаузи на клаузу логіки висловлювань: $A \vee B \Rightarrow A'; B'$.

Спростовуючи цю клаузу і використовуючи третій крок алгоритму, отримаємо $A \vee B, \neg A', \neg B' \Rightarrow 0$.

Згідно із четвертим кроком алгоритму суперечність у даному висловлюванні можливе, якщо можливі дві інші суперечності:

$A, \neg A' \Rightarrow 0$ $B, \neg B' \Rightarrow 0$.

Тобто:

$\forall u P(u, u, a), \neg \exists q P(q, c, q) \Rightarrow 0$;

$\forall y \forall z P(y, y, z), \neg \exists q \forall u P(b, u, q) \Rightarrow 0$.

Звідси і випливає істинність предикатної клаузи.

Затання для самоперевірки:

1. У чому різниця пропозиційної логіки від аксіоматичної системи логік?
2. Які аксіоми має аксіоматична система логік?
3. Запишіть формули аксіом аксіоматичної системи логік.
4. Які правила виведення є спільними для пропозиційної логіки і аксіоматичної системи логік?
5. На які групи і підгрупи діляться правила виведення?
6. Для яких цілей використовують правила введення і правила вилучення?
7. Які обмеження необхідні для правил узагальнення та існування. Які можуть бути наслідки, якщо не дотримуватися зазначених обмежень?
8. Сформулюйте правило перейменування вільних змінних.
9. У чому сутність правил перейменування зв'язних змінних?
10. Дайте визначення випередженої нормальної форми.
11. Сформулюйте алгоритм перетворення виразів довільної форми у ВНФ.
12. Сформулюйте алгоритм подачі матриці *A* формули *Ф* у кон'юнктивній нормальній формі.
13. Що є основною задачею логіки предикатів?
14. Який закон використовують для побудови доведень у логіці предикатів?
15. Закони яких теорій використовують при побудові доведень у логіці предикатів і чому?
16. На якому принципі базується метод ідентифікації?
17. Сформулюйте алгоритм методу доведення предикатних клауз методом ідентифікації.
18. При яких умовах і для яких предикатних клауз або тотожностей можна застосовувати алгоритм ідентифікації?
19. Які основні принципи покладені у метод резолюцій?
20. Сформулюйте алгоритм методу доведення предикатних клауз методом резолюцій.
21. При яких кінцевих умовах предикатна клауза буде істинною, а при яких – хибною?

Задачі для самостійного розв'язування

1. Довести, що
а) $\vdash \forall x \forall y A \to \forall y \forall x A$;
б) $\vdash \forall x (A \to B) \to (\forall x A \to \forall x B)$.
2. Довести, якщо терм і вільний для змінної *x* у формулі $A(x)$, то:
а) $\forall x A(x) \vdash A(t)$;
б) $A(t) \vdash \exists x A(x)$.
3. Довести, що для будь-яких форм л *A* і *B*:
а) $A, B \vdash A \wedge B$;
б) $A \to C, B \to D, \neg A \to B \vdash \neg C \to D$;
в) $\vdash A \vee B \sim \neg (\neg A \wedge \neg B)$.
4. Звести до ВНФ такі вирази:
а) $\neg \forall y \exists x (A(x) \to B(y))$;
б) $\exists x \forall y B(x, y) \Leftrightarrow \exists x A(x)$;
в) $\forall x A(x) \to \neg (\forall x (B(y) \vee \forall y C(x, y))$.
5. Задані формули:
а) $(\neg P \wedge Q) \to \neg R$; б) $(P \wedge (R \to \neg Q)) \to S$; в) $\neg (\neg P \vee \neg Q) \to (\neg R \vee S)$;
перетворити у КНФ.
6. Істинність заданих предикатних клауз довести методом ідентифікації:
а) $\exists x \forall y P(x, y) \to \forall v \forall u P(u, v)$, $\exists x \forall v P(x, v) \to \forall x \forall y P(x, y)$;
б) $\exists x \forall y P(x, y) \to \forall u \forall v P(v, u)$, $\forall x \forall v P(x, v) \to \exists y \exists u P(y, u)$, $\neg \exists v \exists x P(x, v) \to \neg \exists y \forall x P(x, y)$;
в) $\forall x \forall y P(x, y) \to \exists x \forall y Q(x, y)$, $\exists x \exists y \neg R(x, y) \to \exists x \forall y \neg Q(x, y)$, $\exists x \exists y \neg R(x, y) \to \forall x \forall y \neg P(x, y)$.
7. Довести істинність предикатних клауз методом резолюцій:
а) $[\forall x \forall y P(x, y) \to \exists x Q(x)] \to [\exists x \neg \forall y P(x, y) \wedge \exists x \exists y P(x, y)] \to \neg \exists x \exists y \neg P(x, y) \vee \neg \forall x \forall y \neg P(x, y)$;
б) $\exists x P(x, y) \to \neg \exists y \neg P(x, y) \to \forall x \exists y Q(x, y)$, $\forall y P(x, y) \to \neg \forall y \neg P(x, y)$, $\forall y P(x, y) \to \neg \exists x \neg P(x, y) \to \forall y \exists x Q(x, y)$;
в) $\exists x \exists y P(x, y)$, $\forall x \exists y P(x, y) \to \exists y Q(y)$,
$\forall x Q(x) \vee [\exists y \exists x P(x, y) \to \neg \exists x \forall y \neg P(x, y)]$,
$\neg \exists x \neg Q(x) \to \exists x Q(x, y) \to \neg \forall y \neg P(x, y)$.

Коментарі. У цьому розділі система аксіом і правил виведення взяті з [14,19,20]. Більш детальне викладення цього матеріалу можна знайти в [12,27]. Властивості числення предикатів першого порядку і випереджені нормальні форми випливають з [27], а побудова доведень узята з [1,7,23,26].

1

Тема 5 Некласична математична логіка

- Нечітка логіка
  - 5.1. Основні визначення
  - 5.2. Операції над нечіткими множинами
  - 5.3. Поняття нечіткої і лінгвістичної змінної
  - 5.4. Нечіткі висловлювання та нечіткі предикати
  - 5.4.1. Основні логічні операції з нечіткими висловлюваннями
  - 5.5. Правило нечіткого логічного виводу
  - 5.5.1 Композиційне правило нечіткого виводу Заде

Нечітка логіка

Логіка в звичайному сенсі слова є уявленням механізму мислення, повинна бути завжди строго формалізованою. Однак в дійсності існує не одна логіка (наприклад, булева), а стільки, скільки ми побажаємо, тому що все визначається вибором відповідної системи аксіом. Звичайно, як тільки аксіоми прийняті, всі твердження, що побудовані на їх основі, повинні бути строгими, без протиріч пов'язані за правилами, встановленими в цій системі аксіом.

Нечітка логіка є узагальненням класичної логіки на випадок, коли істинність розглядається як лінгвістична змінна, що приймає значення типу: "дуже істинно" , "більш-менш істинно" , "не дуже хибно" і т.п. Зазначені лінгвістичні значення представляються нечіткими множинами. Основна відмінність від класичної логіки полягає в тому, що замість значень " Істина " і " Хибність " в нечіткій логіці використовується ступінь істинності, що приймає значення з нескінченної множини від 0 (Хибність) до 1 (Істина) включно. Отже логічні операції вже не можуть бути представленими таблицями істинності. У нечіткій логіці вони задаються функціями і лише в крайніх випадків - коли значення змінних виключно 1 або 0 - згадані вище функції дають таблиці істинності операцій класичної логіки.

Основи нечіткої логіки були закладені в кінці 1960-х років у працях відомого американського математика Лотфі А. Заде, що започаткував теорію нечітких множин. Термін "нечітка логіка" використовується зазвичай в двох різних значеннях. У вузькому сенсі, нечітка логіка - це логічне числення, що є розширенням багатозначної логіки. В її широкому сенсі, який сьогодні є переважаючим у використанні, нечітка логіка рівнозначна теорії нечітких множин. З цієї точки зору, нечітка логіка у вузькому сенсі є розділом нечіткої логіки в широкому сенсі.

5.1. Основні визначення

Означення 5.1.1. Нечіткою множиною $\tilde{X}$ з універсальної множини *U* називають сукупність упорядкованих пар $\tilde{X}$ = {($\mu_x(a)$, *a*): *a* ∈ *U* }, де $\mu_x(a)$ – функція належності нечіткої множини ($\mu_x : U \to [0, 1]$), що приписує кожному елементу *a* ∈ *U* ступінь його належності $\tilde{X}$.

Використовують спеціальний запис нечітких множин. Якщо *U* є універсумом зі скінченою кількістю елементів, то $\tilde{X}$ можна представити так:

$$\tilde{X} = \frac{\mu_x(a_1)}{a_1} + \frac{\mu_x(a_2)}{a_2} + \ldots + \frac{\mu_x(a_n)}{a_n} = \sum_{i=1}^{n} \frac{\mu_x(a_i)}{a_i} \quad (5.1)$$

Якщо *U* є нескінченою множиною, то $\tilde{X}$ записують так:

$$\tilde{X} = \int_U \frac{\mu_x(a)}{a} \quad (5.2)$$

Знаки Σ і $\int$ [?][?] в формулах (5.1) та (5.2) мають інтерпретацію множини елементів. Мають місце також табличний та графічний спосіб представлення $\tilde{X}$.

Приклад 5.1.1. Визначити за допомогою нечіткої множини поняття "Людина середнього зросту ", задавши універсум множиною *U* = {150, 155, 160,... 190} (зріст заданий у см).

Розв'язання. Виходячи з умови, з позицій першого дослідника, таке поняття може бути визначеним такою нечіткою множиною

$$\tilde{A} = \frac{0}{150} + \frac{0}{155} + \frac{0{,}1}{160} + \frac{0{,}15}{165} + \frac{0{,}3}{170} + \frac{0{,}9}{175} + \frac{1}{180} + \frac{1}{185} + \frac{0{,}7}{190}.$$

Інший дослідник, вважаючи, що людина має середній зріст 175-180 см, представить шуканий результат іншою нечіткою множиною –

$$\tilde{B} = \frac{0}{150} + \frac{0}{155} + \frac{0{,}2}{160} + \frac{0{,}4}{165} + \frac{0{,}8}{170} + \frac{1}{175} + \frac{1}{180} + \frac{0{,}9}{185} + \frac{0{,}4}{190}.$$

Приклад 5.1.**2**. Введемо означення нечітких множин : $\tilde{X}$ - " малий зріст людини ", $\tilde{Y}$ - " середній зріст людини ", $\tilde{Z}$ - " великий зріст людини ". В якості універсуму застосуємо U = {x∈R : 0 ≤ x ≤ a_{max}}, де a_{max}– можливий максимальний зріст людини в см.

На рисунку 5.1 подано графіки функцій належності, що визначають уведені множини.

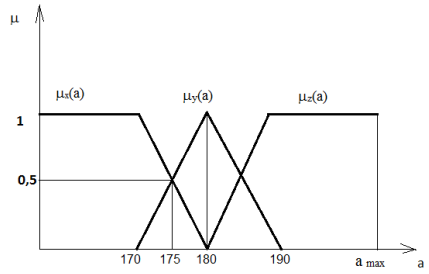


Рисунок 5.1 – Графіки функцій належності множин $\tilde{X}$, $\tilde{Y}$, $\tilde{Z}$.

У точці a = 175 см, $\mu_x(a) = \mu_y(a) = 0{,}5$, тоді як $\mu_z(a) = 0$.

Функції належності зручно задавати в параметричній формі.Найбільшу популярність отримали трикутна, трапецевидна, гаусова, сигмоїдальна та Пі-подібна функції належності.

Рисунок 5.2 – Види графіків функцій належності (в порядку слідування: Z – функція; П - функція; Л - функція; S - функція).

Для побудови функцій належності можна використати метод, що базується на статистичній обробці думок групи експертів. У наведених вище прикладах використані прямі методи, коли експерт або просто задає для будь-якого a ∈ U значення $\mu_x(a)$, або визначає функцію належності. Як правило, прямі методи задання функції належності використовуються для вимірних понять, таких як швидкість, година, відстань, тиск, температура і так далі, тобто коли виділяються полярні значення.

Непрямі методи визначення значень функції належності використовуються у випадках, коли немає елементарних вимірних властивостей для визначення нечіткої множини. Як правило, це методи попарних порівнянь.

Означення 5.1.2. Висотою нечіткої множини $\tilde{X}$ називають величину

$$h(\tilde{X}) = \max_{a \in U} \mu_X(a).$$

Приклад 5.1.3. Якщо U = {2, 3, 4, 5, 6}, а $\tilde{X} = \frac{0{,}1}{2} + \frac{0{,}5}{3} + \frac{0}{4} + \frac{0{,}8}{5} + \frac{0}{6}$, то висотою нечіткої величини $\tilde{X}$ буде $h(\tilde{X}) = 0{,}8$.

Означення 5.1.3. Нечітку множину $\tilde{X}$ називають нормальною, якщо $h(\tilde{X}) = 1$. Якщо множина $\tilde{X}$ не є нормальною, то нормалізацію нечіткої множини можна провести за правилом $\mu_{X_n}(a) = \frac{\mu_X(a)}{h(\tilde{X})}$, де $\tilde{X}_n$ вже стає нормальною нечіткою множиною.

Приклад 5.1.4. Нечітку множину прикладу 5.1.3 нормалізувати.

$\tilde{X}_n = \frac{0{,}125}{2} + \frac{0{,}625}{3} + \frac{0}{4} + \frac{1}{5} + \frac{0}{6}$.

Розв'язання. За означенням 5.1.3 одержимо

Означення 5.1.4. Нечітку множину $\tilde{X}$ називають субнормальною, якщо $h(\tilde{X}) < 1$.

Означення 5.1.5. Нечітку множину $\tilde{X}$ називають порожньою, якщо $\mu_x(a) = 0$ для всіх *a* ∈ *U*.

Означення 5.1.6. Нечітка множина $\tilde{X}$ називається унімодальною, якщо $\mu_x(a) = 1$ лише для одного елемента з *U*.

Означення 5.1.7. α-перерізом нечіткої множини $\tilde{X}$ називають множину X_α ={a ∈ U: $\mu_x(a) \geq \alpha$} (α ∈ [0,1]). Множина X_α відноситься до звичайних (чітких) множин.

Приклад 5.1.5. Для заданої універсальної множини *U* = {0,1,2..,10} і нечіткої множини $\tilde{X}$ = $\frac{0{,}5}{3} + \frac{0{,}8}{4} + \frac{1}{5} + \frac{1}{6} + \frac{0{,}8}{7} + \frac{0{,}6}{8}$ визначити $X_{0,6}$ і X_1.

Розв'язання. Виходячи з означення 5.1.5 $X_{0,6}$ ={4,5,6,7,8}; X_1 ={5,6}.

Означення 5.1.8. Лінгвістичною змінною називається змінна, значеннями якої можуть бути слова або словосполучення деякої природної або штучної мови.

Означення 5.1.9. Терм-множиною називається множина всіх можливих значень лінгвістичної змінної.

Означення 5.1.10. Термом називається будь-який елемент терм-множини. В теорії нечітких множин терм формалізується нечіткою множиною за допомогою функції належності.

Приклад 5.1.6. Розглянемо змінну "швидкість автомобіля", яка оцінюється за шкалою "низька", "середня", "висока" і "дуже висока".

У цьому прикладі лінгвістична змінна є "швидкість автомобіля", термами - лінгвістичні оцінки "низька", "середня", "висока" і "дуже висока", які і складають терм-множину.

Означення 5.1.11. Фаззіфікацією називається процедура перетворення чітких даних в нечітку множину.

Не вдаючись у подробиці, процедура фаззіфікації виявляється в формалізації даних, як лінгвістичних змінних, введення всіх можливих термів, що їх характеризують, та побудові функцій належності для кожного терму.

Означення 5.1.12. Дефаззіфікацією (в нечіткій логіці) називається процедура перетворення нечіткої множини в чітке число.

В теорії нечітких множин процедура дефаззіфікації аналогічна знаходження характеристик становища (математичного очікування, моди, медіани) випадкових величин в теорії ймовірності. Наприклад, таким числом може стати максимум функції належності, центр ваги функції належності, чи щось інше.

5.2. Операції над нечіткими множинами

Визначення нечітких теоретико - множинних операцій можуть бути узагальнені зі звичайної теорії множин.

Означення 5.2.1. Дві нечіткі множини $\widetilde{A}$ і $\widetilde{B}$, що задані на U, є рівними, якщо вони складаються з одних і тих же елементів для всіх $x \in U$ та $\mu_A(x) = \mu_B(x)$. Позначають як $\widetilde{A} = \widetilde{B}$.

Означення 5.2.2. Дві нечіткі множини $\widetilde{A}$ і $\widetilde{B}$ доповнюють одна одну, якщо $\mu_A(x) = 1 - \mu_B(x)$ для всіх $x \in U$

Означення 5.2.3. Перетином двох нечітких множин $\widetilde{A}$ і $\widetilde{B}$ є така множина $\widetilde{D} = \widetilde{A} \cap \widetilde{B}$, що складається з елементів для всіх $x \in U$, що одночасно належать $\widetilde{A}$ і $\widetilde{B}$ з функцією належності $\mu_D(x) = \min(\mu_A(x), \mu_B(x))$.

Означення 5.2.4. Об'єднанням двох нечітких множин $\widetilde{A}$ і $\widetilde{B}$ є така множина $\widetilde{D} = \widetilde{A} \cup \widetilde{B}$, що складається з елементів для всіх $x \in U$, що належать $\widetilde{A}$ або $\widetilde{B}$ з функцією належності $\mu_D(x) = \max(\mu_A(x), \mu_B(x))$.

Означення 5.2.5. Різницею двох нечітких множин $\widetilde{A}$ і $\widetilde{B}$ є така множина $\widetilde{D} = \widetilde{A} - \widetilde{B}$, що складається з елементів для всіх $x \in U$, що мають функцію належності $\mu_D(x) = \mu_A(x) - \mu_B(x)$, якщо $\mu_A(x) > \mu_B(x)$, інакше $\mu_D(x) = 0$.

На рис. 5.3 представлені графічно основні операції над нечіткими множинами: а) $\widetilde{A} \subset \widetilde{B}$; б) $\widetilde{A} = 1 - \widetilde{B}$; в) $\widetilde{A} \frown \widetilde{B}$ г) $\widetilde{A} \smile \widetilde{B}$.

Означення 5.2.6. Декартовим добутком $\widetilde{A} \times \widetilde{B}$ двох нечітких множин $\widetilde{A}$ і $\widetilde{B}$, визначених відповідно на універсамах U_1 та U_2, є нечітка множина $\widetilde{D}$ пар (кортежів), визначених U_1 та U_2 з функцією належності $\mu_D(x) = \min(\mu_A(x), \mu_B(x))$. Тобто $\widetilde{D} = \widetilde{A} \times \widetilde{B} = \{(\min(\mu_A(x), \mu_B(x)), (a,b)) : a \in U_1, b \in U_2\}$.

Приклад 5.2.1. Для заданих нечітких множин

$$\widetilde{A} = \frac{0}{150} + \frac{0}{155} + \frac{0,1}{160} + \frac{0,15}{165} + \frac{0,3}{170} + \frac{0,9}{175} + \frac{1}{180} + \frac{1}{185} + \frac{0,7}{190},$$

$$\widetilde{B} = \frac{0}{150} + \frac{0}{155} + \frac{0,2}{160} + \frac{0,4}{165} + \frac{0,8}{170} + \frac{1}{175} + \frac{1}{180} + \frac{0,9}{185} + \frac{0,4}{190}$$

знайти $\widetilde{A} \frown \widetilde{B}$ та $\widetilde{A} \smile \widetilde{B}$.

Розв'язання. За означеннями 5.2.4 та 5.2.5 знаходимо

$$\widetilde{A} \frown \widetilde{B} = \frac{0}{150} + \frac{0}{155} + \frac{0,1}{160} + \frac{0,15}{165} + \frac{0,3}{170} + \frac{0,9}{175} + \frac{1}{180} + \frac{0,9}{185} + \frac{0,4}{190},$$

$$\widetilde{A} \smile \widetilde{B} = \frac{0}{150} + \frac{0}{155} + \frac{0,2}{160} + \frac{0,4}{165} + \frac{0,8}{170} + \frac{1}{175} + \frac{1}{180} + \frac{1}{185} + \frac{0,7}{190}.$$

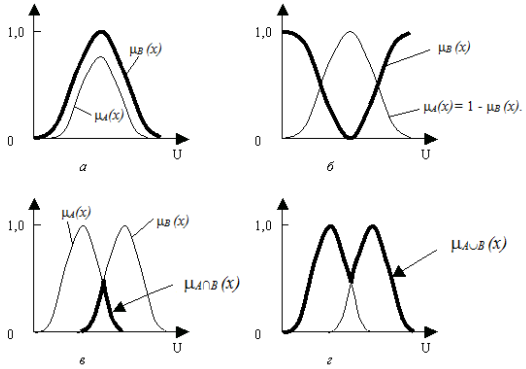


Рисунок 5.3 – Операції над нечіткими множинами.

Аналогічно операціям на чітких множинах, вводиться поняття нечіткого відношення.

Означення 5.2.7. Нечітким відношенням $\widetilde{R}$ між множинами $\widetilde{A}$ і $\widetilde{B}$ називається підмножина їх декартового добутку, тобто $\widetilde{R} \subset \widetilde{A} \times \widetilde{B}$.

5.3. Поняття нечіткої і лінгвістичної змінної

При описі об'єктів та явищ за допомогою нечітких множин використовують поняття нечітких і лінгвістичних змінних.

Означення 5.3.1. Нечіткою змінною називають змінну, яка характеризується трійкою $\langle x, U, \widetilde{A} \rangle$ [?][?], де x – ім'я змінної; U – універсальна множина (область визначення x); $\widetilde{A}$ – нечітка множина на U, що описує функцію належності змінної.

Означення 5.3.2. Лінгвістичною змінною називають змінну, яку можна описати набором даних $\langle x, T, U, G, M \rangle$, де x – ім'я лінгвістичної змінної, T – терм-множина (сукупність значень змінної), кожний елемент якої (терм) є нечіткою множиною, U – універсальна множина, G – синтаксична процедура (правило), що породжує терми множини T на U, M – семантична процедура (правило), яка кожному терму задає функцію належності.

Приклад 5.3.1. Розглянемо лінгвістичну змінну з ім'ям "температура в кімнаті". Тоді набір даних, що її характеризує можна визначити так:

Універсальну множину приймемо U = {x ∈ N: 5 ≤ x ≤ 35}.

Терм-множину виберемо T={"прохолодно", "комфортно", "жарко"}, в якій терми є нечіткими множинами з функціями належності:

$$\mu_{\text{"прохолодно"}}(u) = \frac{1}{1 + \left(\frac{u-10}{7}\right)^{12}},$$

$$\mu_{\text{"комфортно"}}(u) = \frac{1}{1 + \left(\frac{u-20}{3}\right)^{6}},$$

$$\mu_{\text{"жарко"}}(u) = \frac{1}{1 + \left(\frac{u-30}{6}\right)^{10}}.$$

3. Синтаксичні правила G, що породжують нові терми задамо за допомогою словосполучень (квантифікаторів) - "не", "дуже", "більш-менш".
4. Семантичні правила M представлені в табл. 5.1.

Таблиця 5.1. Правила розрахунку функцій належності

Квантифікатор	Функція належності
Не t	$1 - \mu_t(u)$
Дуже t	$(\mu_t(u))^2$
Більш-менш t	$\sqrt{\mu_t(u)}$

Графіки функцій належності термів "прохолодно", "не дуже прохолодно", "комфортно","більш-менш комфортно","жарко" та "дуже жарко" лінгвістичної змінної "температура в кімнаті" представлені на рис. 5.4.

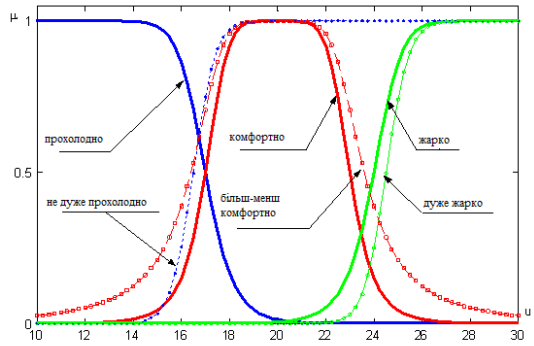


Рисунок 5.4 – Графіки належності термів лінгвістичної змінної.

5.4. Нечіткі висловлювання та нечіткі предикати

Поряд з поняттям нечіткої множини, Л. Заде запропонував узагальнення класичної логіки на основі розгляду нескінченної кількості значень істинності. Тобто, множина значень істинності висловлювань узагальнюється до інтервалу дійсних значень [0, 1], що дозволяє висловлюванню приймати будь-яке значення істинності з цього інтервалу. Це чисельне значення буде собою являти кількісну оцінку ступеня істинності висловлювання, що дозволяє побудувати логічну систему, в рамках якої виявилося можливим виконувати міркування з невизначеністю і оцінювати істинність висловлювань типу: “Швидкість автомобіля надмірно висока”, “Тиск в системі вельми значний”, “ Висота польоту літака гранично низька ” та ін. Вихідним поняттям нечіткої логіки є поняття елементарного нечіткого висловлювання.

Означення 5.4.1. Елементарним нечітким висловлюванням називається осмислений вираз звичайної мови, якому можна приписати значення істинності чи хибності тільки з деякою мірою впевненості.

Елементарні нечіткі висловлювання для зручності будемо позначати такими же літерами, що й нечіткі множини, й будемо ототожнювати з лінгвістичними змінними (див. означення 5.1.8, 5.3.2).

Приклад 5.4.1. Простий вислів “тиск великий” припускає, що лінгвістична змінна “тиск” має терм “великий”, для якого на універсальній множині U задається нечітка множина, яка визначається функцією належності.

Приклад 5.4.2. Нижче наводиться кілька прикладів елементарних нечітких висловлювань:

1. О. Бендер має досить високий зріст.
2. Завтра буде хмарно з проясненнями.
3. 3 - мале число.
4. Можливо, нам подадуть гарячу каву.

В цьому прикладі невизначеність нечітких висловлювань має різну природу. Невизначеність оцінки істинності у висловлюваннях 1,3 пов'язана з нечіткістю визначення понять “високий зріст” та “мале число”. Так, поняття “високий зріст” можна описати нечіткою множиною, щось на зразок з прикладом 5.1.1. Що стосується висловлювань 2 і 4, то тут окрім визначення нечітких змінних “похмура погода” і “гаряча кава” слід оцінити їх істинність відносно деякого моменту часу в майбутньому. Спільним для всіх цих висловлювань є та обставина, що значення їх істинності можна кількісно оцінювати дійсним числом з інтервалу [0, 1].

Для оцінки ступеня істинності довільної нечіткого висловлення зручно ввести в розгляд спеціальне відображення T, що діє з множини нечітких висловлювань $\widetilde{U}$ в інтервал [0, 1], тобто $T : \widetilde{U} \to [0, 1]$. Це відображення будемо називати відображенням істинності нечітких висловлювань. У цьому випадку значення істинності деякого нечіткого висловлення $\widetilde{A} \subset \widetilde{U}$ будемо позначати через $T(\widetilde{A})$. Так, якщо позначити нечітке висловлення 1 з прикладу 5.4.2 через $\widetilde{A}_1$, то його істинність формально може бути записана як $T(\widetilde{A}_1)$, а кількісно дорівнює, наприклад, 0,7, тобто $T(\widetilde{A}_1) = 0,7$.

Означення 5.4.2. Нечіткий предикат $P(<x_1, x_2, ..., x_k>)$ або, більш строго, k-місний нечіткий предикат, формально визначається як деяке відображення з декартового добутку універсумів $X_1, X_2, ..., X_k$ в деяку цілком впорядковану множину значень істинності, зокрема, в інтервал [0,1], тобто

$P: X_1 \times X_2 \times ... \times X_k \to [0, 1]$.

За аналогією з звичайними (чіткими) предикатами, змінні $x_1, x_2, ..., x_k$ називаються предметними змінними нечіткого предиката $P (<x_1, x_2, ..., x_k>)$, а декартовий добуток універсумів $X_1 \times X_2 \times ... \times X_k$ - його предметною областю.

Нечітке узагальнення логіки предикатів першого порядку, так само як і відповідні їй нечіткі числення, не знайшли широкого застосування при вирішенні прикладних завдань. Найбільш конструктивним напрямком в нечіткій логіці залишається використання нечітких висловлювань у формі означення лінгвістичних змінних. У цьому випадку нечіткі висловлювання можуть комбінуватися за допомогою нечітких логічних операцій або зв'язок, щось на зразок операцій над нечіткими множинами (див. розділ 5.2) .

5.4.1. Основні логічні операції з нечіткими висловлюваннями

Очевидно, що прийнятий в математичній логіці спосіб визначати логічні операції за допомогою таблиць істинності не може бути використаний в нечіткої логіки. Причина цього полягає в континуальної потужності множини значень істинності [0, 1]. Саме тому центральну роль у визначенні логічних операцій в нечіткої логіки набуває відображення істинності T, яке має допоміжне значення в класичній математичній логіці.

Нехай $\widetilde{U}$ - деяка множина елементарних нечітких висловлювань, а

$T: \widetilde{U} \to [0, 1]$ - відображення істинності висловлювань.

Означення 5.4.3. Запереченням нечіткого висловлювання $\widetilde{A}$ (записується як $\neg \widetilde{A}$ і читається “ не $\widetilde{A}$ ”, "невірно, що $\widetilde{A}$ ») називається унарна логічна операція, результат якої є нечітким висловлюванням, значення істинності якого за визначенням приймає значення:

$T(\neg \widetilde{A}) = 1 - T(\widetilde{A})$. (5.3)

Приклад 5.4.3. Застосуємо операцію заперечення до нечіткого висловлювання $\widetilde{A}$ - “О. Бендер має досить високий зріст” (див. приклад 5.4.2). Висловлювання $\neg \widetilde{A}$ -“Неправильно, що О. Бендер має досить високий зріст” буде мати значення істинності $T(\neg \widetilde{A}) = 1 - T(\widetilde{A}) = 1 - 0,7 = 0,3$.

Означення 5.4.4. Кон'юнкцією нечітких висловлювань $\widetilde{A}$ та $\widetilde{B}$ (записується як: $\widetilde{A} \wedge \widetilde{B}$ і читається – “ $\widetilde{A}$ і $\widetilde{B}$ ”) називається бінарна логічна операція, результатом якої являється нечітке висловлювання, істинність якого визначається за формулою:

$T(\widetilde{A} \wedge \widetilde{B}) = \min\{T(\widetilde{A}), T(\widetilde{B})\}$. (5.4)

Означення 5.4.5. Диз'юнкцією нечітких висловлювань $\widetilde{A}$ та $\widetilde{B}$ (записується як: $\widetilde{A} \vee \widetilde{B}$ і читається – “ $\widetilde{A}$ або $\widetilde{B}$ ”) називається бінарна логічна операція, результатом якої являється нечітке висловлювання, істинність якого визначається за формулою:

$T(\widetilde{A} \vee \widetilde{B}) = \max\{T(\widetilde{A}), T(\widetilde{B})\}$. (5.5)

Означення 5.4.6. Нечіткою імплікацією або просто - імплікацією нечітких висловлювань $\widetilde{A}$ і $\widetilde{B}$ (записується як $\widetilde{A} \supset \widetilde{B}$ і читається – “з $\widetilde{A}$ слідує $\widetilde{B}$ ”, “ЯКЩО $\widetilde{A}$, ТО $\widetilde{B}$ ”) називається бінарна логічна операція, результат якої є нечітким висловлюванням, істинність якого може приймати значення, яке визначається за однією з наступних формул.

Нечітка імплікація Заде:

$T(\widetilde{A} \supset \widetilde{B}) = \max\{\min\{T(\widetilde{A}), T(\widetilde{B})\}, 1 - T(\widetilde{A})\}$. (5.6)

Цю форму називають також класичною нечіткою імплікацією.

Нечітка імплікація Гьоделя :

$T(\widetilde{A} \supset \widetilde{B}) = \max\{T(\neg \widetilde{A}), T(\widetilde{B})\}$. (5.7)

Нечітка імплікація Мамдані :

$T(\widetilde{A} \supset \widetilde{B}) = \min\{T(\widetilde{A}), T(\widetilde{B})\}$. (5.8)

Нечітка імплікація Лукасевича:

$T(\widetilde{A} \supset \widetilde{B}) = \min\{1, 1 - T(\widetilde{A}) + T(\widetilde{B})\}$. (5.9)

Нечітка імплікація Гогена:

$T(\widetilde{A} \supset \widetilde{B}) = \min\{1, T(\widetilde{B})/T(\widetilde{A})\}$, де $T(\widetilde{A}) > 0$. (5.10)

Існують і інші способи обчислення нечіткої імплікації.

Нечітка імплікація відіграє важливу роль у процесі нечітких логічних умовиводів. Так само, як і в математичній логіці перший її операнд (нечітке висловлювання) називається засновком, а другий - висновком. Хоча класична нечітка імплікація знаходить найбільше застосування при

розв'язанні прикладних задач і вона залишається справедливою у випадку звичайних висловлювань класичної логіки, інші способи обчислення нечіткої імплікації в окремих ситуаціях виявляються більш ефективними.

Приклад 5.4.4.Розглянемо складене нечітке висловлення у формі нечіткої імплікації: "Якщо О. Бендер має досить високий зріст, то завтра буде хмарно з проясненнями", при цьому істинність входять до нього елементарних нечітких висловлювань дорівнює Т($\widetilde{A}$) = 0,7 і Т ($\widetilde{B}$) = 0,2. Знайти значення істинності імплікації.

Розв'язання. Тоді істинність цієї нечіткої імплікації, обчислена за формулою (5.6), дорівнює Т($\widetilde{A} \supset \widetilde{B}$) = 0,3, за формулою (5.8) - Т($\widetilde{A} \supset \widetilde{B}$)= 0,2, за формулою (5.9) - Т($\widetilde{A} \supset \widetilde{B}$) = 0,5, за формулою (5.10) - Т($\widetilde{A} \supset \widetilde{B}$) =0,29.

5.5. Правило нечіткого логічного виводу

Основним правилом виводу в класичній логіці є правило modus ponens, згідно з яким ми можемо судити про істинність висловлювання В по істинності висловлювання А і імплікації А → В (див. розділ 2). Це правило виводить висновок "В є істинно ", якщо відомо, що "А є істинно " і існує правило "Якщо А, то В" (А і В - чіткі логічні твердження). Нижче ми приведемо спосіб формалізації наближених міркувань, заснований на поняттях та операціях в нечіткій логіці.

Поняття нечіткого виводу займає центральне місце в нечіткій логіці і в теорії нечіткого управління. Цей процес включає в себе всі основні концепції теорії множин: функції належності, лінгвістичні змінні, методи нечіткої імплікації і т.д. Розробка й застосування систем нечіткого виводу включає в себе ряд етапів, реалізація яких виконується на основі положень нечіткої логіки, що розглянуті вище (рис. 5.5).

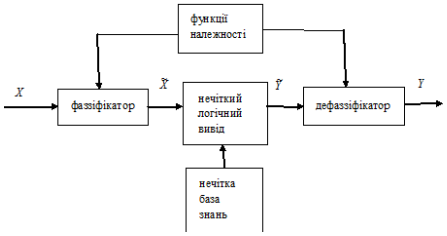


Рисунок 5.5 – Система нечіткого логічного виводу

Алгоритми нечіткого висновку розрізняються головним чином видом використовуваних правил, логічних операцій і різновидом методу дефаззіфікації. Розроблено моделі нечіткого висновку Мамдані, Сугено, Ларсена, Цукамото.

Означення 5.5.1. Нечіткою базою знань називається сукупність нечітких правил "Якщо – то", що визначають взаємозв'язок між входами і виходами і досліджуваного об'єкта.Узагальнений формат нечітких правил такий – "Якщо *засновок правила*, то *висновок правила*".

Засновок правила або антецедент являє собою твердження типу "х є низький", де "низький" -; це терм (лінгвістичне значення), заданий нечіткою множиною на універсальній множині лінгвістичної змінної х. Квантифікатори "дуже", "більш-менш", "не", "майже" і т.п. можуть використовуватися для модифікації термів антецедента.

Висновок або наслідок правила являє собою твердження типу "у є d", в якому значення вихідної змінної (d) може задаватися:

1. 1. нечітким термом: "у є високий";
2. 2. класом рішень: "у є бронхіт";
3. 3. чіткої константою: "у = 5";
4. 4. чіткої функцією від вхідних змінних: "у = 5 +4 * х".
5. **Приклад 5.5.1.** Наступна нечітка база знань описує залежність між віком водія (х) і можливістю дорожньо-транспортної пригоди (у):

"Якщо х = Молодий, то у = Висока";
"Якщо х = Середній, то у = Низька";
"Якщо х = Дуже старий, то у = Висока".

Розв'язання. Представимо засновок нечітких правил лінгвістичною змінною х, що має 3 терми: "молодий", "середній","дуже старий", висновок – у (можливість ДТП) має терми "низька" і "висока". Відповідні функції належності представлені на рис. 5.6.

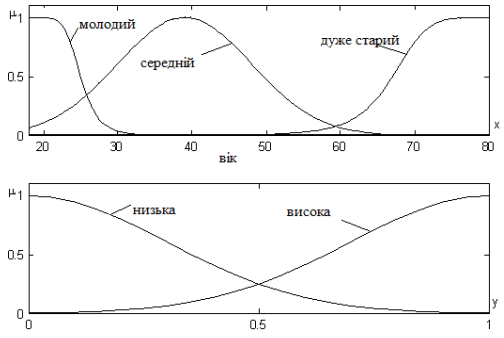


Рисунок 5.6 – Функції належності термів змінних х і у.

5.5.1 Композиційне правило нечіткого виводу Заде

Означення 5.5.2. Нечітким логічним висновком називається отримання висновку у вигляді нечіткої множини, відповідного поточним значеннях входів, з використанням нечіткої бази знань і нечітких операцій.

Означення 5.5.3. Композиційне правило виведення Заде формулюється таким чином: якщо відомо нечітке відношення $\widetilde{R}$ між вхідними (х) і вихідний (у) змінними, то при нечіткому значенні вхідної змінної $x = \widetilde{A}$, нечітке значення вихідної змінної визначається так: $y = \widetilde{A} \circ \widetilde{R}$, де $\circ$ - максмінна композиція.

Приклад 5.5.2. Дано нечітке правило "Якщо $x = \widetilde{A}$, то $y = \widetilde{B}$" з нечіткими множинами $\widetilde{A} = \frac{0}{1} + \frac{0,1}{2} + \frac{0,5}{3} + \frac{0,8}{4} + \frac{1}{5}$ і $\widetilde{B} = \frac{1}{5} + \frac{0,8}{10} + \frac{0,4}{15} + \frac{0,2}{20}$. Визначити значення змінної *у* на виході, якщо $x = \widetilde{C} = \frac{0,3}{1} + \frac{0,5}{2} + \frac{1}{3} + \frac{0,7}{4} + \frac{0,4}{5}$.

Розв'язання. Спочатку визначимо нечітке відношення $\widetilde{R}$ між множинами, що відповідає заданому правилу, використавши означення 5.2.6, 5.2.7, та представимо його матрицею (Табл. 5.5.1).

Таблиця 5.5.1.

$\widetilde{R}$	5	10	15	20
1	0	0	0	0
2	0,1	0,1	0,1	0,1
3	0,5	0,5	0,4	0,2
4	0,8	0,8	0,4	0,2
5	1	0,8	0,4	0,2

Тепер за формулою $y = \widetilde{C} \circ \widetilde{R}$ визначаємо нечітке значення *у* - $\widetilde{Y} = \frac{0,7}{5} + \frac{0,7}{10} + \frac{0,4}{15} + \frac{0,2}{20}$.

- Запитання для самоперевірки:

Запитання для самоперевірки:

1. Що таке нечітка логіка?

2. Що є математичною базою для нечітких множин?
3. Яких значень може набувати характеристична функція належності у впорядкованій множині $M = \{0,1\}$?
4. Сформулюйте основні характеристики логіки нечітких множин.
5. Які є методи побудови функції належності нечітких множин?
6. Сформулюйте основні операції, які виконують над нечіткими множинами.
7. Яким чином будується правило нечіткого виводу.
8. Що таке нечітка і лінгвістична змінні?
9. Де можуть застосовуватися нечіткі й лінгвістичні змінні?

Задачі для самостійного розв'язування

1. Визначити графічно функцію належності елемента *x*, який набуває значень від 0 до 15, до підмножини $A = \{ 6,7,8,9,10 \}$.

2.Нормалізувати нечітку множину $\tilde{X} = \frac{0,1}{1} + \frac{0,4}{3} + \frac{0,8}{5}$.

3. Нехай універсум – множина натуральних чисел. Записати за допомогою нечіткої множини поняття натуральних чисел "близьких до числа 80".

4. Універсумом нечіткої множини $\tilde{X} = \frac{0,1}{2} + \frac{0,8}{4} + \frac{0,5}{6}$ є множина $U=\{2, 3, 4, 5, 6\}$. Визначити характеристики нечіткої множини.

5. Заданий універсам U ={2, 3, 4, 5, 6, 7, 8}, на якому визначені нечіткі множини $\tilde{X} = \frac{0,3}{2} + \frac{1}{4} + \frac{0,5}{6} + \frac{0,5}{8}$ та $\tilde{Y} = \frac{0,6}{3} + \frac{1}{5} + \frac{0,2}{6} + \frac{0,3}{7} + \frac{0,6}{8}$. Знайти множини $\tilde{X} \cup \tilde{Y}, \tilde{X} \cap \tilde{Y}, \tilde{X} \times \tilde{Y}$.

Коментарі. Основні відомості щодо поняття нечіткої логіки які викладені у цьому розділі, випливають з [19,20,25], характеристичні функції належності, їх побудова і наочне подання взяті із [20,22], виконання операцій над нечіткими множинами базуються на [25], а нечіткі вислови взяті із [19,20,25].

Основна частина прикладів взята з Інтернет - видання : http://matlab.exponenta.ru/fuzzylogic

Тема 6

- Модальна логіка
  - 6.1. Особливості побудови модальних систем
  - 6.2. Семантика Кріпке
  - 6.3. Теорія "двійників" К. Льюїса

Модальна логіка

Термін "модальна логіка" виник в зв'язку з прагненням збагатити мову класичної логіки та розширити її можливості, додаючи в неї елементи, схожі на деякі часто вживані словесні конструкції. В цілому **модальною логікою** називають логічні системи, в яких використовуються оператори, подібні за змістом з модальними дієсловами природної мови: "можна", "потрібно", "можливо", "неможливо", "необхідно", "обов'язково", "дозволено", "заборонено", "впевнений", "було", "буде" та інші. В вузькому змісті, модальною логікою є логіка, в якій використовуються два основні оператори: $[\Box ?]$ - "обов'язково" ("необхідно") й $[\diamondsuit ?]$ - "можливо". Проте термін "модальна логіка" використовується більш широко, щоб покрити сімейство логік з аналогічними правилами. В принципі число груп модальних понять не обмежено, але модальні поняття різних типів мають загальні формальні властивості й визначаються друг через друга по одній і тій же схемі. Наприклад, основні оператори пов'язані між собою формулами:

$$[?][?][\Box ?]\ A = [\urcorner ?]\ [\diamondsuit ?]\ [\urcorner ?]\ A, \quad [\diamondsuit ?]\ [?][?] A = [\urcorner ?]\ [\Box ?]\ [\urcorner ?]\ A \qquad (6.1)$$

І цей взаємозв'язок цілком узгоджується зі звичайними уявленнями. Наприклад, фраза "Необхідно, що буде сьогодні дощ" (ліва частина першої формули) відповідає фразі "Не можливо, що не буде сьогодні дощу" (права частина першої формули).

Перші дослідження в модальній логіці відносять до часів Аристотеля, й пов'язують з його ім'ям, та його учнів. До засновників сучасної модальної логіки відносять К. І. Льюїса та Я. Лукасевича, які в 1910-1932 роках ввели в сучасну логіку термін модальність. Третій період в становленні модальної логіки розпочато роботами С. А. Кріпке (кінець 1950-х початок 60-х років) та продовжено дослідженнями К. Я. Ю. Хінтікка та Д. Льюїса.

Модальні логічні числення можна побудувати на основі числень класичної логіки (висловлювань та предикатів), шляхом розширення їх додатковими операторами, що характеризують ту чи іншу модальність. Наприклад, якщо оцінка того, що стверджується в висловлюваннях дається з позицій законів науки, то використовуються модальні оператори "необхідно", "можливо", якщо - з позицій пізнання, то - модальні оператори "доведено", " спростовано", якщо - з позицій норм права, то – "обов'язково", "дозволено", "заборонено", якщо - з позицій часу – "раніше","пізніше" та інші.

Логіки, сформовані на підставі того або іншого типу модальності, мають назви : "алетична логіка", "деонтична логіка", "епістемічна логіка", "часова логіка" та ін. (термін "логіка" використовують у цьому контексті як "теорія", "формально-логічна система").

Кожен тип модальності визначають з погляду об'єктивних (фактичних) і логічних детермінантів. Алетична модальність, зокрема, визначається з погляду об'єктивних законів природи, суспільства й логічних законів, деонтична модальність - з погляду прийнятних у суспільстві правових і моральних норм, епістемічна модальність - з погляду закономірностей пізнання об'єктивного світу, часова модальність - з погляду часових характеристик того, що відбувається в світі.

6.1. Особливості побудови модальних систем

Побудова формальної теорії в модальній логіці заснована на тих же засадах, що визначені для побудови будь-яких формальних аксіоматичних теорій (див. розділ 2.1) і включає в себе : а) введення алфавіту, що складається з пропозиційних символів , символів логічних та модальних зв'язків, символів дужок; б) визначення які слова з алфавіту є формулами; в) введення аксіом, як деякої підмножини формул; г) задання скінченої множини відношень між формулами, що називаються правилами виводу.

Означення 6.1.1. Модальною системою *називається формально-логічна система (модель, теорія, числення модальних висловлювань), створена на підставі певного типу модальності засобами особливої мови на загальних принципах побудови логік.*

У модальних системах виокремлюють немодальну частину та суто модальну частину. Немодальну частину представляє зазвичай класична математична логіка, а в модальній частині вводяться певного типу модальності та встановлюються логічні відношення між висловлюваннями з введеним типом модальності. Розрізняють семантику та синтаксис модальної логіки.

У семантичному аспекті визначають структуру модальних висловлювань на змістовному рівні. Це дає змогу виявити властивості висловлювань із певним типом модальності й виокремити онтологічні та логічні модальності, ввести терміни, що виокремлюють сферу міркувань з модальними висловлюваннями - висловлювання, властивість, відношення, терміни, які виокремлюють вид модальності - алетична, деонтична, епістемічна, часова та істинніснe значення висловлювання.

У синтаксичному аспекті структуру модальних висловлювань визначають абстрактно від їх змісту і формалізують засобами штучно створеної мови, на підставі чого здійснюють логічні операції над символами, що зображають логічні відношення між модальностями (числення модальностей).

Приклад 6.1.1. *Наведемо опис модальної системи S1 К.І.Льюїса, побудованої за наведеними вище правилами.*

Розв'язання. *Вводимо алфавіт S*, до якого входять : 1) p, q, r, ... - символи для висловлювань; 2) $[\urcorner ?]$, $\vee$, $\wedge$ - символи логічних зв'язків; 3) $[\Box ?]$,$[\diamondsuit ?]$ - символи модальних понять; 4), (- дужки.

Визначаємо, що є формулами : 1) кожний з символів p, q, r, ... є формулою; 2) якщо A та B є формулами, то наступні вирази теж є формулами:

¬ A – заперечення A; A $\vee$ B – диз'юнкція A і B; A $\wedge$ B – кон'юнкція A і B;

$[\Box ?]$ A – обов'язково A; $[\diamondsuit ?]$ A – можливо A; A→B = ¬ A $\vee$ B –

матеріальна імплікація; $A \prec B = [\urcorner?]\ [\diamondsuit?]\ (A \wedge ([\urcorner?]\ B))$ – строга імплікація Льюїса; $A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$ – матеріальна еквівалентність;$A \div B = (A \prec B) \wedge (B \prec A)$ – строга еквівалентність.

Вводимо аксіоми : A1) $p \wedge q \prec q \wedge p$; A2) $p \wedge q \prec p$; A3) $p \prec p \wedge p$; A4) $(p \wedge q) \wedge r \prec p \wedge (q \wedge r)$; A5) $p \prec [\urcorner?]\ ([\urcorner?]\ p)$; A6) $(p \prec q) \wedge (q \prec p) \prec (p \prec q$; A7) $p \wedge (p \prec q) \prec q$

Визначаємо правила виводу :

P1. Правило заміни строго еквівалентним;

P2. Будь-які змінні p, q, r, … можна замінити довільними формулами;

P3. Введення кон'юнкції : із A, B виводиться A $\wedge$ B;

P4. *Modus ponens* зі строгою імплікацією: із A і A $\prec$ B виводиться B.

Систему S1 побудовано.

Якщо додати до аксіом системи S1 аксіому A8) $[\diamondsuit?]\ (p \wedge q) \prec [\diamondsuit?]\ p$, то отримаємо систему S2. Взагалі Льюїсом були запропоновані й інші системи модальної логіки, наприклад S3-S5 й інші.

Відмітимо введення К. І. Льюїсом логіку поняття "строга іплікація" задля усунення парадоксів матеріальної іплікації. Відомо, що в матеріальній імплікації висловлювання A→B істинно в незалежності від B, якщо A - хибно. Чому, наприклад, з того, що 2x2=5 слідує, що Петров – студент? В строгій іплікації це вже виправлено.

Але не позбавлена парадоксів й строга імплікація. Для їх виключення Ф. В. Аккерман побудував свою систему модальної логіки. Серед відомих систем існують також системи Лукасевича та інші.

По відношенню до систем модальної логіки виникають ті ж проблеми, що мають місце в класичній логіці: проблема вирішення, проблема повноти та інші. Для систем Льюїса проблема вирішення розв'язується за допомогою алгебраїчних методів. С. А. Кріпке довів, що в численні одномісних предикатів з доданими модальними операторами проблема вирішення нерозв'язна. Цей результат дуже важливий, тому що він вказує на істотну відмінність модальної логіки від класичної математичної логіки. Доведена також неповнота числень Льюїса в тому сенсі, що в них виводима не всяка істинна формула.

Основною проблемою модальної логіки є те, що модальні оператори не являються істиннісне-функціональними, тобто істинність формули не є функцією от істинності її складових. Формула "$[\diamondsuit?]$ p" може приймати значення істини, навіть якщо p є хибне, а формула "$[\Box?]$ p" буде хибною, навіть якщо p є істинне. Для класичної логіки предикатів формальна семантика була створена, в той час як для модальної логіки її не існувало. Роботи Сола Кріпке дещо розв'язали ці проблеми й ввели нове поняття в модальну логіку – поняття можливого світу. Введення самого цього поняття в наукові праці приписують

Г. В. Лейбніцу, деякі джерела вказують, що поняття можливого світу в наукових дослідженнях з'явилося ще задовго до Лейбніца й повязують з філософом і теологом середньовіччя Й. Д. Скотом.

6.2. Семантика Кріпке

Серед відомих семантичних інтерпретацій модальних логік щонайчастіше застосовується семантика С. Кріпке (також відома як реляційна семантика або семантика фреймів), створена в кінці 1950-х і початку 1960-х років.

У моделі Кріпке задається множина можливих "світів" (станів), в яких може перебувати об'єкт, а також відношення досяжності на цій множині. При цьому модальний оператор "необхідно" інтерпретується як "істинно у всіх можливих досяжних світах", а оператор "можливо" - як "істинно хоча б в одному з можливих досяжних світів".

Означення 6.1.1. Фрейм Кріпке F або модальний фрейм представляє собою пару $\langle W, R \rangle$, де W є непорожньою множиною, а R є бінарним відношенням на W ($R \subset W \times W$). Елементи множини W називають вузлами або можливими світами, а R - відношенням досяжності.

Означення 6.1.2. Моделлю Кріпке M називається пара $\langle F, [\models?] \rangle$, де F - фрейм Кріпке, $\models$ - є відношення між вузлами W і модальними формулами, такі, що:

- $w \models \neg A$ тоді і тільки тоді $w \not\models A$;
- $w \models (A \wedge B)$ тоді і тільки тоді $w \models A$ і $w \models B$;
- $w \models (A \vee B)$ тоді і тільки тоді $w \models A$ або $w \models B$;
- $w \models (A \rightarrow B)$ тоді і тільки тоді $w \not\models A$ або $w \models B$;
- $w \models [\Box?]\ A$ тоді і тільки тоді $\forall u \in W$ якщо $w R u$, то $u \models A$;
- $w \models [\diamondsuit?]\ A$ тоді і тільки тоді $\exists u \in W$ виконується $w R u$ і $u \models A$.

Позначення $w \models A$ читаються так: " w задовольняє A", " A виконується в w".

У відповідності з цією семантикою, висловлювання *p* є необхідно-істинним по відношенню до можливого світу w $\in$ W, якщо воно істинне в кожному світі, досяжному для w, і *p* є можливо-істинним, якщо воно істинне в якомусь світі, досяжному для w. Можливість таким чином, залежить від відношення досяжності R, яке дозволяє висловити відносний характер можливості.

Введене в модель відношення досяжності R дозволяє вести класифікацію модальних логік , на предмет які властивості цих відношень підтримуються в тій чи іншій моделі. Наприклад, система модальної логіки K – не підтримує жодної властивості R; T – логіка з R, що має властивість рефлексивності; S4 – підтримує рефлексивність та транзитивність R; S5 - підтримує еквівалентність R.

Формально $\models$ можна уявити як деяку функцію □ (щонайчастіше її називають оцінкою), що приписує кожній формулі значення істинності по відношенню до кожного світу w□W. Підсумком досліджень С. Кріпке стало те, що формули в модальній логіці будуть приймати значення істинності по відношенню до можливих світів.

Приклад 6.2.1. Представимо, що маємо висловлення *p* – "Лейбніц займався математикою". Насправді, Г. В. Лейбніц займався математикою, тому *p є істинним по відношенню до "нашого" актуального світу* $w_1 \in W$. *В даній моделі φ(p, w1) – "Істина".*

Цілком можливо, що Лейбніц не займався би математикою, а був би ювеліром. Отже, *існує можливий світ w2* $\in$ *W, в якому p та φ(p, w2) приймають значення "Хибність".*

Головна ідея такого аналізу в тому, що дозволяючи значенню істинності формул змінюватись від світу до світу, ми зможемо визначити їх значення істинності так, як це раніш було неможливим.

У логіці висловлювань атомарні висловлювання набувають значення істинності (I або X), потім значення складних пропозицій (формул) розраховуються за таблицями істинності. У модальній логіці при введені семантики Кріпке спочатку вводиться набір W можливих світів. Потім надається значення істинності кожному висловлюванню для кожного з можливих світів в W. Це означає, що присвоєне висловлюванню *p* значення істинності для світу $w_1 \in W$ може відрізнятися від значення, присвоєного для іншого світу $w_2 \in W$. Значення істинності атомарних *p* в світі *w* можна записати наступне *φ(p, w)*. Враховуючи ці позначення, значення істинності (*I* для істинних, *X* для хибних) складних речень (формул) модальної логіки визначається наступне:

- *φ(¬ A, w) = I* тоді і тільки тоді *φ(A, w) = X;*
- *φ(A→B, w) = I* тоді і тільки тоді *φ(A, w) = X* або *φ(B, w) = I;*
- *φ(* $[\Box?]$ *A, w) = I* тоді і тільки тоді $\forall u \in W$, *φ(A, u) = I;*

й тому подібне.

6.3. Теорія "двійників" К. Льюїса

Теорія "двійників" – альтернативна семантиці Кріпке інтерпретація модальної логіки з кванторами, запропонована Д. Льюїсом в 1968 році. Ідея полягала в тому, що модальні оператори не потрібні для квантифікації для можливих світів, тобто можна позбавитися від модальних операторів перетворивши звичайну модальну логіку в теорію "двійників".

"Нова" теорія заснована на предикатах:

- *Wx (x* є можливий світ);
- *Ixy* (*p* знаходиться в можливому світі *y*);
- *Ap* (*p* є актуальним індивідом);
- *Cqy* (*q* є двійником *p*).

Постулати теорії "двійників" наступні:

П1. Немає нічого за межами того, що є у можливому світі, можливі світи замкнуті;

П2. Жодного індивіда не існує одночасно у двох можливих світах;

П3. Чим би не був двійник, він знаходиться в якому-небудь можливому світі;

П4. Те, у чого є двійник, знаходиться у можливому світі;

П5. Ні у чого в можливому світі немає двійника в тому ж самому можливому світі, тобто жодний об'єкт в можливому світі неможна назвати двійником чого-небудь в тому ж можливому світі;

П6. Будь-який об'єкт в можливому світі є власним двійником, єдиним двійником об'єкту в тому ж самому світі буде він саме;

П7. Деякі можливі світи містять всі актуальні речі;

П8. Щось є актуальне, тобто існує актуальний світ.

Приклад 6.3.1. *Записати речення р звичайної мови "Могли б існувати літаючі люди" (що є аналогом питання "Чому люди не літають?") за допомогою модальних операторів треба перевести його в мову теорії "двійників".*

Розв'язання. Спочатку переведемо вхідне речення *р* на мову, що використовує модальні оператори: *р - "Можливо існує людина, що може літати", що в модальній логіці представимо формулою*

$$\exists\, w \exists\, x\big(H(x) \wedge M(x)\big). \qquad (6.3.1)$$

В формулі (6.3.1) *H(x)* та *M(x)* є предикатами, побудованими на множині людей (Представте зміст предикатів самостійно).

В теорії "двійників" формула (6.3.1) набуде виду

$$\exists\, w \exists\, x \ (w \text{ є можливий світ} \ \wedge \ x \text{ знаходиться в } w \ \wedge \ (H(x) \ \wedge \ M(x)). \ (6.3.2)$$

Теорія "двійників" К. Льюїса є найбільш закінченою спробою звести поняття модальної логіки ($[\Box?]$,$[\diamondsuit?]$) до відношень подібності між конкретними сутностями, об'єктами в рамках понятійної моделі можливих світів.

Існують й інші семантики, що розглядають поняття модальної логіки з інших точок зору. Цікавою в цьому плані є теоретико-ігрова семантика

К. Я. Ю. Хінтікка, в основі якої знаходяться, з одного боку, математична теорія ігор, з іншого – теоретико-модальна семантика.

- Запитання для самоперевірки:

Запитання для самоперевірки:

1. Що таке модальність?
2. Що називають модальним висловлюванням?
3. Яку логіку називають модальною?
4. Чим відрізняється модальна логіка від класичної?
5. Що називається модальною системою?
6. На яких засадах будується модальна система? Наведіть приклад.
7. Які основні модальні оператори ті який зв'язок вони мають між собою?
8. Опишіть семантику Кріпке.
9. Опишіть теорію "двійників" К. Льюїса.

Задачі для самостійного розв'язування.

1.Нехай W = {1, 2, 3, 4, 5}, R = {(1,1), (1,2), (2,3), (1,5), (5,4), (4,4), (4,3)}. Представте фрейм Кріпке F= $\langle$ W, R $\rangle$, в графічному виді.

2. Розглянемо модель Кріпке M= $\langle$ F, $\models$ $\rangle$, де F – фрейм попереднього завдання, а відношення $\models$ задається для висловлювань p та q оцінками φ(p, W)= {1, 2, 5}, що означає – висловлювання істинне в світах 1, 2, 5 та φ(q, W) = {1, 3, 4}.

Перевірте твердження: 1) $[\Box?]$ p істинне в 1,3 хибне в 4; 2) $[\diamondsuit?]$ $q \wedge$ $[\diamondsuit?]$ $[\urcorner?]$ q істинне в 1.

3. Запишіть речення "Мій брат міг би бути філософом" (в дійсності він не є їм) на мові модальної логіки та застосуйте мову теорії "двійників".

Коментарі. Основні свідчення щодо поняття модальної логіки представлені за джерелами [15,25]. Розділи 6.2 та 6.3 опираються на статтю А.А. Веретеникова "Философия модальности: аналитическая философия и логика"з журналу "История философии", №13, 2008р.: М.: ИФРАН. - с. 26-48.

Тема 7

- Епістемічна логіка
  - 7.1. Основні визначення
  - 7.2. Оператори знання й переконання, їх властивості
  - 7.3. Формальна мова епістемічної логіки

Епістемічна логіка

7.1. Основні визначення

Епістемічна логіка – це розділ модальної логіки, яка досліджує логічні зв'язки висловів, що містить такі поняття, як *"вважає" , "сумнівається","відкидає","знає","доказово"," нерозв'язно", "спростовно"* і тощо**.** Назва логіки походить від грецького слова " *episteme*" – знання . Є два варіанти епістемічної логіки: **логіка знання** і **логіка переконання.**

Означення 7.1.1. Епістемічною логікою знання називають логіку, яка займається вивченням міркувань , вихідним терміном яких є " *доказово", "істинно"* і т. п.

Одна з перших логік знання була сформульована австрійським математиком і логіком К. Геделем. До її законів входять такі положення:

1. Якщо висловлювання доказове, воно істинне (довести можна тільки істину, доказів брехні не існує).
2. Логічні наслідки доказового також є доказовими.
3. Якщо щось доказове, то воно доказове, але логічна суперечність недоказова і т. ін.

Іншим прикладом логіки знання може служити логіка істини, що встановлює такі закони:

1.Якщо висловлювання істинне, то не так, що його заперечення також істинне, наприклад, *"якщо істинно, що Земля обертається, то не так, що правдиво, ніби вона не обертається".*

2. Кон'юнкція істинна, якщо і тільки якщо обидва висловлювання, що входять до неї, істинні, наприклад, " *істинно, що холодно і йде сніг, тільки якщо істинно, що холодно, і правдиво, що йде сніг",* тощо.

Означення 7.1.2. Епістемічною логікою переконання називають логіку, яка займається вивченням міркувань, вихідним терміном яких є *"вважає" , "сумнівається","відкидає"* і т. п.

У логіці переконань як початковий зазвичай беруть поняття *"вважає" ("переконаний", "вірить"),* через нього визначаються поняття *"сумнівається"* і *"відкидає"*: суб'єкт сумнівається в чомусь, якщо тільки він не переконаний ні в цьому, ні в протилежному; суб'єкт відкидає щось, якщо тільки він переконаний у протилежному.

Серед законів логіки переконань є положення: суб'єкт вважає, що є перше і друге, якщо і тільки якщо він вважає, що є перше, і вважає, що є друге. Наприклад, " *суб'єкт вірить, що Марс – планета і що Місяць – планета, тільки якщо він вірить, що Марс – планета, і вірить, що Місяць – планета".*

Не можна одночасно вірити і сумніватися, бути переконаним і відкидати, сумніватися і відкидати. Суб'єкт або переконаний, що справа йде так ось, або сумнівається в цьому, або відкидає це, наприклад, " *суб'єкт або переконаний, що Венера – зірка, або сумнівається в цьому, або відкидає це".* Неможливо бути переконаним одночасно в цьому і в протилежному, наприклад, *" не можна вірити як у те, що астрологія наука, так і в те, що вона не є наукою",* і т. ін.

Для понять *"знає"," істинно", "доказово"* правильно, що логічні наслідки відомого також відомі, істинного — істинні, доказового — доказові. Аналогічний принцип для поняття *"переконаний",* що здається протиінтуїтивним, отримав назву **парадоксу логічного всезнання**. Він стверджує, що людина переконана у всіх логічних наслідках, що випливають з положень, які він приймає. Наприклад, якщо людина впевнена в п'яти постулатах геометрії Евкліда, то це означає, що вона приймає і всю цю геометрію, оскільки остання випливає з них. Однак це не так. Погоджуючись із постулатами, людина може не знати доведення теореми Піфагора, а відтак сумніватися в тому, що вона правильна.

7.2. Оператори знання й переконання, їх властивості

Згідно з епістемічною логікою, знання відрізняється від переконання. Цій відмінності відповідає відмінність між двома варіантами епістемічної логіки - логікою знання і логікою переконання. Тому інтерпретація понять знання і переконання може бути як особливий випадок застосування епістемічних модальних операторів, які додаються до мови звичайної класичної логіки. У подальшому викладі, щоб уникнути зайвої технічної деталізації, ми використовуватимемо епістемічні оператори без явного посилання на конкретні суб'єкти пізнання при цьому, завжди неявно матимемо на увазі наявність деякого фіксованого суб'єкта. Наприклад, оператор знання *Kp* означає – "*хтось знає, що p*" або просто "*p відоме*" , а оператор переконання *Bp* означає – "*хтось вважає, що p*". Іноді разом з операторами знання й переконання вводяться і інші аналогічні епістемічні оператори, наприклад , "*сумнівається*", "*спростовує*" і т. ін.

Апарат епістемічної логіки дозволяє ставити й успішно вирішувати завдання виявлення формальних (логічних) властивостей операторів знання і переконання (а отже і відповідних понять), формулювання аксіом, що виражають ці властивості, і встановлення взаємозв'язку між даними операторами та поняттями. При цьому активно задіються результати філософського аналізу понять знання і переконання. Почнемо з оператора переконання. Для цього оператора, додатково до аксіом класичної логіки, можна взяти такі постулати:

B1. (p ® q) ® (Bp ® Bq) . (Кожен має бути переконаний в істинності всіх наслідків допущень, що приймаються ним).

B2. Bp ® ¬ B ¬ p . (Неможливо одночасно бути переконаним в істинності будь-якого вислову і його заперечення – раціональний суб'єкт не повинен приймати суперечності).

B3. Bp ® BBp . (Якщо хтось вважає, що ***p***, то він також переконаний у тому, що він так вважає).

B4. ¬ Bp ® B ¬ Bp . (Якщо хтось не вважає, що ***p***, то він має бути переконаний у тому, що він так не вважає), де символ ® позначає множину рішень.

Перші два постулати твердять про те, що ми маємо справу не з дескриптивним, а таким, що з раціоналізувало поняттям переконання. Це поняття виражає не фактичні переконання того або іншого конкретного суб'єкта в тому або іншому конкретному випадку, а принципи, яким повинні підлягати раціональні переконання взагалі. Останні два постулати виражають ту обставину, що ми не можемо помилятися стосовно того, в чому ми переконані, а в чому – ні. Суб'єкт завжди має визначеність щодо висловів про власні переконання.

Перейдемо тепер до оператора знання. Для цього оператора зазвичай беруть такі основоположні постулати:

K1. Kp ® p . (Якщо вислів відомий, то він істинний; знання вислову спричиняє його істинність).

K 2. K(p ® q)® (Kp ® Kq). (Якщо відомо, що вислів ***p*** спричиняє вислів ***q***, а також відоме ***p***, то відоме і ***q***).

K3. Kp ® KKp. (Якщо хтось знає якийсь вислів, то він також знає, що він це знає).

У багатьох системах епістемічної логіки діє таке правило виведення, якому повинен підлягати оператор знання. Якщо вислів ***p*** є доведеним, то доведеним є і вислів ***Kp*** (правило "*навішування*" оператора знання). Згідно з цим правилом, суб'єкт, що пізнає, знає всі теореми логіки (логічне всезнання). Наступним важливим завданням є встановлення взаємозв'язку між операторами знання і переконань. Цей взаємозв'язок, в основному, фіксується за допомогою такого постулату:

KB1. Kp ® Bp. (Якщо хтось знає, що ***p***, то він також вважає, що ***p***).

Постулати ***K1*** і ***KB1*** відображають те розуміння, що необхідними умовами знання вислову є як його істинність, так і переконаність у ньому з боку деякого суб'єкта. У деяких системах епістемічної логіки ці умови вважаються також і достатніми, внаслідок чого отримуємо наступне визначення знання:

Визначення 1. Kp U Bp U p. (Хтось знає, що ***p***, якщо і тільки якщо він переконаний, що ***p*** і ***p*** є істинним).

Незважаючи на те, що з філософської точки зору це визначення є явно неповним, його цілком можна використовувати для цілей логічного аналізу як робоче визначення. Якщо ж увести додатково оператор "*обґрунтованості*" – ***Jp***, який читається як "***p є обґрунтованим***", то можемо сформулювати таке визначення знання як обґрунтованого дійсного переконання:

Визначення 2. ***Kp U Bp U Jp U p***. (Хтось знає, що ***p***, якщо і тільки якщо він переконаний, що ***p*** і ***p*** є істинним, і обґрунтованим).

Перелічені постулати роблять можливим формальний аналіз понять знання і переконання в рамках певної системи аксіом. Для опису та вивчення ситуацій, при яких суб'єкти, знаходячись у загальних умовах, перетворюють свої знання, розроблена формальна мова епістемічної логіки.

7.3. Формальна мова епістемічної логіки

Світ, у якому знаходяться агенти (носії знання), описується його властивостями, наприклад, "*сніг білий*" – властивість нашого світу. Для позначення різних властивостей різних агентів використовуються пропозиціональні літери мови $p_1, p_2, \cdots, p_k, \cdots$, яку будемо позначати як $\varphi = \{p_1, p_2, \ldots\}$.

Цей світ, у якому знаходяться агенти, що моделюються, позначимо такою структурою Кріпке:
$(S, \pi, K_1, K_2, \ldots, K_n)$,
де S – множина можливих станів світу;
π – описує кожен можливий стан $s \in S$;
$K_1, K_2, \ldots, K_n$ – відношення на S, які описують знання агентів;
$\pi(s) : \varphi \to \{$ істинно, хибно $\}$.

1. Якщо $(s_1, s_2) \in K_i$, то агент i, знаходячись у стані s_1, буде розглядувати стан s_2 можливим. Необхідно відмітити, що розглядувані агенти, як правило, не знають реального стану світу, тобто стану, в якому вони знаходяться .
2. **Приклад 7.3.1** Нехай у пісочниці граються 3 дитини. Станом світу для них є інформація про те, що хто із них брудний, а хто чистий. Необхідно графічно побудувати структуру Кріпке можливого стану цих дітей.
3. *Розв'язання.* Стани дітей будемо позначати набором $a_1a_2a_3$, де $a_i \in \{0,1\}$. Наприклад, 010 означає, що стан обличчя другої дитини брудний, а в решти – чистий. У цьому випадку можна обмежитися трьома пропозиціональними літерами p_1, p_2, p_3, де $p_i = 1$ означає "*в i-ї дитини обличчя брудне*". Виходячи із цього графічна структура Кріпке матиме вигляд, наведений на рис. 7.3.1.

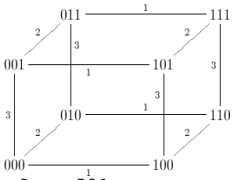


Рисунок 7.3.1

На рис. 7.3.1 ребра позначають дітей, а вершини – їхній стан. Наприклад, для другої дитини можливі стосунки на множині можливих станів матимуть такий вигляд: K_2 = {(000,010), (010,000), (100,110), (110,100), (101, 111), (111, 101), (001, 011), (011, 001), (000, 000), (010, 010), (110, 110), (100, 100), (001, 001), (011, 011), (111, 111), (101, 101)}, де пара (000, 010) означає, що якщо всі діти чисті, то друга дитина – брудна, оскільки вона не бачить свого обличчя , все ж таки розглядатиме стан 010 можливим. Описання кожного можливого стану в структурі Кріпке матиме такий вигляд: $\pi(001)(p_3)$ = *істинно*, $\pi(001)(p_1)$ = *хибно* і так далі.

- Запитання для самоперевірки:

Запитання для самоперевірки:

1. Що називають епістемічною логікою?
2. Які можуть бути епістемічні логіки?
3. Чим відрізняється епістемічна логіка знання від епістемічної логіки переконання?
4. Які є постулати в операторі переконання?
5. Які є постулати в операторі знання?
6. Сформулюйте формальну мову епістемічної логіки.
7. Які є властивості операторів знання і переконання?

Коментарі. Основні відомості щодо поняття епістемічної логіки, визначення різниці між епістемічною логікою знання та епістемічною логікою переконання, мова епістемічної логіки, властивості операторів її знання і переконання наведені в [15].

материалы в разработке

Тема 8

- Некласична математична логіка
- Деонтична логіка
  - 8.1. Основні визначення
  - 8.2. Синтаксис та семантика деонтичної логіки

Некласична математична логіка

Деонтична логіка

8.1. Основні визначення

Деонтична логіка відноситься до одного з розділів модальної логіки, в якому досліджуються нормативні висловлювання та їх відношення в структурі міркувань. Сама назва цієї логіки походить від грецького слова "*deon*", що означає те, що є необхідним або належним.

Деонтична логіка аналізує міркування, посилками або висновками яких є висловлювання про норми (правила). Вона відрізняє правильні (обґрунтовані) міркування від неправильних і таким чином створює теорію нормативних висновків. Такі логічні дослідження норм і правил важливі для математики, логіки, лінгвістики, етики, юриспруденції, соціології та економіки.

Будучи однією з гілок модальної логіки, деонтична логіка розвивалась в її складі разом і невід'ємно була під її впливом з античних часів до XX століття, перша спроба відокремити деонтичну логіку в єдину самостійну систему належить австрійському логіку Е. Маллі (1926 р.). Подальший її самостійний розвиток пов'язаний з фінським філософом та логіком Г. Х. фон Врігтом (1951 р.), що побудував нову систему деонтичної логіки, яка була розвинута в працях А. Андерсона, О. Івіна та інших. Пізніше виявилось, що деонтичній логіці можна надати семантику Кріпке, і фон Врігт приєднався до цього напрямку.

Одним із ключових понять деонтичної логіки є поняття **деонтичної модальності** (нормативної модальності). Деонтична модальність характеризує практичну дію (вчинок) з точки зору певної системи норм і належить до таких понять, як "допустимо" (дозволено), "неприпустимо" (заборонено), "обов'язково", "не обов'язково", "необхідно", "повинно" (слід, треба), "байдужо", "добре", "погано" та інші. Треба відзначити, що деякі з цих понять отримали більше уваги в деонтичній логіці ніж інші і в сучасній деонтичній логіці розглядаються нормативні висловлювання, в які входять модальності:

"*необхідно*" (обов'язково) – позначається деонтичним оператором *O*;
"*дозволено*" - позначається деонтичним оператором *P;*
"*заборонено*" - позначається деонтичним оператором *F;*
"*байдужо*" - позначається деонтичним оператором *I*.

Означення 8.1.1. Деонтичною логікою називають логіку, що досліджує логічні структури і логічні зв'язки нормативних висловлювань.

Нормативні поняття " *необхідно*","*дозволено*" багато в чому схожі на модальні поняття "*обов'язково*","*можливо*"($\Box$,$\diamondsuit$). Власне, нормативні поняття самі по собі є модальними поняттями. Іншими словами, деонтична логіка – це модальна логіка, в якій замість унарних операторів модальності $\Box$ та $\diamondsuit$ вводяться унарні оператори деонтичних модальностей - необхідності і дозволеності (**O і P**).

Деонтична логіка не є етичною теорією, що стверджує , що насправді є дозволеним, забороненим чи необхідним, але, по суті, являється частиною такої теорії. Точно так, як існують різного роду етичні теорії, можуть існувати й різні деонтичні логіки. Вивчення різних ідентичних логік є частиною того, що можна назвати мета-етикою.

8.2. Синтаксис та семантика деонтичної логіки

Деонтичну логіку створюють на загальних принципах побудови сучасних некласичних логік. Розрізняють семантику і синтаксис модальної системи з деонтичними модальностями.

В синтаксичному аспекті деонтичну логіку можна побудувати за правилами побудови модальних логік (див. розділ 6), *ввівши алфавіт S*, до якого входять p, q, r, ... - символи для позначення деонтичних висловлювань; $\ulcorner$,$\vee$,$\wedge$,$\rightarrow$,$\leftrightarrow$ - символи логічних зв'язків; *O, P, F, I* - символи деонтичних понять;), (- дужки.

Потім визначаємо, *що є формулами в деонтичній логіці* :

кожний з символів p, q, r, ... є формулою; якщо A та B є формулами, то наступні вирази теж є формулами: ¬ A – заперечення A; A $\vee$ B – диз'юнкція A і B; A $\wedge$ B – кон'юнкція A і B; *O*A – необхідно,щоб A; *P*A – дозволено,щоб A; *F*A – заборонено, щоб A; *I*A – байдуже, щоб A; A→B - імплікація; A $\leftrightarrow$ B - еквівалентність.

Вводимо аксіоми :

A1) *O*A $\leftrightarrow$ *P*A (A необхідно еквівалентно тому, що A дозволено);
A2) *F*A $\leftrightarrow$ ¬ *P*A (A заборонено еквівалентно тому, що A не дозволено);
A3) *F*A $\leftrightarrow$ *O*¬A (A заборонено еквівалентно тому, що необхідно не A);
A4) *P*A $\leftrightarrow$ ¬*O*¬A (A дозволено еквівалентно тому, що не A не необхідно);
A5) ¬(*O*A $\wedge$ *F*A) є хибно (A не може бути одночасно необхідним і забороненим).

Визначаємо правила виводу : П1. Правило заміни строго еквівалентним;
П2. Будь-які змінні p, q, r, ... можна замінити довільними формулами;
П3. Введення кон'юнкції : із A, B виводиться A $\wedge$ B;

$$\frac{A, A \rightarrow B}{B} \qquad \frac{A}{\Box A}$$

П4. *Modus ponens*: із A і A→B виводиться B ( ). П5. Із A виводиться *O*A ( ).

Таким чином побудована деонтична логіка має назву стандартної деонтичної логіки (SDL). Інший набір аксіом може привести до деонтичної логіки іншого типу. Побудована навіть деяка упорядкованість деонтичних логік, в залежності від набору саме аксіом. Наприклад, зустрічається така впорядкованість – DKr $\subseteq$ D $\subseteq$ DM $\subseteq$ DS4 $\subseteq$ DS4.2 $\subseteq$ DS4.3 $\subseteq$ DS5.

Семантика деонтичної логіки вміщує категорії:

- висловлювання, властивість, відношення;
- терміни, які позначають деонтичні модальності, - "необхідно", "дозволено", "заборонено", "байдуже";
- терміни на позначення сфери дії соціальних норм: "норма", "моральна норма", "правова норма", "імператив", "команда", "правило", "регулятив поведінки";
- терміни, що позначають значення істинності висловлювань; за значенням істинності деонтична логіка є багатозначною, тобто висловлюванням з деонтичною модальністю надають істиннісних значень: істинне (I); хибне (X); невизначене (1/2).

Сучасна деонтична логіка продовжує розвиватись, норми в ній інтерпретуються як складно - структуровані багаторівневі відношення, на основі яких пропонуються формалізм, побудований за допомогою спеціальних розширень базової мови класичної логіки.

Залежно від специфіки розуміння дії, виділяються три методологічні платформи логіки норм: деонтична логіка дій, прагма-деонтична логіка, немонотонна деонтична логіка. Межі між цими напрямками досить умовні. Для всіх трьох платформ характерне використання апарату мультиагентних систем, де дія, а також подання про деяке стан справ чи намір є суб'єктним, тобто приписується якомусь агенту. Виразні можливості мультиагентних логік дозволяють формально представити взаємодію нормативних систем, що розуміються як стратегії або програми дій різних агентів.

- Запитання для самоперевірки:

Запитання для самоперевірки:

1. Що називають деонтичною логікою?
2. Назвіть деонтичні оператори та їх позначення.
3. Які поняття називають деонтичними?
4. Яке поняття називають дозволеним?
5. Яке поняття називають обо'вязковим?
6. Яке поняття називають забороненим?
7. Що таке байдужа дія?
8. Назвіть складові частини стандартної деонтичної логіки.

Коментарі. Основні відомості щодо поняття деонтичної логіки, визначення різниці між дозволеними, обов'язковими та забороненими поняттями, закони і дії деонтичної логіки описані в [15].

материалы в разработке

Тема 9

- Інтуїціоністська логіка
  - 9.1. Числення висловлювань в інтуїціоністській логіці
  - 9.2. Доведення формул числення висловлювань в інтуїціоністській логіці висловлювань
  - 9.3. Застосування моделі Кріпке в інтуїціоністській логіці

Інтуїціоністська логіка

Інтуїціоністська логіка є однією з найбільш важливих гілок некласичної логіки. Ставлячи на перший план математичну інтуїцію, інтуїціоністи не надавали великого значення систематизації логічних правил. Тільки в 1930 р. голландський математик і логік А. Гейтінг - учень творця інтуїціонізму Л. Брауера – дав аксіоматичне формулювання інтуїціоністської логіки. Так, у інтуїціоністській логіці не діє закон виключаючого третього. Відкидання закону виключаючого третього не означає прийняття заперечення цього закону, а навпаки, інтуїціоністська логіка стверджує, що заперечення заперечення цього закону, тобто його подвійне заперечення, є правильним.

У класичній логіці центральну роль відіграє поняття істини. На його основі визначаються логічні зв'язки, що дозволяють будувати складні висловлювання. В інтуїціоністській логіці сенс зв'язок задається шляхом зазначення тих необхідних і достатніх умов, при яких може стверджуватися складне висловлювання.

Так, наприклад, якщо p і q – деякі висловлювання, то їх кон'юнкцію p і q можна стверджувати, тільки якщо можна стверджувати як p , так і q. Диз'юнкцію висловлювань p або q можна стверджувати тоді і тільки тоді, коли можна стверджувати хоча б одне з висловлювань p або q. Математичне висловлювання p можна стверджувати лише після проведення деякої математичної побудови з певними властивостями; відповідно заперечення p можна стверджувати, якщо і тільки якщо є побудова, що приводить до суперечностей припущення про те, що побудова p виконана. Імплікацію висловлювань, якщо p, то q, можна стверджувати так, якщо є така побудова, яка, будучи об'єднана з побудовою p, автоматично дає побудову q.

Інтуїціоністське розуміння логічних зв'язок є таке, що з доказу істинності висловлювання завжди можна витягнути спосіб побудови об'єктів, існування яких стверджується.

Мова інтуїціоністської логіки висловлювань є така, як і в класичної логіки: формули будуються із множини змінних $Var = \{P, Q,..\}$ за допомогою зв'язок $< \&, \vee, \neg, \rightarrow >$.

9.1. Числення висловлювань в інтуїціоністській логіці

Означення 9.1.1. Численням висловлювань в інтуїціоністській логіці називають числення висловлювання , серед аксіом якого немає закону виключаючого третього $A \vee \neg A$. Таке числення називають інтуїціоністським численням висловлювань.

Інтуїціонізм відкидає ідею про те, що всі висловлювання діляться на істинні й помилкові. Із цієї точки зору закон виключаючого третього є безпідставним: $A \vee \neg A$ означає, що для задовільного твердження A можна встановити або A, або його заперечення.

Інтуїціоністське числення висловлювань має вивідні формули:

1. Щоб зрозуміти сенс формули $(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$, необхідно згадати, що заперечення $\neg X$ можна розуміти , як $(X \rightarrow \perp)$, де $\perp$ – наперед хибне твердження. Ця формула показує те, що якщо із A слідує B, а із B випливає наперед хибне твердження, то із A також випливає наперед хибне твердження (поодинокий випадок транзитивності відношення слідування). Виведення цієї формули не використовує закону виключаючого третього. Насправді, за лемою про дедукцію (доведення якої залишається таким самим і для інтуїціоністського числення висловлювань) достатньо довести, що із $(A \rightarrow B)$ і із $\neg B$ виводиться $\neg A$.

2. Щоб вивести формулу $(A \rightarrow \neg\neg A)$, необхідно показати, що із A виводиться $\neg\neg A$, для цього достатньо із A і $\neg A$ вивести дві взаємо протилежні формули (тривіально підходять самі формули A і $\neg A$).

3. Формула $(\neg \neg \neg A \rightarrow \neg A)$ отримується із двох попередніх: візьмемо B таким, що дорівнює $\neg \neg A$ у першій із них.

4. Формула $(\neg A \rightarrow \neg\neg\neg A)$, з іншого боку, є поодиноким випадком другої формули, оскільки три заперечення еквівалентні одному.

5. Комутативність та асоціативність операцій $\wedge$ і $\vee$, також як і дві властивості дистрибутивності, не використовують закону виключаючого третього.

6. Як і раніше, формула $((A \vee B) \rightarrow C)$ рівносильна формулі $((A \rightarrow C) \wedge (B \rightarrow C))$ (імплікації в обидві сторони, які з'єднують ці формули, вивідні в ітуїціоністському численні висловлювань).

7. Застосувавши знак $\perp$ як C у попередніх формулах, можна спостерігати, що один із законів де Моргана , а саме закон $\neg (A \vee B) \leftrightarrow \neg A \wedge \neg B$ не спирається на закон виключаючого третього.

8. Формулу $\neg \neg (A \vee \neg A)$ за допомогою збереженого закону де Моргана можна переписати у вигляді $\neg (\neg A \wedge \neg \neg A)$, і необхідно лише вивести із $(\neg A \wedge \neg \neg A)$ дві протилежні формули, що очевидно.

9.2. Доведення формул числення висловлювань в інтуїціоністській логіці висловлювань

Теорема 9.2.1. Формула $p \vee \neg p$ не вивідна в інтуїціоністській логіці.

Доведення. У класичній логіці кожна пропозиційна змінна може набувати два значення – істина (I) або хибність (X). Розширимо множину істинністних значень шляхом введення нового значення H (назвемо це значення словом «*невідомо*»). Оскільки ми ототожнюємо значення I з одиницею, а значення X з нулем, то логічно можливо ототожнювати значення H з числом ½. Ми доведемо, що інтуїціоністські вивідні формули завжди набувають значення I, а формула $p \vee \neg p$ не така і тому не вивідна.

Щоб визначити значення формул у тризначній логіці, необхідно задати таблиці істинності для всіх пропозиційних зв'язок. Кон'юнкцію визначимо як мінімум із двох значень ($X \wedge H = X$, а $I \wedge H = H$), а диз'юнкцію – як максимум ($X \vee H = = I$, $X \vee I = I$, $H \vee I = I$). Заперечення діє як : $\neg I = X$, $\neg X = I$, $\neg H = X$. В останньому не можна вважати $\neg H = H$, тому що тоді формула $\neg (p \wedge \neg p)$, яка вивідна в ітуїціоністській логіці , матиме значення H при $p = H$.

Найскладніше визначення істинності для імплікації. Ми вважаємо, що

$$(I \rightarrow x) = x \; i \; (X \rightarrow x) = I$$

для будь-якого значення істинності x, а також що

$$(H \rightarrow X) = X, \; (H \rightarrow H) = I \; i \; (H \rightarrow I) = I.$$

Назвемо формулу потрійною тавтологією, якщо вона набуватиме значення I при будь-яких значеннях змінних із множини $\{ I, X, H \}$. Тепер потрібно перевірити дві речі: усі аксіоми інтуїціоністського числення є потрійними тавтологіями; якщо посилка імплікації й уся імплікація є потрійною тавтологією, то і висновок теж є потрійною тавтологією. Друге відразу випливає з визначення імплікації, а перше – із доведення усіх аксіом за допомогою таблиць істинності. Отже, будь-яка інтуїціоністська формула, що виводиться, є потрійною тавтологією. Тепер помітимо, що формула $p \vee \neg p$ набуває значення H при $p = H$ і тому не є потрійною тавтологією , а це означає, що вона невивідна , що й потрібно було довести.

9.3. Застосування моделі Кріпке в інтуїціоністській логіці

Для задання моделі Кріпке необхідно:

1. Вказати частково впорядковану множину $\{ W, \leq \}$, яку називають множиною світів.

2. Для кожного світу вказати, які із пропозиційних змінних вважаються істинними у цьому світі (інші змінні вважаються хибними у цьому світі). Якщо змінна x істинна у світі W, то ми пишемо $W \Vdash x$.

При цьому необхідно, щоб була виконана наступна умова, якщо $u \leq v$ і $u \Vdash x$, то $v \Vdash x$ (область істинності будь-якої формули спадкова вгору). Коли модель Кріпке задана, то можна визначити істинність будь-якої формули у цьому світі індукцією щодо побудови самої формули.

Приклад 9.3.1. Визначити істинність формули $W \Vdash A \wedge B$ у світі W .

Розв'язання. Користуючись індукцією визначення побудови формули визначаємо, якщо $W \Vdash A$ і $W \Vdash B$, то тоді є істинною і формула $W \Vdash A \wedge B$.

Приклад 9.3.2. Визначити істинність формули $W \Vdash A \vee B$ у світі W .

Розв'язання. Користуючись індукцією визначення побудови формули визначаємо, якщо $W \Vdash A$ або $W \Vdash B$, то тоді є істинною і формула $W \Vdash A \vee B$.

Приклад 9.3.3. Визначити істинність формули $W \Vdash A \rightarrow B$ у світі W .

Розв'язання. Користуючись індукцією визначення побудови формули в будь-якому світі $u \geq w$, в якому істинна формула A, тоді буде також істинною і формула B.

Приклад 9.3.4. Визначити істинність формули $W \Vdash \neg A$ у світі W .

Розв'язання. Користуючись індукцією, якщо ні в якому світі $u \geq w$, формула A не є істинною , тоді формула $W \Vdash \neg A$ є істинною у світі W .

Індукцією щодо побудови формули A легко перевірити, що якщо вона істинна в будь-якому світі, то вона є істинною і в усіх більших світах.

Формула, яка не є істинною в заданому світі, називається хибною в ньому.

Визначення істинності для заперечення узгоджене з розумінням $\neg A$ як $A \rightarrow \perp$, де $\perp$ - тотожно хибна у всіх світах формула.

Теорема 9.3.1. Якщо формула вивідна в інтуїціоністському численні висловлювань, то вона істинна в усіх світах усіх моделей Кріпке.

Доведення. Для доведення необхідно перевірити, що всі аксіоми істинні в усіх світах, а також, що правило « *modus ponens* » зберігає цю властивість.

Інше очевидно , тому що , якщо $A \rightarrow B$ істинна в усіх світах і A істинна в усіх світах, то за визначенням істинності імплікації B також буде істинна в усіх світах. Тепер перевіримо істинність усіх аксіом. Для першої аксіоми $A \rightarrow (B \rightarrow A)$ можна сказати, що якщо формула A істинна в деякому світі, то внаслідок монотонності вона також істинна і вище , тому і $B \rightarrow A$ також є істинною. Для другої аксіоми $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$ можна сказати таке. Нехай $u \Vdash A \rightarrow (B \rightarrow C)$. Необхідно впевнитися, що $u \Vdash (A \rightarrow B) \rightarrow (A \rightarrow C)$. Але це означає, що якщо $v \geq u$ і $v \Vdash A \rightarrow B$, то $v \Vdash A \rightarrow C$. Останнє, в свою чергу, означає, що якщо $w \geq v$ і $w \Vdash A$, то $w \Vdash C$. Проте внаслідок монотонності ми знаємо, що $w \Vdash A \rightarrow (B \rightarrow C)$ і $w \Vdash A \rightarrow B$. Тому із $w \Vdash A$ випливає $w \Vdash B$, $w \Vdash (B \rightarrow C)$ і $w \Vdash C$, що і потрібно було довести. Доведення решти аксіом відбувається ще

простіше, а тому їх доведення пропонується виконати самостійно.

Теорема 9.3.2. Якщо деяка формула не вивідна в інтуїціністському численні висловлювань, то достатньо подати модель Кріпке в одному зі світів , у якому вона хибна.

Доведення. Для доведення покажемо , що в цьому випадку є модель, у якій серед світів є найменший і в ньому формула хибна. Для формули $p \vee \neg p$ така модель будується просто. Візьмемо два світи так, щоб перший був менший, ніж інший. Нехай p є істинним тільки в другому світі. Тоді $\neg p$ буде не істинним ніде, а $p \vee \neg p$ буде істинним тільки в другому світі. Це доведення, по суті, збігається з наведеним у § 9.2 , оскільки в цій моделі Кріпке для формули є три можливості: вона істинна в обох світах; вона істинна тільки в другому світі; вона не істинна ні в одному зі світів. Ці три інтерпретації моделі Кріпке відповідають таким значенням тризначної логіки : *I, H, X* § 9.2 . Для формули $\neg \neg p \to p$ можна застосувати ту саму шкалу (p істинне тільки в другому світі). Цю шкалу можна застосувати для формули $(\neg q \to c) \to (q \to p)$, поклавши p істинним у двох світах, а q – тільки в другому. Для решти трьох формул доцільно розглянути шкали з трьома світами: початковим світом u , із якого можливо потрапити в $v \geq u$ і $w \geq u$; світи v і w є непорівнянними. Якщо формула p істинна тільки у світі v , то формула $\neg p$ істинна тільки у світі w, а $\neg \neg p$ – істинна тільки у світі v , так, що у світі u обидві формули $\neg p$ і $\neg \neg p$ є хибні й диз'юнкція $\neg p \vee \neg \neg p$ теж хибна. Для того, щоб побудувати контрмодель для формули $\neg (p \wedge q) \to \neg p \vee \neg q$, необхідно вважати,що p істинна тільки у світі v , а q істинна тільки у світі w. Таку саму шкалу можна застосовувати і для формули $((p \vee q) \to p) \vee ((p \vee q) \to q)$.

Запитання для самоперевірки:

1. Сформулюйте визначення інтуїціністської логіки.
2. Яка мова використовується в інтуїціністській логіці ?
3. Що називають численням висловлювань в інтуїціністській логіці ?
4. Які формули має числення висловлювань в інтуїціністській логіці ?
5. На чому ґрунтується доведення формул в інтуїціністській логіці ?
6. Що таке модель Кріпке ?
7. Що необхідно для задання моделі Кріпке ?
8. Для яких цілей застосовують модель Кріпке ?
9. Яку формулу називають вивідною в численні висловлювань інтуїціністської логіки ?
10. Яку формулу називають не вивідною в численні висловлювань інтуїціністської логіки ?

Коментарі. Основні відомості, щодо по поняття інтуїціонітської логіки, викладені у цьому розділі, випливають із [15], числення висловлювань взяті з [7,15,25], а моделі Кріпке і доведення формул з інтуїціністської логіки випливають із [8,9].

материалы в разработке

Розділ 2 Елементи теорії алгоритмів

Тема 10

- Основні визначення, властивості та способи задання
  - 10.1. Поняття про алгоритм. Еволюція тлумачення та властивості
  - 10.2. Способи задання алгоритмів

Основні визначення, властивості та способи задання

А чи існує алгоритм любові?
(риторичне запитання)

Поняття алгоритму належить не лише до фундаментальних наукових понять, а й до людських надбань. Ми в оточенні різноманітності алгоритмів у всіх сферах життя і діяльності. Наші дії багато в чому доведені до несвідомого автоматизму й ми часом і не усвідомлюємо, що вони вже регламентовані тим чи іншим алгоритмом. Адже життя кожного із нас складається з виконання повсякденних алгоритмів, починаючи із засвоєння алгоритмів дитинства, як – от складання пірамідки, і закінчуючи пошуком алгоритмів для вирішення наукових проблем. Узагалі будь-яку людську діяльність можна подати у вигляді алгоритмічного процесу. Автори огляду основних досягнень теорії алгоритмів [1] стверджують: "Алгоритмічні концепції відіграють у процесі навчання і виховання сучасної людини фундаментальну роль, порівнянну лише з роллю писемності".

На одній із наукових конференцій (1984 р.) академік А. А. Самарський зобразив на слайді Інформатику красунею, яка мчить по неосяжному науковому океану на трьох китах. Імена тих китів: Модель, Алгоритм, Програма. Центральний кит - теорія алгоритмів, поєднавшись із математичною логікою, становлять теоретичний фундамент сучасних комп'ютерних наук.

І хоча інтуїтивно поняття алгоритму зрозуміле, мабуть, кожному, формально визначити цей об'єкт у нетривіальній теорії алгоритмів зовсім не так просто, як здається на перший погляд. По-перше, насторожує надмірна популярність цього поняття і, як у таких випадках кажуть, його розширене тлумачення. Так, вам могли траплятися такі різні алгоритми : "алгоритм Евкліда", "алгоритм шахіста" , "алгоритм пошуку інформації", "алгоритм, як бути красунею" та інші. По-друге, алгоритми, як самі по собі суть об'єкти специфічного типу, які важко описати математично. Вони мають нетипову для математичних об'єктів властивість, а саме семантичну властивість "мати зміст". У цьому відношенні теорія алгоритмів подібна до математичної логіки. Із семантичними об'єктами, що несуть у собі зміст, математики, ще не звикли поводитися. Сенс терму або формули в математичній логіці "вказівний": терм вказує на річ (тобто позначає), а формула - на факт. Сенс алгоритму "наказовий": алгоритм повинен бути виконаний. Таким чином, теорія, що вивчає алгоритми, може трактуватися як свого роду лінгвістика наказових пропозицій. І, по-третє, теорія алгоритмів виникла на стику математики та інформатики. Тому вчені з обох боків "тягнули до себе ковдру" про побудові теорії алгоритмів. Цю специфіку можна помітити, вивчаючи монографії та підручники з теорії алгоритмів. Залежно від того, ким вони написані - математиками чи знавцями інформатики, ми відчуваємо особливість змісту матеріалу, відображеного в них [1-5].

У сучасності слово "алгоритм" як науковий термін вийшло за межі математики. Цей термін застосовується в найрізноманітніших галузях науки і техніки, розуміючи при цьому точно сформульоване правило, призначення якого – керувати діями для досягнення необхідного результату.

10.1. Поняття про алгоритм. Еволюція тлумачення та властивості

Поняття "**алгоритм** " - концептуальна основа різноманітних процесів у інформатиці. Саме наявність алгоритмів і забезпечує можливість автоматизації. Більше того, значною мірою теорія алгоритмів допомагає для проникати математичним методам у всі сфери життя і науки, навіть біологію, лінгвістику, економіку аж до філософії природознавства [1-12].

Поняття алгоритму формувалося з найдавніших часів і до тридцятих років XX ст., у науковому колі склалось інтуїтивне уявлення про цей об'єкт. Одним із найдавніших алгоритмів, у його інтуїтивному розумінні, як скінченної послідовності елементарних дій, що вирішують поставлене

завдання, вважається алгоритм знаходження найбільшого спільного дільника двох чисел (алгоритм Евкліда). Відзначимо, що протягом тривалого часу, аж до початку XX століття, термін "алгоритм" вживався лише в стійкому сполученні "алгоритм Евкліда", хоча історично встановлено, що задовго до Евкліда цей алгоритм був уже відомий, наприклад, Аристотелю.

Змістовні явища, що привели до утворення поняття "алгоритм", простежуються в математиці впродовж усього часу її існування. Саме слово "алгоритм", як стверджують історики та математики, з'явилося декілька століть тому й означало не термін, а ім'я. Узбецький математик **Аль-Хорезмі**, вчений, якому математика й людство зобов'язані багатьма відкриттями, був відомий європейським математикам у латинській транскрипції як Алгоризмі. А повне його ім'я - Абу Абдулла Абу Джафар Мухаммад ібн Муса аль-Хорезмі (близько 780 р. — близько 850 р.) - у перекладі буквально - батько Абдулли, Мухаммед, син Муси, уродженець Хорезмі. Його книга - "Хисаб аль-джабр у аль-мукабаля" . Саме із трактату Аль-Хорезмі з арифметики почалося знайомство Європи з алгоритмами - строгими процедурами для виконання арифметичних операцій. Тобто алгоритм, точніше алгоризм, розуміли як керівництво для вирішення завдань.

Подальший розвиток математики затвердив ту думку, що вирішення будь-якої проблеми повинне бути алгоритмічним. Р. Декарт, Г. Ф. Лейбніц,

Д. Гільберт стимулювали алгоритмічні дослідження. В 1900 на Другому міжнародному конгресі математиків (Париж) Давид Гільберт сформулював 23 важливі математичні проблеми, знаходження алгоритму розв'язування яких, на його думку, сприяло б подальшому розвитку математики.

Саме в ті часи були поширені дві точки зору :

1. Усі проблеми є алгоритмічно розв'язними. Просто ще не існують знання для побудови алгоритму.
2. Існують класи завдань, для вирішення яких взагалі не існує алгоритмів. Це дуже сильне твердження, тому що воно поширюється на все майбутнє.

Для доведення не існування алгоритму потрібно було мати його точне математичне визначення, тому після сформування поняття алгоритму як нової та окремої сутності першочерговою постала проблема знаходження адекватних формальних моделей алгоритму та дослідження їх властивостей. Формальні моделі були запропоновані як для первісного поняття алгоритму, так і для похідного поняття алгоритмічно обчислюваної функції. Вперше алгоритм як термін з'явився у працях Е. Бореля (1912) та Г. Вейля (1921).

Початковою точкою відліку сучасної теорії алгоритмів можна вважати працю німецького математика Курта Ґьоделя (1931 рік - теорема про неповноту символьних логік), у якій було показано, що для будь-якої несуперечливої системи аксіом існує твердження, яке в рамках прийнятої аксіоматичної системи не може бути ні доведене, ні спростоване. Виникле у зв'язку з цією теоремою припущення про неможливість алгоритмічного вирішення багатьох математичних проблем (зокрема, проблеми вивідності в численні предикатів) викликало необхідність уточнення поняття алгоритму.

Перші фундаментальні праці з теорії алгоритмів були опубліковані незалежно в 1935-37 роках А. Тюрінгом, А. Черчем, С. Кліні, К. Гьоделем і

Е. Постом. Запропоновані ними машина Тюрінга, машина Поста, частково рекурсивні функції й лямбда-вирахування Черча були еквівалентними формалізмами алгоритму. Сформульовані ними тези (Поста й Черча-Тюрінга) постулювали еквівалентність запропонованих ними формальних систем та інтуїтивного поняття алгоритму. Важливим розвитком цих праць стало формулювання й доведення алгоритмічно нерозв'язних проблем. У 1950-ті - роки істотний внесок у теорію алгоритмів зробили праці

А. М. Колмогорова та А. А. Маркова (молодшого), а в 1970-х роках у теорії алгоритмів плідно працював Ю. В. Матіясевич, зокрема над розв'язанням проблем Гільберта.

Запропоноване А. Марковим поняття "нормального алгоритму" є величезним внеском у світову науку, а А. Колмогорову належить пріоритет у визначенні загального поняття алгоритму та створення теорії складності конструктивних об'єктів.

У 1960-70-ті роки сформулювалися такі напрямки в теорії алгоритмів:

- **Класична теорія алгоритмів** (викладення завдання у термінах формальних мов, поняття задачі вирішення, введення класів складності, формулювання в 1965 році Едмондсом проблеми P=NP, відкриття класу NP-повних задач і його дослідження)[1].
- **Теорія асимптотичного аналізу алгоритмів** (поняття складності й трудомісткості алгоритму, критерії оцінки алгоритмів, методи одержання асимптотичних оцінок, зокрема для рекурсивних алгоритмів, асимптотичний аналіз трудомісткості або часу виконання), у розвиток якої зробили істотний внесок Кнут, Ахо, Хопкрофт, Ульман [2,4].
- **Теорія практичного аналізу обчислювальних алгоритмів** (одержання явних функцій трудомісткості, практичні критерії якості алгоритмів, методика вибору раціональних алгоритмів), основними роботами в цьому напрямку, мабуть, потрібно вважати фундаментальні праці [1,2].

Відтодібагато вчених були причетними до розвитку теорії алгоритмів. Необхідно відзначити також чималий внесок у теорію алгоритмів, зроблений Д. Кнутом, А. Ахо і Дж. Ульманом. Потрібно пригадати і друге видання книги "Алгоритми: побудова й аналіз" Томаса Х. Кормена, і праці Чарльза І. Лейзерсона, Рональда Л. Ривеста, Кліффорда Штайна.

Учені, що упорядковували теорію алгоритмів, рано чи пізно стикалися з тим, що потрібно означити об'єкт досліджень. Виявилося, що всі різноманітні означення, подані ними щодо алгоритму, є еквівалентними поняттями, й чудовим науковим результатом є доведення еквівалентності цих формальних визначень у змісті їх рівнозначності. Наводячи всі означення, як кажуть математики, "до загального знаменника", можна дати загальне визначення алгоритму:

Алгоритм – спосіб перетворення інформації за правилами, сформульованими певною мовою.

Незважаючи на зусилля дослідників, відсутнє єдине вичерпно строге визначення поняття алгоритму. Всі вони є інтуїтивними, так само й те, що наведене вище. Інтуїтивно кажучи, щоб отримати розв'язок будь-якої задачі потрібно виконати дії за строго визначеним формальним приписом. Цей формальний припис і є алгоритмом.

У загальному випадку зрозуміло, що алгоритм – це детермінована процедура, яку можна застосувати до будь-якого елемента певного класу символьних входів і яка для кожного такого входу видає через скінченне число дій (кроків) відповідний результат своєї дії. Нехай D - область (множина) вихідних даних завдання, а R - множина можливих результатів, тоді ми можемо говорити, що алгоритм A здійснює відображення A : D ® R.

Будь-який алгоритм повинен мати такі основні властивості:

Визначеність (однозначність) - кожна команда алгоритму однозначно визначає дії виконавця і не припускає подвійного тлумачення.

Дискретність – можливість розбивки алгоритму на скінченну кількість етапів, причому результати попереднього етапу є вихідними для наступного.

Масовість – алгоритм може бути використаний для вирішення цілого класу однотипових завдань, для яких він створений.

Зрозумілість – алгоритм повинен бути зрозумілим конкретному виконавцеві, який повинен виконати кожну команду алгоритму в строгій відповідності із призначенням.

Результативність (збіжність) – виконання алгоритму повинне закінчуватися результатом або інформацією про те, що не може бути отриманим результат.

Саме через ці властивості часто дається інтуїтивне визначення алгоритму як скінченної однозначно визначеної послідовності операцій, формальне виконання якої приводить до розв'язання поставленої задачі за скінченне число кроків.

Приклад 10.1. Розглянемо алгоритм Евкліда знаходження найбільшого спільного дільника двох натуральних чисел *а* і *в* (НСД (а, в)). Цей алгоритм переробляє слова виду (*а*,*в*) у алфавіті, що складається з 10 цифр, дужок " (" та ")", знака пунктуації " , ", у слова того самого алфавіту (результатом буде слово із цифр). Скінченна система правил, що визначають цей алгоритм, складатиметься з правила ділення, визначення остачі від ділення і правила визначення, коли алгоритм закінчує роботу і видає результат.

Приклад 10.2. Розглянемо процес приготування страви швидкого харчування : 1. Висипати в порожній посуд вміст пакетика. 2. Залити в ємність 200 *мл* гарячої води. 3. Ретельно перемішати. Щоб виконати скінченну систему правил за цим "алгоритмом», потрібно проробити суто механічні дії.

Проте останній приклад не є алгоритмом у тому сенсі, що ми визначили. Оскільки він не відповідає всім вимогам до його властивостей, дамо йому визначення "майже" алгоритм, так само як і медичним рецептам. Подумайте чому саме? Хоча такий розподіл.

Зрозуміло, що *алгоритм* як сутність прийшов до фундаментальних надбань людства з математики, а в сучасності одержав статус центральної вісі програмування та інформатики в цілому. За своєю природою алгоритм – це інтелектуальний продукт, який є винайденою його створювачем послідовністю дій над вхідною інформацією, строге виконання якої приведе до бажаного чи не бажаного результату. Алгоритм існує тоді і тільки тоді (або починає існувати), коли існує в той самий час деякий об'єкт (найчастіше математична його модель) або проблема, для вирішення якої він і створений. За призначенням алгоритм зі вхідними даними зіставляє вихідні дані й у цьому він чимось схожий з позицій математики на звичайну функцію, а з позицій інформатики - на спосіб перетворення інформації. Проте головною особливістю алгоритму є те, що він містить опис саме дій, як це зробити. Функція може бути задана неявно, а алгоритм - ні. Алгоритм описує одним із можливих способів свого подання (див. п.10.2), що потрібно зробити із вхідними даними, щоб отримати результат.

Подане вище тлумачення алгоритму не можна вважати строгим у науковому сенсі, адже не цілком зрозуміло, що таке "послідовність дій, що забезпечує отримання необхідного результату". Тому зазвичай формулюють декілька загальних властивостей алгоритмів, що дозволяють відрізняти алгоритми від інших подібних інструкцій (див. п.10.1). Однак і зі знанням цих властивостей іноді буває важко визначити, чи є подані інструкції алгоритмом, чи ні. Мабуть, однією із визначальних для вирішення цієї проблеми, є властивість однозначності виконання кожної команди, поданої в інструкціях. Якщо результат виконання кожного кроку інструкції відбувається однозначно і незалежно від виконавця, то ці інструкції становлять саме алгоритм.

10.2. Способи задання алгоритмів

Якщо взяти за основу, що алгоритми з позицій математики є строгими правилами розв'язання будь-якої задачі, а з позицій інформатики є центральною ланкою в ланцюгу модель-алгоритм-програма (див. вище), то правильність алгоритмів є запорукою належного функціонування комп'ютерних програм, достовірності результатів у розв'язанні наукових та життєвих проблем і взагалі адекватного світосприйняття. Коротку просту програму можна написати безпосередньо з постановки завдання без попереднього розроблення алгоритму. Проте реальну програму так само важко створити без алгоритму, як важко виготовити досить складну деталь без креслення.

У цьому сенсі, крім правильності самих алгоритмів по суті вирішення самої проблеми, потрібно мати зручне і зрозуміле їх подання для практичного застосування конкретному виконавцю. При заданні алгоритму необхідно подбати про те, щоб алгоритм сприймався всіма можливими виконавцями однозначно і точно, щоб його можна було виконати за будь-яких допустимих вихідних умов, і щоб необхідний результат був отриманий за прийнятний час. Очевидно, що припис "Піди туди, не знаю куди, принеси те, не знаю що" алгоритмом не є.

Існує декілька методів запису або подання алгоритмів. Вибір методу залежить від виконавця й представлених розробником алгоритмів варіантів подання.

Перший спосіб задання алгоритмів - це словесно-формульний (опис здійснюється у словесній формі з використанням математичних або інших формул). Він описує виконання дій алгоритму в певній послідовності за допомогою слів і пропозицій природної мови. Форма викладу довільна і встановлюється розробником. У математиці наявність формул дозволяє розв'язати задачу, навіть "не використовуючи слів".

Приклад 10.3. *Записати алгоритм знаходження найбільшого спільного дільника (НСД) двох натуральних чисел (алгоритм Евкліда).*

Розв'язання. Алгоритм може бути поданий таким чином:

1. Поставити два числа.
2. Якщо числа однакові, то взяти будь-яке з них як відповідь і зупинитися, інакше продовжити виконання алгоритму.
3. Визначити більше із чисел.
4. Замінити більше із чисел різницею більшого і меншого із чисел.
5. Повторити алгоритм з кроку 2.

Другий спосіб задання алгоритмів - це подача алгоритму у вигляді таблиць, формул, схем, рисунків тощо. Наприклад, всіх нас навчали в дитячому садочку правил поведінки на дорозі. І найкраще діти, вочевидь, сприймають алгоритм, поданий у вигляді схематичних малюнків. Дивлячись на них, дитина, а потім і доросла людина, відпрацьовує ту лінію поведінки, що їй пропонується.

Третій спосіб задання алгоритмів - це запис алгоритмів за допомогою блок-схеми. Цей метод був запропонований в інформатиці для наочності подання алгоритму за допомогою набору спеціальних блоків. Блок-схема – це таке графічне подання алгоритму, при якому розв'язання задачі зображується у вигляді різних геометричних фігур, усередині яких записані тексти, що разом із формою фігури пояснюють дію, яку потрібно виконати, та взаємозв'язків між ними.

Схема алгоритму чітко визначає послідовність дій, задану цим алгоритмом. Відповідно до властивості дискретності схема може представляти алгоритм із різним ступенем деталізації. При цьому важливо найбільш точно і наочно зображувати керуючі структури алгоритму, що показують логіку переходу від однієї дії до іншої. Проте вимоги до графічного виконання схем викликають упередження деяких програмістів, представляються основним недоліком схем і змушують шукати нові способи запису алгоритмів.

Приклад 10.4. *Використовуючи необхідні для представлення блоки, можна подати, наприклад, алгоритм чищення картоплі в такому вигляді:*

Четвертий спосіб задання алгоритмів – це запис алгоритму навчальною алгоритмічною мовою (псевдокодом). Псевдокодом називається система правил запису алгоритму з використанням набору певних конструкцій для опису керуючих дій. Псевдокод є основним способом подання алгоритмів для теоретичних досліджень у теорії алгоритмів. Саме на цих конструкціях відпрацьовується, наприклад, знаходження асимптотичних оцінок алгоритмів.

Єдиного, або формального, визначення псевдокоду не існує, тому можливі різні псевдокоду, що відрізняються набором службових слів й основних (базових) конструкцій. На відміну від стандартизації синтаксису мов програмування, на синтаксис псевдокоду зазвичай не встановлює стандартів, оскільки останній безпосередньо не компілюється у виконувану програму. Тому можна сказати, що зазвичай автор кожної публікації застосовує свій оригінальний псевдокод, запозичуючи потрібні їм конструкції з будь-якої мови програмування. Їх використання можна пояснити як особистими симпатіями автора, так і поширеністю на момент написання публікації. У разі російськомовних публікацій як псевдокод часто використовується переклад ключових слів мов програмування з англійської на російську.

У ряді випадків псевдокодом називають систему команд абстрактної машини, наприклад, P-код, псевдокод вигаданої машини MIX і т. д. На відміну від псевдокоду неформального характеру такий псевдокод уже строго формалізований і може бути транслюваний у працюючу програму за наявності програми-емулятора даної гіпотетичної машини.

Відомі прогнози, які стверджують, що подальший розвиток мов програмування піде шляхом їх зближення із псевдокодом, що в кінцевому етапі дозволить здійснювати програмування на природних мовах.

Приклад 10.5. Записати на псевдокоді алгоритм Евкліда – знаходження найбільшого спільного дільника двох чисел.

Розв'язання.

```
1 функція нсд(a, b)
2     якщо b = 0
3         поверни a
4   інакше
5         поверни нсд(b, a mod b)
```

Ви, мабуть, помітили, що цей варіант алгоритму відрізняється від алгоритму, поданого під час розв'язання прикладу 10.1. Насправді існує декілька варіантів написання алгоритму Евкліда, в даному разі записано у псевдокоді рекурсивний варіант.

П'ятий спосіб, максимально наближений до комп'ютера, задання алгоритмів - це мови програмування. Справа в тому, що найчастіше у практиці виконавцем створеного людиною алгоритму є машина, і тому він повинен бути написаний мовою, зрозумілою для комп'ютера, тобто мовою програмування. Мова програмування - це знакова система, призначена для опису процесів вирішення завдань та їх реалізації на ЕОМ. Реалізація означає, що описи можуть бути введені в ЕОМ і однозначно нею зрозумілі. До мов програмування належать мови команд або машинні мови та мови високого рівня.

Запитання для самоперевірки

1. Сформулюйте основні властивості алгоритмів.
2. Укажіть напрямки розвитку теорії алгоритмів.
3. Які способи подання алгоритмів існують?
4. Що є псевдокод?
5. Сформулюйте поняття "масовість» у теорії алгоритмів.

- Задачі для самостійного розв'язання

Задачі для самостійного розв'язання

1. Чому кулінарні рецепти не є алгоритмами?
2. Знайдіть за допомогою алгоритму Евкліда найбільший спільний дільник чисел 2142 та 924. Опишіть алгоритм Евкліда в термінах відображень.
3. За наведеним нижче псевдокодом визначте результат дії алгоритму:

1 Покласти кількість K=0
2 ЦИКЛ Перебір всіх елементів масиву від 1 до КількістьЕлементів(A)
3 ЯКЩО A(i) = 0 ТОДІ K=K+1
4 КІНЕЦЬ ЦИКЛУ
5 Вивід на екран K.

4. За наведеним нижче псевдокодом визначте алгоритм у словесному описі:

1 Покласти I=0
2 ЦИКЛ ПОКИ I < n ВИКОНУВАТИ
3 ЯКЩО X=A(i) ТОДІ m=i
4 I=I+1
5 КІНЕЦЬ ЦИКЛУ
6 ЯКЩО m=0 ТОДІ m= -1
7 КІНЕЦЬ.

Коментарі. Цей розділ є вступним і тому в ньому задіяна вся бібліографія з теорії алгоритмів, а також матеріали INTERNET, наприклад сайту - www.intuit.ru.

Тема 11

- Алгоритми та обчислювальні функції
  - 11.1. Основні означення
  - 11.2. Оператор суперпозиції
  - 11.3. Оператор примітивної рекурсії
  - 11.4. Оператор мінімізації
  - 11.5. Гіпотеза Черча та примітивно-рекурсивні функції

Алгоритми та обчислювальні функції

11.1. Основні означення

Період до початку XX століття можна вважати етапом накопичування інформації про алгоритми. Інтуїтивне розуміння алгоритму було достатнім, щоб вважати ту чи іншу процедуру розв'язання задач алгоритмом. Проте одночасно нагромаджувались проблеми, пошук алгоритму вирішення яких ні до чого не приводив. Та для того щоб довести, що алгоритм існує, достатньо його пред'явити, а ось для доведення його відсутності необхідне вже строге визначення шуканого об'єкта. Цим питанням і потрібно було зайнятися.

Оскільки в математиці поняття алгоритму тісно пов'язане з поняттям функції, то історично першою формалізацією алгоритму став клас обчислювальних функцій (К. Гьодель, А.Черч, 1935-1936 рр.). Основна ідея в тому, що довільний алгоритм можна звести до обчислення значення деякої числової функції, тобто з кожним алгоритмом можна зв'язати функцію, яку він обчислює. При цьому виникає ряд запитань: чи для будь-якої функції існує обчислювальний її алгоритм; для яких функцій алгоритми існують і як описати такі алгоритмічні функції? Пошук відповіді на ці питання й привів до створення теорії рекурсивних функцій.

При цьому необхідно зазначити, що в теорії обчислювальних функцій

визначають множину натуральних чисел $N=\{0,1,2,3,\ldots\}$ і розглядають тільки числові функції, розуміючи під ними функції k змінних (k - місні функції), аргументи і значення яких належать N. Тобто, об'єкти з областю визначення $D_f \subseteq N^k$ (k – ціле додатне) та з областю значень $R_f \subseteq N$ будемо називати k - місними частковими функціями. Термін "часткова" нагадує, що функція визначена на підмножині N^k(звичайно, може статися, що $D_f = N^k$, в такому разі функція стає всюди визначеною). Виходячи з вищезазначене, дамо означення обчислювальної функції.

Означення 11.1.1. Числову функцію $f: N^k \to N$ називають обчислювальною, якщо існує алгоритм, за допомогою якого можна обчислити значення функції для будь-якого набору значень аргументів із області визначення функції.

Надалі використовуватиме ідею К. Гьоделя та С. Кліні (1936 р.), за якою всі обчислювальні функції можна одержати із множини базисних функцій та алгебраїчних операцій. Самі операції прийнято називати операторами.

Розглянемо клас числових функцій, що використовуються як базис для побудови обчислювальних функцій:

1. $O(x)=0$ - нуль-функція (можна задати і n - місну нуль - функцію $O^n(x_1, x_2, \ldots, x_n) = 0$).
2. $S(x)=x+1$ - функція слідування або наступності (але не додавання одиниці).
3. $I^n_m (x_1, x_2, x_3,.., x_m, \ldots x_n) = x_m$ - функція проекції або введення фіктивних змінних або вибору аргументу.

Як операторів, застосування яких до базисних функцій призводить до утворення нових функцій оберемо такі три оператори:

1. оператор суперпозиції ;
2. оператор примітивної рекурсії ;
3. оператор мінімізації або найменшого кореня.

У цьому розділі застосуємо алгебраїчний підхід до визначення класу обчислюваних функцій. Кожну обчислювальну функцію будемо одержувати з деяких найпростіших обчислювальних базисних функцій за допомогою операцій, обчислювальність яких також не викликає сумніву. Загальна схема знаходження, яка дала назву цьому підходу, - рекурсія, спосіб задання функції шляхом визначення кожного її значення в термінах раніше визначених її значень та інших уже визначених функцій.

Таким чином, нетривіальні обчислювальні функції можна одержати за допомогою композиції (суперпозиції) вже відомих обчислювальних функцій. Цей спосіб є явно алгоритмічним.

11.2. Оператор суперпозиції

Операція суперпозиції полягає у підстановці одних арифметичних функцій замість аргументів інших функцій.

Нехай задана m-місна функція $F(x_1, x_2, x_3,.., x_m)$ та m-на кількість n-місних функцій $f_1(x_1, x_2, x_3,.., x_n)$, $f_2(x_1, x_2, x_3,.., x_n)$, $f_3(x_1, x_2, x_3,.., x_n),\ldots,f_m(x_1, x_2, x_3,.., x_n)$. Тоді говорять, що n-місна функція $\varphi(x_1, x_2, x_3,.., x_n)$ утворилася в результаті підстановки у функцію F замість її аргументів m функцій $f_1, f_2, f_3,..., f_m$. Така підстановка називається суперпозицією S^n_m. Тоді

$$S^n_m(F, f_1, f_2, \ldots, f_m) = F(f_1(x_1, x_2, .., x_n), f_2(x_1, x_2, .., x_n),\ldots,f_m(x_1, x_2, .., x_n)) = \varphi(x_1, x_2, .., x_n). \quad (11.2.1)$$

Приклад 11.2.1.. Здійснити суперпозицію нуль-функції та функції слідування, тобто знайти $S^2(S(x); O(x))$.

Розв'язання. Для визначення результату операції суперпозиції потрібно функцію $O(x)=0$ підставити в $S(x) = x+1$ замість значення аргументу. Отримаємо $S^2(S(x);O(x)) = S(O(x)) = O(x) + 1 = 0+1 = 1$.

11.3. Оператор примітивної рекурсії

Оператор примітивної рекурсії (R^n) дозволяє будувати $n+1$-місну

арифметичну функцію $f(x_1, x_2, x_3,.., x_n, y)$ з двох заданих функцій, одна з яких є n-місна $\varphi(x_1,x_2,..,x_n)$, а інша - $n+2$-місна функція $\psi(x_1,x_2,..,x_n,y,z)$ за такою схемою: $f(x_1, x_2, x_3,.., x_n,0) = \varphi(x_1,x_2,..,x_n)$;

$$f(x_1, x_2, x_3,.., x_n,y+1)= \psi(x_1,x_2,..,x_n,y, f(x_1, x_2, x_3,.., x_n, y)). \quad (11.3.1)$$

Таким чином, $f(x_1, x_2, x_3,.., x_n, y) = R^n(\varphi,\psi)$.

Для розуміння операції примітивної рекурсії необхідно зазначити, що будь-яку функцію від меншої кількості аргументів можна розглядати як функцію від більшої кількості аргументів. Зокрема, функції - константи (n=0) є одномісними і відповідно: $f(x)=a$; $f(x, y+1) = \psi(x,y, f(y))$, де a – константа.

Схеми примітивної рекурсії визначають функцію f тільки через інші функції φ та ψ, а й через значення f у попередніх точках – значення f у точці $(y+1)$ залежить від значення f у точці (y).

Приклад 11.3.1. Знайти значення функції $f(3,2)$, якщо вона задана співвідношеннями
$f(x, 0) = 0$; $f(x, y+1) = f(x, y) + x$.

Розв'язання. У даному разі за схемою (11.3.1) маємо : $f(x, 0) = \varphi(x) = 0$,

$\psi(x,y,z) = y + z$. Виходячи з того, що $f(x, 0) = \varphi(x) = 0$ при будь-якому x, тоді й $f(2, 0) = 0$. Обчислюючи послідовно, одержимо:

$f(2, 1) = f(2 ,0) + 2 = 0+ 2 = 2;$

$f(2, 2) = f(2, 1) + 2 = 2+ 2 = 4;$

$f(3, 2) = f(2, 2) + 2 = 4+ 2 = 6$, що є остаточною відповіддю.

Нескладно довести, що в даному прикладі $f(x,y)=x \bullet y$.

11.4. Оператор мінімізації

Розглянемо обчислювальну n+1 -місну функцію $f(x_1, x_2, x_3,.., x_n, x_{n+1})$. Зафіксуємо значення $x_1, x_2, x_3,.., x_n$ її перших n аргументів та розглянемо рівняння $f(x_1, x_2, x_3,.., x_n, y) = 0$, тобто знайдемо значення, y при якому функція дорівнює нулю. Більш складною буде задача відшукати найменше з усіх значень y, при якому $f(x_1, x_2, x_3,.., x_n, y) = 0$. Оскільки результат знаходження залежить від $x_1, x_2, ...,x_n$, то й найменше значення y є також їх функцією. Введемо позначення

$$\varphi (x_1, x_2, ...,x_n) = \mu_y [f(x_1, x_2, ...,x_n, y) = 0], \qquad (11.4.1)$$

де y таке, що $f(x_1, x_2, ...,x_n, y) = 0$, а μ_y – оператор мінімізації.

Для знаходження функції $\varphi(x_1, x_2, ...,x_n)$ можна запропонувати таку процедуру.

1. Обчислюємо $f(x_1,...x_n,0)$; якщо її значення буде нуль, то $\varphi(x_1,...x_n)=0$. Якщо $f(x_1, ...x_n,0) \neq 0$, то переходимо до наступного кроку.
2. Обчислюємо $f(x_1,...x_n, 1)$; якщо її значення буде нуль, то $\varphi(x_1,...x_n)= 1$. Якщо $f(x_1, ...x_n,0) \neq 0$, то переходимо до наступного кроку.

І так далі, поки не знайдемо перше значення y, при якому $f(x_1, ...x_n, y) = 0$.

Якщо визначиться, що для всіх y $f(x_1, ...x_n, y) \neq 0$, то функція $\varphi(x_1,...x_n)$ вважається невизначеною.

Приклад 11.4.1. Розглянемо функцію $\varphi(x,y)=x - y$, яку можна отримати за допомогою μ-оператора $d(x,y)=\mu_z[y+z=x]=\mu_z[S(I^3_2 (x,y,z), I^3_3 (x,y,z),)]= I^3_1 (x,y,z)$ та обчислимо $f(7,2)$.

Розв'язання. Задамо y значення 2 та встановимо змінній z послідовно значення 0,1,2.., кожного разу обчислюючи суму $y+z$. Як тільки при якомусь першому в заданому порядку z сума дорівнюватиме 7, то відповідне значення візьмемо за значення $d(7,2)$. Проведемо обчислення:

$z=0$ 2+0=2 < > 7

$z=1$ 2+1=3 < > 7

$z=2$ 2+2=4 < > 7

$z=3$ 2+3=5 < > 7

$z=4$ 2+4=6 < > 7

$z=5$ 2+5=7=7

Таким чином, $d(7,2)=5$.

11.5. Гіпотеза Черча та примітивно-рекурсивні функції

Означення 11.5.1. Функцію називають примітивно-рекурсивною, якщо вона утворена з найпростіших функцій за допомогою скінченного числа застосувань операторів суперпозиції S^n_m та примітивної рекурсії R^n .

Означення 11.5.2. Функцію називають частково-рекурсивною, якщо вона утворена з найпростіших функцій за допомогою скінченного числа застосувань операторів суперпозиції S^n_m , примітивної рекурсії R^n та мінімізації μy.

З цих означень та зауваження щодо зберігання операторами властивості обчислювальності функцій випливає, що кожна стандартно задана частково-рекурсивна функція є обчислювальною за певною процедурою, яка відповідає інтуїтивному уявленню алгоритму, а з іншого боку які б досі не будувалися класи точно визначених алгоритмів, завжди з'ясовувалося, що числові функції, які обчислювалися за алгоритмами цих класів, були частково-рекурсивними. Тому загальноприйнятою є така наукова гіпотеза (теза Черча).

Гіпотеза (теза) Черча. *Клас алгоритмічно обчислювальних часткових числових функцій збігається з класом усіх частково-рекурсивних функцій.*

До формулювання цієї тези входить інтуїтивне поняття обчислювальності, тому його не можна ні спростувати, ні довести. Це факт, на користь якого свідчить багаторічна математична практика.

Історично це була перша гіпотеза щодо зв'язків між класами інтуїтивних і точних алгоритмів.

Зрозуміло, що будь-яка примітивно рекурсивна функція є частково рекурсивною, оскільки за визначенням оператори для побудови частково рекурсивних функцій містять у собі оператори для побудови примітивно рекурсивних функцій. Також зрозуміло, що примітивно рекурсивна функція визначена скрізь і тому є загальнорекурсивною функцією (у примітивно рекурсивної функції немає приводу «*зависати*», тому що при її побудові використовуються оператори, що визначають скрізь визначені функції).

Досить складно довести існування і навести приклад загальнорекурсивної функції, що не є примітивно рекурсивною.

Функція Аккермана — приклад такої обчислювальної функції, яка не є примітивно рекурсивною. Вона набуває два невід'ємних цілих числа як параметри і повертає натуральне число, позначається $A(m;n)$. Ця функція зростає дуже швидко, наприклад, число A (4;4) настільки велике, що кількість цифр у порядку цього числа у багато разів перевершує кількість атомів в спостережуваної частини Всесвіту. Функція Аккермана визначається рекурсивно для невід'ємних цілих чисел m та n таким чином:

$$A(m,n) = \begin{cases} n+1, & m = 0 \\ A(m-1,1) & m>0, n=0; \\ A(m-1,A(m,n-1)), & m > 0, n>0 \end{cases} \qquad (11.5.1)$$

За означенням примітивно-рекурсивної функції, наведеним вище, неважко знайти процедуру, що породжує всі примітивно-рекурсивні функції.

Надамо цьому означенню більш формального індуктивного вигляду.

1. Функції $O(x) =0, S(x)=x+1, I^n_m (x_1, x_2, ..., x_m,...,x_n) = x$ для всіх натуральних n, m є примітивно-рекурсивними.
2. Якщо $g_1(x_1,...,x_n), ..., g_m(x_1,...,x_n), h(x_1,...,x_m)$ – примітивно-рекурсивні функції, то $S^n_m(h,g_1, ...,g_m)$ – примітивно-рекурсивні функції для будь-яких натуральних n,m.
3. Якщо $g(x_1,...,x_n)$, та $h(x_1,...,x_n,y,z)$ — примітивно-рекурсивні функції, то $R^n (g,h)$ — примітивно-рекурсивна функція.
4. Інших примітивно-рекурсивних функцій немає.

Приклад 11.5.1. Навести приклади арифметичних функцій, що є примітивно-рекурсивними.

Розв'язання . **Функція додавання** двох натуральних чисел $Sum(a,b) = =a+b$. Ця функція може бути розглянута як примітивно-рекурсивна функція двох змінних, одержувана внаслідок застосування оператора примітивної рекурсії до функцій I^1_1 та F, другу з яких одержимо підстановкою функції проекції I^3_3 у функцію слідування S. Покажемо це математично:

$Sum (x,0) = I^1_1(x),$

$Sum(x,y+1) = x+y+1=Sum(x,y)+1 = S(Sum(x,y)),$

$Sum(x, y) = R^1(I^1_1(x), F(x, y, z)),$ *де* $F(x, y, z) = S(I^3_3(x, y, z))=z+1$.

Функція множення *двох натуральних чисел* $Mul(a,b) = a \bullet b$.

Ця функція може бути розглянута як примітивно-рекурсивна функція двох змінних, одержувана внаслідок застосування оператора примітивної рекурсії до функцій O та G, другу з яких одержимо підстановкою функції проекції I^1_3та I^3_3 у функцію додавання Sum. Покажемо це математично.

$Mul(x, 0) = O(x),$

$Mul(x,y+1) = x \times y+x = G(x, y, Mul(x, y)),$

$G(x, y, z) = Sum(I^1_3(x, y, z), I^3_3(x, y, z)),$

$Mul(x, y) = R^1(O(x), G(x, y, z)).$

- Запитання для самоперевірки

Запитання для самоперевірки

1. Сформулюйте основну ідею про спосіб побудови обчислювальних функцій.
2. Які функції належать до базисних функцій?
3. Назвіть основні оператори, що застосовуються для породження класу обчислювальних функцій?
4. Яка функція називається частково рекурсивною?
5. Яка гіпотеза є історично першою щодо зв'язків між класами інтуїтивних та точних алгоритмів?
6. Подайте за допомогою діаграм Ейлера - Венна співвідношення між класами обчислювальних та частково-рекурсивних функцій.

- Задачі для самостійного розв'язування

Задачі для самостійного розв'язування

1. Знайти за допомогою оператора регулярної суперпозиції функцію *h(x,y)*, якщо

F(x,y,z) = 2*x*+*y*+3*z* -1;
$f_1(x,y)$ = x+y; $f_2(x,y)$ = *xy*; $f_3(x,y)$ = 3*x*+2*y*+1.

2. Показати, що функцію

$$h(x,y) = x \dot{-} y = \begin{cases} x-y, & якщо \ x \geq y \\ 0, & якщо \quad x < y \end{cases}$$

можна одержати з базисних функцій за допомогою оператора примітивної рекурсії.

3. За допомогою оператора мінімізації знайти функцію *h(x,y)*, якщо *f(x,y,z)=x+y+z*.

4. Знайти функції, які одержуються із базисних функцій за допомогою одноразового застосування оператора регулярної суперпозиції.

5. За означенням примітивно-рекурсивної функції та процедурою породження примітивно-рекурсивних функцій покажіть, що симетрична різниця двох натуральних чисел *Sub(a,b)* = |*a*-*b*| можна розглядати як примітивно-рекурсивну функцію.

Коментарі. Основні відомості цього розділу містяться в [5,6,11,17,21,22]. У розділі не представлена відома формалізація, що використовується в теорії обчислювальності і яка має назву лямбда - числення. Це числення було запропоноване А. Черчем та С. Кліні в 1930-ті роки як спроба розробити базис математики на основі функцій. Однак через знайдені парадокси ця система функцій виявилася суперечливою. З лямбда - численням можна ознайомитися, наприклад, у [8,9,12].

Тема 12

- Алгоритмічні моделі
  - 12.1. Основні визначення
  - 12.2. Машини Тюрінга
  - 12.3. Нормальні алгоритми Маркова

Алгоритмічні моделі

12.1. Основні визначення

Точне визначення класу рекурсивних функцій разом із тезою Черча надає один із можливих варіантів уточнення поняття алгоритму. Проте це уточнення в цілому не пряме, оскільки поняття обчислювальної функції є вторинним по відношенню до поняття алгоритму. Виникає питання, чи не можна уточнити безпосередньо саме поняття алгоритму, а вже потім за його допомогою визначити точно клас обчислювальних функцій?

Основна ідея полягала в тому, що алгоритмічні процеси – це процеси, які може здійснювати відповідним чином влаштована "машина". У зв'язку з розвитком сучасної обчислювальної техніки така алгоритмічна модель становить особливий інтерес, тому що в ній поняття алгоритму базується на командно-адресному принципі. Для наукового аналізу обчислювальних процесів, які можуть бути реалізовані машиною, бажано було знайти просту за своєю логічною структурою схему алгоритмічної машини, яка може бути предметом точної математичної теорії.

Відповідно до цієї думки вчені створили в точних математичних термінах "машинні моделі" (математичні абстракції), на яких вдалось імітувати дію алгоритмів. Це було зроблено Постом і Тюрінгом у 1936-1937 рр. незалежно один від одного і майже одночасно із працями Черча. Пізніше, у 50-х роках двадцятого століття, до їх приєднався А. А. Марков. Алгоритмічні моделі цих вчених є претендентами на право бути основними формалізаціями поняття "алгоритм". Це означає, що вони повинні бути універсальними, тобто в їх рамках можна описати будь-який алгоритм. Це дійсно так, тому що, по-перше, можна довести зведення однієї моделі до іншої, тобто будь-який алгоритм, описаний засобами однієї моделі, може бути описаний засобами іншої; по-друге, завдяки взаємному зведенню моделей у теорії алгоритмів удалося виробити інваріантну по відношенню до моделей систему понять, що дозволяє вести мову про властивості алгоритмів незалежно від того, яка формалізація обрана. Ця система понять базується на понятті обчислювальної функції, тобто для обчислення якої існує алгоритм.

Було доведено, що клас функцій, обчислюваних на цих машинах, збігається із класом рекурсивних функцій. Таким чином, це є ще одним підтвердженням тези Черча.

12.2. Машини Тюрінга

Перший важливий і достатньо широкий клас алгоритмів був уведений

А. Тюрінгом та Е. Постом у 1936-1937 рр. Алгоритми цього класу здійснюються особливими машинами, що називаються зараз машинами Тюрінга-Поста, або просто машинами Тюрінга. Машини Тюрінга копіюють в істотних рисах роботу людини і часто розглядаються як математичні моделі для вивчення функціонування людського мозку.

Машина Тюрінга – це математична модель, яка породжує обчислювальні процеси. Ідея машини Тюрінга базується на загальному аналізі процесів обчислення значень функцій обчислювачем. При цьому спостерігається основна гіпотеза теорії алгоритмів (теза Тюрінга) : будь-який алгоритм може бути реалізований у машині Тюрінга. Теза має сенс у тому, що кожна функція, для якої складений алгоритм знаходження її значень, представима машиною Тюрінга, тобто є обчислювальною.

Зафіксуємо два скінченних алфавіти – $A = \{a_0,a_1,\ldots,a_n\}$, де $n \geq 0$, і $Q = \{q_0,q_1,\ldots,q_m\}$, де $m \geq 1$. При цьому *A* будемо називати зовнішнім алфавітом, а *Q* – внутрішнім алфавітом, або алфавітом станів. Додатково поставимо вимогу, щоб $A \cap Q = \emptyset$ і символи $\rightarrow$, *L*, *R* не належать $A \cup Q$. Один символ з *A* називають порожнім, зазвичай його позначають Λ . Усі інші букви з *A* називають непорожніми.

Візьмемо об'єднаний алфавіт $B = A \cup Q \cup \{\rightarrow , L, R\}$. Виділимо серед слів алфавіту *B* команди, які представляють слова одного з трьох видів:

1. $q_ia_j \rightarrow q_ka_i$; 2) $q_ia_j \rightarrow q_kL$; 3) $q_ia_j \rightarrow q_kR$;

де $1 \leq i \leq m$, $0 \leq j \leq n$, $0 \leq k \leq m$, $0 \leq l \leq n$.

Ці команди читаються відповідно так:

1. перебуваючи у стані q_i і спостерігаючи символ a_j, перейти у стан q_k і написати символ a_i;
2. перебуваючи в стані q_i і спостерігаючи символ a_j, перейти у стан q_k і перемістится вліво на один символ ;
3. перебуваючи в стані *i* спостерігаючи символ, перейти у стан q_k і перемістится вправо на один символ .

Скінченна послідовність команд становить програму *P* із зовнішнім алфавітом *A* і внутрішнім алфавітом *Q*. У кожній команді програми виділимо її ліву частину – це підслово до символу "→" і праву частину – підслово після символу "→".

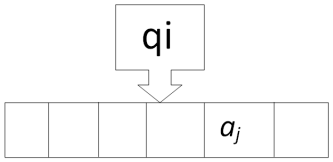
Означення 12.2.1. Машиною Тюрінга (МТ) називають упорядковану шістку { *A, Q, a_0, q_0, q_1, P* }, яка задовольняє такі умови:

1. множини *A* і *Q* скінченні, не перетинаються і не містять символів → , *L*, *R*;
2. $a_0 \in A$, $q_0 \in Q$, $q_1 \in Q$. При цьому a_0 називається символом порожньої комірки, q_1 - початковий стан машини, q_0 - стан, у якому машина зупиняється;
3. *P* – програма із зовнішнім алфавітом *A* і внутрішнім алфавітом *Q*, причому

а) програма не містить двох різних команд з однаковими лівими частинами;

б) будь-яка з команд не починається символом q_0.

Отже, МТ – це алгоритмічна модель, а не фізична машина. Проте для кращого розуміння роботи цієї машини її можна уявити у вигляді автоматичного пристрою (див. рисунок).



Пристрій оглядає стрічку, поділену на комірки, яку можна уявити необмеженою в обох напрямках. У кожний конкретний момент часу пристрій знаходиться в деякому стані q_i, спостерігаючи символ a_j, записаний в одній із комірок. Залежно від того, в якому стані перебуває пристрій ті який символ він спостерігає, пристрій здійснює дію, яка може полягати в тому, що пристрій на місце a_j запише a_i, а сам перейде в новий стан q_k *(виконується команда першого виду)*, або переміститься вліво на одну комірку і змінить свій стан на q_k *(виконується команда другого виду)*, або переміститься вправо на одну комірку і змінить стан на q_k *(виконується команда третього виду)*.

Охарактеризуємо тепер роботу машини Тюрінга більш детально.

Означення 12.2.2. Машинним словом, або конфігурацією, називається будь-яке слово в алфавіті $A \cup \{q_i\}$, для якого $q_i \in Q$, причому символ q_iвходить у це слово один раз і не на останньому місці.

Приклад 12.2.1 Нехай задані алфавіти $A=\{a_0, 1\}$, $Q=\{q_0,q_1,q_2\}$.

Тоді слова $a_01a_0a_0q_1a_0$, a_01q_21, q_0111a_0 є конфігураціями, а слова a_0111q_1 , $q_11a_0q_21a_0$, $a_0q_1q_11$ – до конфігурацій не належать.

У процесі роботи МТ здійснюється перехід від одного машинного слова до іншого. При цьому відбувається певне перетворення слів алфавіту A. Будемо говорити, що МТ перетворює машинне слово α в машинне слово γ

(записують $МТ(\alpha)=\gamma$), якщо існує скінченна послідовність слів $\gamma_0, \gamma_1, \dots, \gamma_n$, для яких виконуються умови :

1. $\gamma_0 = \alpha$, $\gamma_n = \gamma$;
2. слово γ_{i+1} одержане з γ_i дією МТ в один крок за визначеним правилом.

МТ може перетворити задане слово тільки в одне слово. Якщо МТ не перетворює слово α ні в яке слово, то говорять, що ця машина не застосовна до слова α, або значення $МТ(\alpha)$ не визначене.

Приклад 12.2.2. Знайти результат застосування МТ із зовнішнім алфавітом $\{a_0, 1,2\}$, внутрішнім алфавітом $\{q_0,q_1,q_2,q_3,q_4,q_5\}$ і програмою $q_11\to q_2R$, $q_22\to q_21,q_21\to q_3R$, $q_32\to q_3R$, $q_31\to q_42$, $q_42\to q_4R$, $q_4a_0\to q_5L$, $q_52\to q_5L$, $q_51\to q_5L$, $q_5a_0\to q_0R$ до слова 1221.

Розв'язання. Виходячи з початкової конфігурації $q_1$1221, знайдемо кінцеву конфігурацію, якщо вона існує. Маємо перетворення МТ за заданою програмою:

$q_1$1221→ 1 $q_2$221→1 $q_2$121→ 1 1 $q_3$21→112$q_3$1→112$q_4$2→1122q_4 a_0→112$q_5$2→11 $q_5$22→1 $q_5$122→ $q_5$1122→ q_5 $a_0$1122→ $q_0$1122.

Отже, слово 1221 МТ перетворює в слово 1122.

Значно складнішою, ніж задача застосування конкретних машин до конкретних слів, є задача створення МТ із наперед заданими властивостями.

Важливим класом МТ є машини, які обчислюють значення тих чи інших числових функцій для будь-якого набору значень аргументів із області визначення функції.

Нагадаємо, що числовою називають функцію $f(x_1,x_2,\dots,x_n)$, значеннями якої та значеннями її аргументів є невід'ємні цілі числа. Розглянемо часткові числові функції, визначені загалом не для всіх значень аргументів.

Означення 12.2.3. Функція називається обчислювальною за Тюрінгом, якщо існує машина Тюрінга, яка обчислює значення цієї функції для будь-якого набору значень аргументів із області визначення функції і не застосовна до наборів значень аргументів, що не входять до області визначення цієї функції.

Для обчислення числових функцій на МТ часто використовують спеціальне кодування чисел. Наприклад, невід'ємне ціле число m можна позначити словом (задати набором) з $(m+1)$ одиниць, або скорочено 1^{m+1}. Тобто 0 задають як 1, 1 – як 11, 2 – як 111 тощо. Символом порожньої комірки буде слово 0. Тоді для побудови МТ буде достатньо зовнішнього алфавіту $A=\{0,1\}$.

12.3. Нормальні алгоритми Маркова

Розглянемо ще один підхід до уточнення поняття алгоритму, запропонований російським математиком А. А. Марковим на початку 1950-х років. Досвід вивчення і застосування математики показує, що всі відомі алгоритми можна розбити на найпростіші кроки — елементарні операції. Як елементарну операцію, на базі якої побудовано нормальні алгоритми, А. А. Марков запропонував застосувати підстановку одного слова замість іншого.

Нормальні алгоритми Маркова (НАМ) перетворюють рядки, які задані у будь-якому скінченному алфавіті , у рядки у тому самому алфавіті.

Перейдемо до точних означень. **Алфавітом** будемо називати не порожню скінченну множину ***E*** деяких символів. Символи "→ " та "• " не повинні належати алфавіту ***E***. Елементи алфавіту називатимемо також буквами. ***Слово*** в алфавіті ***E*** — це скінченна, або порожня, послідовність його букв. Порожнє слово позначимо Λ . Множину всіх слів в алфавіті ***E*** позначимо через ***E****. Нехай ***P, Q*** $\in$ ***E****, тоді вирази ***P*** →***Q, P***→ •***Q*** називаються відповідно формулами простої та заключної підстановки. При цьому перша з них означає, що замість ***P*** потрібно вставити слово ***Q*** та перейти до наступної підстановки, а в другій формулі після підстановки процес закінчується.

Нехай ***P***→ ***(•)Q*** означає будь-яку з формул підстановки (просту чи заключну). **Нормальний алгоритм в алфавіті *E*** вважається заданим, якщо задана скінченна схема (таблиця) формул підстановок слів алфавіту ***E***

$$S = \begin{cases} P_1 \to (\bullet) Q_1 \\ P_2 \to (\bullet) Q_2 \\ \dots\dots\dots\dots \\ P_n \to (\bullet) Q_n \end{cases}$$

(12.3.1)

1. **Означення 12.3.1. Нормальним алгоритмом Маркова (НАМ) в алфавіті E** називають пару**{ *E,S*}**, яка складається з алфавіту ***E*** та схеми ***S*** в цьому алфавіті.
2. Роботу нормального алгоритму Маркова можна описати у такий спосіб. Нехай задане слово $\alpha \in$ ***E****. Знаходимо першу в схемі ***S*** таку формулу підстановки $\alpha_i \to (\bullet)\beta_i$, що α_iє підсловом α. Підставляємо в слово α слово β_i замість першого входження α_iв α. Нехай γ_i – результат цієї підстановки. Якщо формула підстановки виявилася заключною, тобто $\alpha_i \to \bullet\beta_i$, то робота алгоритму закінчується і $A(\alpha) = \gamma_i$. Якщо формула підстановки виявилась простою, тобто $\alpha_i \to \beta_i$, то до слова γ_i застосовуємо той самий пошук, який застосовувався до слова α і так далі. Якщо врешті-решт одержимо таке слово γ_i, що жодне із слів $\alpha_1, \alpha_2, \dots \alpha_n$ не входить в γ_i як підслово, то робота алгоритму закінчується і $A(\alpha) = \gamma_i$. Якщо описаний процес не закінчується ніколи, то говоримо, що алгоритм А не застосовний до слова α.
3. **Приклад 12.3.1.** Розглянемо алгоритм який заданий у алфавіті $E=\{a,b\}$ схемою підстановок :

$$S = \begin{cases} ab \to bb, \\ aaa \to \bullet a, \\ ba \to aa. \end{cases}$$

(12.3.1)

Застосуємо його дію до слова $X = abba$.

Першою підстановкою зі слова "<u>ab</u>ba" одержимо слово X_1 = bbba. Далі перша і друга підстановка до слова X_1не діє, за третьою підстановкою зі слова "bb<u>ba</u>" одержимо слово X_2= bb<u>ba</u>a, до якого застосовна лише третя підстановка, яка перетворює його у слово X_3=baaa. До слова X_3 застосовні друга і третя підстановки, причому спочатку повинна виконуватися друга підстановка, але оскільки вона є заключною, то після її дії процес перетворення слів закінчується. Отже, слово «b<u>aaa</u>» переходить в слово X_4 = =ba. Таким чином, $A(abba) = ba$.

Означення нормального алгоритму Маркова, на перший погляд, не свідчить про якусь універсальність цього поняття. Проте виявляється, що клас нормальних алгоритмів має досить широкі можливості, зокрема під час обчислення значень функцій.

Означення 12.3.2. *Функція називається нормально обчислювальною, якщо існує такий*

нормальний алгоритм, який обчислює значення цієї функції для будь-якого набору значень аргументів із області визначення функції і не застосовний до наборів значень аргументів, що не входять до області визначення цієї функції.

Постає питання про клас функцій, які можна обчислювати за допомогою нормальних алгоритмів. З цього приводу А. А. Марковим була сформульована гіпотеза, що одержала назву принципу нормалізації Маркова.

Принцип нормалізації Маркова. Клас нормально обчислювальних функцій збігається з класом обчислювальних функцій.

Аналогічно тому, як не можна довести гіпотези Черча і Тюринга, неможливо довести і принцип нормалізації Маркова.

Запитання для самоперевірки:

1. Що називається машиною Тюрінга?
2. Що називається нормальним алгоритмом Маркова?
3. Яка функція називається обчислювальною за Тюрінгом?
4. Сформулюйте гіпотезу Тюрінга.
5. Яка функція називається нормально обчислювальною?
6. Сформулюйте принцип нормалізації Маркова.
7. Сформулюйте теорему про зв'язки між собою всіх формальних уточнень поняття алгоритму.

- Задачі для самостійного розв'язування

Задачі для самостійного розв'язування

1. Знайти результат застосування машини Тюрінга із зовнішнім алфавітом $A=\{a_0,1\}$, внутрішнім алфавітом $Q=\{q_0,q_1,q_2,q_3,q_4,q_5,q_6,q_7\}$ і програмою

$q_1a_0\to q_4R$, $q_11\to q_2L$, $q_2a_0\to q_6R$, $q_21\to q_3L$, $q_3a_0\to q_6R$, $q_31\to q_1L$,
$q_4a_0\to q_01$, $q_41\to q_5a_0$, $q_5a_0\to q_4R$, $q_51\to q_5a_0$, $q_6a_0\to q_0a_0$, $q_61\to q_7a_0$,
$q_7a_0\to q_6R$, $q_71\to q_7a_0$ до слів а) 11111; в) 111111; с) $11a_0111$.

2. Побудувати машини Тюрінга та нормальні алгоритми Маркова, які обчислюють функції: а) $f(x)=0$; b) $f(x)=1$; c) $f(x)=x$; d) $f(x,y)=x+y$.

3. Нехай $E=\{a,b,c\}$ та задано таблицю формул підстановок

$$S=\begin{cases}\beta a\to a\beta & (1)\\ \beta b\to b\beta & (2)\\ \beta c\to c\beta & (3)\\ \beta\to\bullet a & (4)\\ \Lambda\to\beta & (5)\end{cases}$$

До якого результату приведе дія цього нормального алгоритму над словом *abb*.

4. Винайдіть такий нормальний алгоритм Маркова, за допомогою якого можна перетворити слово «муха» у слово «слон».

Коментарі. Таким чином, ми ознайомились із трьома теоріями, кожна з яких уточнює поняття алгоритму. Докладніше матеріал цього розділу можна знайти в [6,11,16,21,22]. Виникає питання про зв'язки цих теорій між собою. Відповідь на це запитання дає така теорема. Наступні класи числових функцій збігаються:

1) клас усіх частково рекурсивних функцій; 2) клас усіх функцій, обчислювальних за Тюрінгом; 3) клас усіх нормально обчислювальних функцій. Вона означає, що теорії частково рекурсивних функцій, машин Тюрінга і нормальних алгоритмів Маркова рівносильні між собою.

Тема 13

- Нумерації алгоритмів
  - 13.1. Попередні відомості
  - 13.1.1. Нумерація за Гьоделем
  - 13.1.2. Головні універсальні функції та множини
  - 13.1.3. Канторові нумерації кортежів натуральних чисел
  - 13.1.4. Нумерації Кліні та Поста
  - 13.2 Нумерація машин Тюрінга та частково - рекурсивних функцій
  - 13.3. S-m-n- теорема
  - 13.4. М – зведення та властивості злічених множин
  - 13.5 Теореми Кліні про нерухому точку

Нумерації алгоритмів

Теорію нумерації можна вважати новою самостійною галуззю математики, породженою теорією алгоритмів. Самостійність цієї нової галузі виправдовується наявністю в ній глибоких математичних результатів, як майже очевидних, так і абсолютно несподіваних і нетривіальних.

Узагалі нумерація - це відображення деякої підмножини множини натуральних чисел N на досліджуваний клас конструктивних об'єктів (формул, слів, матриць і т. п.). Під конструктивним об'єктом розуміють об'єкт, який може бути повністю описаний за допомогою деякої скінченної послідовності символів скінченного алфавіту.

Теорія нумерації - розділ теорії алгоритмів, у якому вивчаються властивості класів об'єктів, занумеровані за допомогою натуральних чисел. Надавши номери об'єктам (машинам Тюрінга, частково - рекурсивним функціям і т. д.), зможемо в багатьох випадках отримати глибокі наукові результати про ці об'єкти. Ідея використання нумерації об'єктів для отримання різних тверджень про ці об'єкти належить К. Гьоделю.

Нумерація алгоритмів відіграє важливу роль у дослідженні та аналізі самих алгоритмів. Оскільки будь-який алгоритм можна задати у вигляді скінченного слова (подати у вигляді скінченної послідовності символів деякого алфавіту), а множина всіх скінченних слів у скінченному алфавіті зліченна, то множина всіх алгоритмів також зліченна. Це означає існування взаємно однозначного відображення між множиною натуральних чисел та множиною алгоритмів, тобто можливість присвоїти кожному алгоритму номер.

Нумерація алгоритмів є водночас і нумерацією всіх алгоритмічно обчислюваних функцій, причому будь-яка функція може мати нескінченну кількість номерів. Існування нумерації дозволяє працювати з алгоритмами так само, як із числами. Особливо корисна нумерація в дослідженні алгоритмів, що працюють з іншими алгоритмами.

Під час реалізації ідей нумерації виявилося, що деякі результати теорії алгоритмів є наслідком закономірностей теорії нумерацій.

13.1. Попередні відомості

Узагалі теорія нумерації призначена для вивчення загальних властивостей класів об'єктів, занумерованих за допомогою будь-яких конструктивних об'єктів. У ролі конструктивних об'єктів найчастіше виступають натуральні числа. Ними можуть також бути впорядковані пари або скінченні послідовності натуральних чисел, раціональні числа, алгоритми, але ірраціональні числа, нескінченні підмножини натурального ряду, функції $f:N\to N$ не є такими.

Ідея нумерації об'єктів нечислової природи натуральними числами та перенесення змістовних тверджень про ці об'єкти у сферу формальної арифметики натуральних чисел уперше були використані К. Гьоделем під час доведення теореми про неповноту арифметики Пеано. Надалі ці ідеї були використані для нумерації фундаментальних об'єктів теорії алгоритмів, таких як машина Тюрінга, частково - рекурсивних функцій та інше. Нумеризація цих класів об'єктів дозволила в багатьох випадках більш чітко з'ясувати природу цих об'єктів, виявити ряд їх важливих і нових властивостей.

До появи ЕОМ алгоритми у сферах своїх застосувань вважалися різними за природою. З ерою комп'ютеризації прийшло усвідомлення того, що будь-яку інформацію можна закодувати (читай подати) числами, наприклад застосуванням в ЕОМ двійкового алфавіту {0,1}, й переконання, що виконання будь-якого алгоритму зводиться до дій над числами на базі арифметики та елементарної логіки. Тому перші наукові праці з теорії нумеризацій, що з'явилися до появи ЕОМ, викликають захоплення.

Нехай A - довільна непорожня скінченна або зліченна множина. *Нумерацією множини A* називають сюр'єктивне функціональне відображення

ν: N →A. *Однозначною нумерацією множини A* називають ін'єктивне (а отже, бієктивне) відображення ν: N →A.

Якщо ν: N →A – нумерація, то для довільного a ∈ A знайдеться хоча б одне значення n ∈ N, для якого ν(n) = A. Кожне таке n будемо називати ν- номером елемента a.

Позначимо елемент a ∈ A з номером n через a_n. Тоді елементи нумерованої множини A можна подати у вигляді послідовності: a_0, a_1, a_2, ... У загальному випадку в цій послідовності можливі повторення, тобто рівності

$a_i = a_j$, при i ≠ j. Якщо дана однозначна нумерація, то таких повторень немає.

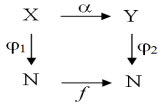
Нехай A-множина, що складається з конструктивних об'єктів. *Множина A називається ефективно зліченною множиною, якщо існує однозначна нумерація ν: N → A така, що виконані дві умови:*

- Існує алгоритм x_1, що обчислює за елементом a ≠ A його номер ν(a) =N.
- Існує алгоритм x_2, що встановлює за номером n ≠ N елемент a ≠ A з даним номером.

Нумерація ν називається в цьому випадку ґьоделівською нумерацією, умови можемо виразити в такому вигляді: функції ν: N→A і ν^{-1}: A→N обчислювальні. Таким чином, φ: N →A - ефективна нумерація ⇔ φ^{-1}: A→N - кодування A на N.

Нехай α - довільний алгоритм, що переробляє об'єкти множини X в об'єкти множини Y. Нехай φ_1 і φ_2 - ґьоделівські нумерації множин X і Y відповідно. Алгоритм α можна розглядати як процедуру обчислення функції

f: N → N



Якщо відображення φ: N → A є нумерацією множини A і φ (n) = a, то натуральне число n називається номером елемента a ∈ A при нумерації φ.

Нижче введемо нумерації для об'єктів , що вивчаються в теорії алгоритмів. Синонімом терміна «нумерація» є термін «кодування» - спосіб задання об'єктів натуральними числами. По суті, кодування - частковий випадок нумерацій.

13.1.1. Нумерація за Ґьоделем

К. Ґьодель увів важливий метод арифметизації, в основі якого лежить однозначна нумерація об'єктів формальної системи (символів, термів, формул, доведень і т. д.) деякими натуральними числами. Ці числа були названі ґьоделівськими номерами цих об'єктів. Кожному формальному символу, що входить в алфавіт системи, ставиться у відповідність деяке число, відношення та операції, визначені на словах, переходять у відношення та операції, визначені на номерах.

Нехай *K* – конструктивний простір – множина конструктивних об'єктів однакового виду. Очевидно, що *K* не більше ніж зліченна множина (елементам K відповідає їх опис, що складається із слів скінченного алфавіту, а множина слів скінченного алфавіту є зліченною множиною).

Означення. *Нумерація ν: N→K називається ґьоделівською, якщо існує алгоритм, який дозволяє за записом будь-якого натурального числа одержати опис об'єкта, номером якого є це число.*

Теорема 13.1. *Нехай K – конструктивний простір (не дорівнює Ø). Тоді :*

1. *існує ґьоделівська нумерація простору K;*
2. *будь-яка ґьоделівська нумерація простору K розв'язана;*
3. *будь-які дві ґьоделівські нумерації простору K еквівалентні;*
4. *будь-яка нумерація простору K, еквівалентна ґьоделівській, сама є ґьоделівською.*

Доведення наведеної теореми є предметом завдань для самостійного розв'язування.

Основна ідея, на якій базується нумерація за Ґьоделем, полягає в тому, що будь - яке число N можна розкласти на прості множники :

$$N = 2^{a_0} \cdot 3^{a_1} \cdot 5^{a_2} \cdot \ldots p_n^{a_n}, \quad (13.1)$$

що встановлює відповідність n - кам $\{a_1, a_2, \ldots, a_n\}$ номерів з N.

Нехай задано алфавіт зліченної множини A = $\{a_0, a_1, a_2, \ldots\}$. Визначимо ґьоделівські номери для кожної літери $g(a_i) = 2i + 3$, i ∈ N. Для кожного слова P= $a_{i0}\ a_{i1} \ldots a_{ik}$ визначимо його ґьоделівський номер:

$g(P) = 2^{g(a_{i0})} 3^{g(a_{i1})} \ldots p_k^{a_{ik}}$, де p_k – k - е просте число. Послідовність формул (вона може становити і будь-які доведення) $F_1, F_2, \ldots, F_m$ одержить ґьоделівський номер $2^{G_1} \cdot 3^{G_2} \cdot \ldots \cdot p_m^{G_m}$, де p_m – m –е просте число, $G_1, G_2, \ldots G_m$ – ґьоделевські $F_1, F_2, \ldots F_m$.

Відмінність нумерації за Ґьоделем від інших нумерацій в тому, що множина номерів має «дірки», які частково заповнюються при збільшенні n. Цей «недолік» обертається перевагами під час нумерації виразів, що складаються не тільки з наборів цифр, а будь-якого тексту в будь-якому алфавіті.

Приклад 13.1. *Занумерувати формулу ∀x(xxS(0) = x) у системі елементарної арифметики.*

Розв'язання. Існують різні способи побудови ґьоделівської нумерації для системи елементарної арифметики (*EA*). Виберемо 10 основних символів алфавіту теорії *EA*. Задамо відповідність цих символів натуральним числам за таблицею.

0	S	(	)	+	x	⊃	¬	∀	=
1	2	3	4	5	6	7	8	9	10

Предметним змінним зіставимо прості числа, більші за 10 (наприклад x одержить номер 11, y – 12 і т. д.). Інші логічні зв'язки можна подати через ⊃, ¬, ∀. Наприклад, A˅B ≡ ¬A ⊃ B, ∃xA(x) ≡ ¬∀x¬A(x).

Формула, що потребує нумерації складається з 12 символів, причому деякі входять в неї не один раз. Візьмемо перші 12 простих чисел, зведемо кожне з них у степінь, що дорівнює номеру відповідного символа й перемножимо одержані числа. Знайдене таким чином число й буде ґьоделівським номером цієї формули :

$2^9.3^{11}.5^3.7^{11}.11^6.13^2.17^3.19^1.23^4.29^{10}.31^{11}.37^4$
=3512466963791134964962551783462053411408361516343794496506561501312862272000.

Отже, кожній формулі або послідовності формул можна поставити у відповідність єдине натуральне число. Не будь-яке число є ґьоделівським номером, але будь-який ґьоделівський номер однозначно визначає відповідний вираз.

13.1.2. Головні універсальні функції та множини

Означення. *Функція U двох натуральних аргументів є універсальною для класу обчислюваних функцій одного аргументу, якщо для кожного n функція*

$U_n : x \to U(n, x)$ *(13.2)*

(«переріз» функції U при фіксованому n) є обчислюваною і якщо всі обчислювані функції (одного аргументу) трапляються серед U_n. (Нагадаємо, що ні функція U, ні обчислювана функція одного аргументу не мають бути обов'язково усюди визначеними).

Аналогічне визначення можна дати і для інших класів функцій (одного аргументу): наприклад, функція U двох аргументів буде універсальною для класу всіх всюди визначених обчислюваних функцій одного аргументу, якщо її перерізи U_n є всюди визначеними обчислюваними функціями одного аргументу і вичерпують усі такі функції.

Ключовим моментом у цьому сенсі є можливість ефективної нумерації обчислюваних функцій, $f_1(x), f_2(x), \ldots, f_n(x), \ldots,$ (13.3)

яка дозволяє вважати двомісну функцію

$U(n, x) = f_n(x)$ (13.4)

універсальною.

Теорема 13.2. *Існує обчислювана функція двох аргументів, що є універсальною функцією для класу обчислюваних функцій одного аргументу.*

Доведення. Запишемо всі програми, що обчислюють функції одного аргументу, в послідовність $p_0, p_1, \ldots$(наприклад, у порядку зростання їх довжини). Покладемо значення U (i; x) однаковому результату роботи i-ї програми при вході x. Тоді функція U і буде шуканою обчислюваною універсальною функцією. Переріз U_i буде обчислюваною функцією, що обчислюється програмою p_i.

Алгоритм, який обчислює саму функцію U, є, по суті, інтерпретатором для використовуваної

мови програмування (він застосовує перший аргумент до другого, якщо ототожнити програму і її номер).

Означення. *Нумерація і функція U(n, x) називаються гьоделівськими, якщо існує всюди визначена обчислювана функція s(n) така, що для будь-якої двомісної функції f(n, x) справедливе*

$f(n, x) = U(s(n), x).$ *(13.5)*

Нехай F – деякий клас функцій від k - змінних. Функцію $U(n, x_1,\ldots,x_k)$ від змінних називають універсальною для класу F, якщо виконуються умови:

а) для будь-якого $n \in N$

$f_n(x_1, \ldots, x_k) = U(n,x_1,\ldots,x_k) \in F;$

б) для будь-якого $f(x_1, \ldots, x_k) \in F$ існує $n \in N$ таке, що

$f(x_1,\ldots, x_k) = U(n,x_1,\ldots,x_k).$

Для множин використовується аналогічна термінологія.

Означення. *Множину* $W \subset N \times N$ *називають універсальною для деякого класу множин натуральних чисел, якщо всі перерізи*

$W_n = \{x \mid (n, x) \in W\}$

множини W належать цьому класу й інших множин у класі немає.

Іншими словами, це означає існування множини пар $W = \{(n, x) \in W_n\}$, що в перерізі при фіксованому n визначають W_n.

Теорема 13.3. *Область визначення універсальної функції U(n, x), а також множини пар {(n, U(n, x))} є гьоделівськими універсальними множинами.*

13.1.3. Канторові нумерації кортежів натуральних чисел

Уведемо однозначні ефективні нумерації пар та n-ок натуральних чисел, які називаються канторовими нумераціями.

Нехай $A = \{(x, y) \mid x, y \in N\}$. Усі пари натуральних чисел розташуємо в послідовність так: пара (x, y) передує парі (u, v) ⇔ x + y< u + v або (x+y= u + v та x<y).

$$\underbrace{№0\ (0,0),}_{\tilde{B}_0}\ \underbrace{№1\ №2\ (0,1),(1,0),}_{\tilde{B}_1}\ \underbrace{№3\ №4\ №5\ (0,2),(1,1),(2,0),}_{\tilde{B}_2}\ \underbrace{№6\ №7\ №8\ №9\ (0,3),(1,2),(2,1),(3,0)}_{\tilde{B}_3},\ldots$$

(13.6)

Таку послідовність можна подати діагональним варіантом :

$$\begin{array}{lllll} (0;0) \swarrow & (0;1) \swarrow & (0;2) \swarrow & (0;3) & \ldots \\ (1;0) \swarrow & (1;1) \swarrow & (1;2) \swarrow & (1;3) & \ldots \\ (2;0) \swarrow & (2;1) \swarrow & (2;2) \swarrow & (2;3) & \ldots \\ (3;0) \swarrow & (3;1) \swarrow & (3;2) \swarrow & (3;3) & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{array}$$

Пари (x; y) в даній послідовності виписані за блоками B_0, B_1, ... так, що в кожному блоці B_S поміщені пари (x; y) із заданою сумою: s = x + y.

Усередині блоку пари впорядковані за зростанням першої компоненти:

$B_S = \{(0; x+y), (1; x+y-1), \ldots, (x; y), \ldots, (x+y), 0)\}$.

З наведених міркувань випливає: кожен елемент множини A має номер (номери починаються з нуля); для довільного натурального числа можна вказати відповідну пару (однозначно).

Номер пари (x, y) в такій послідовності позначають C(x, y) і називають канторовим номером пари (x, y) : C(1;0)=2, C(2;0)=5, C(0;3)=6, C(2;1)=8 і т. д.

Неважко переконатися, що $\mathbf{C(x, y) = ((x+y)^2 +3x +y) / 2}$

(13.7)

Функція C(x, y) задає бієкцію NXN→N. За допомогою цієї нумерації можна отримати нумерацію кортежів (трійок, четвірок і т. д.) натуральних чисел.

З формули (13.7) також випливає, що існують функції r, l такі, що належать класу загальнорекурсивних функцій і однозначно визначаються

$C(x, y) = n, l(C(x, y)) = x, r(C(x, y)) = y,$

(13.8)

для будь-яких $x, y \in N$.

Зауважимо, що важливим є навіть не вигляд функцій l(n) , r(n) , а той факт, що вони є ефективно обчислювальні.

Використовуючи канторову функцію (13.7), можна визначити послідовність загальнорекурсивних функцій $C_1, C_2, \ldots, C_n$ таку, що C_n– n - місна функція, що здійснює бієкцію $N^n \to N$.

Канторовими нумерувальними функціями називаються такі функції:

$$\begin{cases} C_1(x_1) = x_1 \\ C_{n+1}(x_1, x_2, \ldots, x_n) = C_n(x_1, \ldots, x_{n-1}, C(x_n, x_{n+1})) \end{cases},$$

(13.9)

в яких C(x,y) обчислюється за формулами (13.7). Наприклад,

$$C_1(x_1) = x_1$$
$$C_2(x_1, x_2) = C_1\big(C(x_1, x_2)\big) = C(x_1, x_2)$$
$$n + 1 = 2$$
$$n = 1$$
$$n - 1 = 0$$
$$C_3(x_1, x_2, x_3) = C_2\big(x_1, C(x_2, x_3)\big) = C\big(x_1, C(x_2, x_3)\big)$$
$$n + 1 = 3$$
$$n = 2$$
$$n - 1 = 1$$

Приклад 13.2. *Знайти номери впорядкованих трійок $C_3(0,1,1)$ та $C_3(1,1,1)$.*

Розв'язання.

$C_3(0, 1, 1) = C_2(0, C(1, 1)) = C(0, 4) = 10.$

$C(1, 1) = ((1+1)^2 + 3+1) / 2 = 4; C(0, 4) = ((0+4)^2 +0 + 4) / 2 = 10.$

$C_3(1, 1, 1) = C_2(1, C(1, 1)) = C(1, 4) = 16.$

$C(1, 1) = ((1+1)2 + 3+1) / 2 = 4; C(1, 4) = ((1+4)^2 +3 + 4) / 2 = 16.$

13.1.4. Нумерації Кліні та Поста

Нумерації Кліні та Поста використовуються, в основному, в теорії рекурсивних функцій. Наведемо спосіб їх подання та означення.

Введемо стандартні позначення :

$[x, y] = C(l(x), C(r(x),y)),$

(13.10)

де C(x, y) – канторів номер пари (x, y); l(x) та r(x) – відповідно ліве та праве числа пари C з номером x. З тотожностей

$C(l(x), r(x)) = x, l(C(x, y)) = x, r(C(x, y)) = y$ та формули (13.6) випливає, що якщо [x, y] = n, то

$x = C(l(n), l(r(n))), y = r(r(n)).$

(13.11)

Уведемо позначення $[x]_{21} = C(l(x), l(r(x)))$, $[x]_{22} = r(r(x))$ і матимемо тотожності $[[x]_{21}, [x]_{22}] = x$, $[[x, y]]_{21} = x$, $[[x, y]]_{22} = y$, що вказують на взаємно однозначну нумерацію пар натуральних чисел. Отже, функція [x, y], також як і функція Кантора C(x, y), здійснює взаємно однозначну нумерацію пар натуральних чисел.

Функція $K^2(x_0, x_1) = U(l(x_0), C(r(x_0), x_1))$ називається клініївською нумерувальною функцією. Ця функція визначається через головну універсальну функцію U(x, y), аргументи якої визначаються канторівською нумерувальну функцією C(x, y). Якщо $f(x) = K^2(m, x)$ для всіх x та деякого m, то це число *m назвемо клініївським номером функції f.*

Ставлячи кожному натуральному числу N у відповідність одномісну функцію $K^2(n, x)$, одержимо відображення множини натуральних чисел N на сукупність усіх одномісних частково - рекурсивних функцій $Ч_1$. Це відображення будемо називати нумерацією Кліні та позначати $\varkappa$ ($\varkappa$: $N \to Ч_1$).

Переваги функції $K^2(n, x)$ не набагато принциповіші, хоча вони спрощують формули, що задають номери композицій функцій, суми та інших комбінацій рекурсивного характеру. Головним є те, що клініївська нумерація є гьоделівською.

Нумерація Поста зустрічається в теорії перелічуваних множин. Якщо S – перелічувана множина, то постівський номер цієї множини є клініївським номером функції $f_n(x) = K^2(n, x)$,

(13.12)

множина значень якої збігається із S.

13.2 Нумерація машин Тюрінга та частково - рекурсивних функцій

Нехай задані зліченні множини символів A ={a_0, a_1, ..., a_i, ...} та Q = { q_0, q_1, ..., q_j, ...} }, що є зовнішнім алфавітом та алфавітом внутрішніх станів машини Тюрінга (МТ). При цьому a_0 інтерпретуємо як порожній символ (символ порожньої комірки), а q_0, q_1 – заключний та початковий стани МТ. Розглянемо множину {L, R, E, a_0, a_1, ..., a_i, ..., q_0, q_1, ..., q_j, ...} та проведемо її нумерацію за табл. 13.1.

Таблиця 13.1. Таблиця нумерації символів МТ.

	Символ	Код	Число нулів у коді
Символи зсуву	R	10	1
	L	100	2
	E	1000	3
Символи алфавіту стрічки	a_0	10000	4
	a_1	1000000	6
	...	...	...
	a_i	100...00	2i+4
	...	...	...
Символи алфавіту станів	q_0		5
	q_1		7
	...	...	...
	q_j	1000...000	2i+5
	...	...	...

Тоді команді I машини Тюрінга, що має вигляд $qa \to q'a'd$, ставиться у відповідність двійковий набір вигляду

$$Код(I)=Код(q)Код(a)Код(q')Код(a')Код(d) , \quad (13.13)$$

в якому коди всіх літер приписані один поряд з іншим.

Упорядкуємо команди МТ відповідно до лексикографічного порядку лівих частин команд : q_1a_0, q_1a_1, ..., q_1a_m,q_2a_0,...,q_2a_m,...,q_na_0,...,q_na_m .

Нехай задана відповідна послідовність команд. Тоді МТ поставимо у відповідність двійковий набір виду

$$Код(MT)=Код(I_1)Код(I_2)... Код(I_{n(m+1)}) , \quad (13.14)$$

який одержимо приписуванням один поряд з іншим кодів команд.

Приклад 13. 5. *Нехай задана МТ A={a_0,a_1}, Q={q_0,q_1} з програмою :*
$q_1a_0 \to q_0a_0E$; $q_1a_1 \to q_0a_1E$. Знайдемо код цієї машини.

Розв'язання. Використовуючи таблицю кодів (див. табл. 13.1), маємо
$Код(MT) = 10^7 10^4 10^5 10^4 10^3 10^7 10^6 10^5 10^6 10^3$.

Бачимо, що в даному випадку МТ переводить конфігурацію $q_1a_1^x$ у конфігурацію $q_0a_1^x$, то, подаючи натуральні числа у вигляді a_1^{n+1} , одержимо, що МТ обчислює, по суті, функцію f(x) = x.

Зрозуміло, що вказане кодування є алгоритмічною процедурою. Маючи код МТ, можна однозначно відновити множину її команд (програму МТ). Для цього потрібно виділити півслова, що починаються з одиниці з нулями до слідкуючої одиниці. П'ятірка таких півслів утворює одну з команд.

Будемо розглядати код МТ як двійковий запис натурального числа. Дане число й назвемо номером МТ. Оскільки всі коди починаються з одиниці, то різним кодам відповідають різні числа. Впорядкуємо МТ за зростанням чисел, подаючи їх кодами, і занумеруємо їх

$$MT_0, MT_1, ..., MT_n, ... \quad (13.15)$$

Таке упорядкування є ефективним у тому смислі, що існує алгоритм, який за номером n видає код (MT_n), та існує алгоритм, який, навпаки, за кодом MT_n видає n_{MT}.

Якщо через f(x) позначити одномісну функцію, яку обчислює машина Тюрінга MT_n, то в результаті одержимо нумерацію всіх одномісних частково - рекурсивних функцій :

$$f_0(x), f_1(x), ... ,f_n(x), ... \quad (13.16)$$

Аналогічно можна ввести нумерацію n - місних функцій.

13.3. S-*m*-*n*- теорема

S-m-n - теорема (також відома як лема про трансляцію, теорема параметрів, або теорема параметризації) належить до основних досягнень теорії алгоритмів (гьоделівської нумерації обчислюваних функцій). Уперше була доведена Кліні у 1943 р..

Крім теоретичного, ця теорема має також застосування в практиці програмування. Теорема означає, що для заданої мови програмування та додатних цілих m та n існує певний алгоритм, який оперує кодом програм із m+n - вільними змінними. Цей алгоритм ефективно прив'язує m даних значень до перших m вільних змінних у програмі й залишає інші вільними.

Найпростішою формою, до якої застосовна теорема, є функція двох аргументів. Суть теореми в такому. Допустимо, що *f(x,y)* – обчислювальна функція. Для кожного фіксованого значення *a* змінної *x* функція *f* породжує одномісну обчислювальну функцію $g_a(y) = f(a, y)$. З обчислювальності функції g_aвипливає існування такого індексу *b* , що $f(a, y) = f_b(y)$. Зясовується, що індекс *b* можна ефективно обчислювати за параметром *a*.

Теорема 13.3.1 (*s-m-n*-теорема, проста форма). *Нехай f(x,y) - обчислювальна функція. Існує всюди визначена обчислювальна функція S(x), така, що $f(x, y) = f_{s(x)}(y)$.*

Зауваження. Сформульовану теорему також називають *теоремою параметризації,* оскільки вона дозволяє за заданою обчислювальною функцією *f(x,y)* та фіксованим параметром *a* знайти гьоделівський номер *s(a)* програми, що обчислює функцію $f_{s(a)}(y) = f(a, y)$.

Маючи дану гьоделівську нумерацію φ рекурсивних функцій, існує примітивно-рекурсивна функція s двох аргументів із такою властивістю: для кожного номера p часткової обчислюваної функції f із двома аргументами, $\varphi_{s(p,x)}(y)$ та f(x,y) визначені для однакових комбінацій x- ів та y-ів однакові для тих комбінацій.

Теорема 13.3.2 (*s-m-n*-теорема). *Для довільних m, n*>1 *існує* (*m*+1) *-арна РФ* $s_n^m(z, x_1, ..., x_m)$ *така, що для всіх z, x_1, ..., $x_т$, y_1, ..., $y_п$ маємо* $\varphi_z^{m+n}(x_1, ..., x_т, y_1, ..., y_п) = \varphi_{s_n^m(z,x_1,...,x_m)}^n(y_1, ..., y_п)$.

Теорема 13.3.3 (*s-m-n*-теорема у спрощеній формі). *Для кожної ЧРФ f(x_1, ..., $x_т$, y_1, ..., $y_п$) існує РФ s(x_1, ..., x_m) така, що для всіх x_1, ..., $x_т$, y_1, ..., $y_п$* маємо $f(x_1, ..., x_т, y_1, ..., y_п) = \varphi_{s(z,x_1,...,x_m)}^n(y_1, ..., y_п)$.

При *т=п*=1 спрощена *s-m-n*-теорема збігається з 13.3.1.

Розглянемо приклади застосування *s-m-n*-теореми.

Приклад 13.6. Існує РФ *s*(*x*, *y*) така, що для всіх *x*, *y*, *z* $\in$ *N* маємо $\varphi_{s(x,y)}(z)=\varphi_x(z)+\varphi_y(z)$.

Розглянемо функцію $f(x, y, z)=\varphi_x(z)+\varphi_y(z)$. Функція *f* є ЧРФ, тому за *s-m-n*-теоремою існує РФ *s*(*x*, *y*) така, що для всіх *x*, *y*, *z* $\in$ *N* маємо $f(x, y, z)=\varphi_{s(x,y)}(z)=\varphi_x(z)+\varphi_y(z)$.

13.4. M – зведення та властивості зліченних множин

Надалі, в цьому розділі під множинами, якщо не обумовлено протилежне, ми будемо розуміти підмножини натурального ряду.

Означення. *Множина A m - зводиться до множини B (позначається $A \le_m B$), якщо існує обчислювана функція f: N → N така, що для будь-якого x ∈ N справедлива еквівалентність x ∈ A ⇔ f (x) ∈ B.*

Про функцію f говорять, що вона зводить множину A до множини B. Насправді потрібно було б говорити про зведення проблеми входження в множину A до проблеми входження в множину B.

Термін «m-зведення» історично походить з терміна «many-one-reduci-bility»; втім, як зазначає M. Cіпсер у своєму підручнику з теорії складності обчислень, замість цього краще говорити «mapping reducibility»(Зводиться за допомогою відображень), що також зберігає літеру m у позначенні.

Означення. *Множина A одно-зводиться до множини B (позначається $A \le_1 B$), якщо існує ін'єктивна обчислювана функція f: N → N така, що x ∈ A ⇔ f (x) ∈ B для всіх x ∈ N.*

Множина A називається m-еквівалентною (1-еквівалентною) множині B, якщо A $\le_m$ B і B $\le_m$ A (A $\le_1$ B і B $\le_1$ A).

Якщо множини A і B m-еквівалентні (1-еквівалентні), то ми пишемо A $\equiv_m$ B (A $\equiv_1$ B). Неважко перевірити, що відношення $\equiv_m$ є відношенням еквівалентності. Класи еквівалентності цього відношення називаються m –степенями. Можна стверджувати, що всі множини з одного m –степеня «рекурсивні» (або «нерекурсивні») однаковою мірою.

Нарешті, ми говоримо, що множини A і B зліченно ізоморфні (і пишемо A ≈ B), якщо існує обчислювана перестановка натурального ряду f така, що для будь-якого x ∈ N справедливе x ∈

$A \Leftrightarrow f(x) \in B$.

Інтуїтивно m-зведення - це спроба порівняти множини А та В.

Теорема 13.4.1. *Справедливі такі твердження.*

1. *Будь-яка рекурсивна множина m – зводиться до будь-якої множини.*
2. *Якщо множина m – зводиться до рекурсивної множини, то вона рекурсивна.*
3. *Якщо множина m – зводиться до рекурсивно - зліченної множини, то вона рекурсивно - зліченна.*
4. *Будь - яка рекурсивно - зліченна множина m – зводиться до множини К.*

13.5 Теореми Кліні про нерухому точку

Теорема Кліні про нерухому точку є основним інструментом дослідження в теорії рекурсивних функцій (РФ) та частково - рекурсивних функцій (ЧРФ). Він надає витончений і економічний метод поводження з конструкціями, що в іншому випадку вимагало б довгих і складних міркувань.

Ця теорема може бути наведена в декількох формах і може розглядатися з кількох точок зору. У певному сенсі теорема підсумовує певний клас діагональних методів. З іншого боку, ця теорема встановлює певний результат про нерухому точку і, подібно до теорем про нерухому точку з математичного аналізу, може бути використана для доказу існування багатьох неявно заданих функцій.

Теорема Кліні. *Якою б не була обчислювана всюди визначена функція q(n), знайдеться таке n, при якому U(n, x) = U(q(n), x),*

де U(n, x) - геделівська універсальна функція. Отже, завжди існує таке число n, що $U_n = U_{q(n)}$, тобто n і q(n) – номери однієї функції.

Теорема 13.5.1. *Нехай f - (n+1) - арна РФ. Тоді існує n - арна РФ g така, що* $\varphi^{g(x_1,\dots,x_n)} = \varphi^{f(g(x_1,\dots,x_n),x_1,\dots,x_n)}$ *для всіх значень* $x_1, \dots, x_n$.

Переформулюємо для випадку $n=0$.

Теорема 13.5.2. *Нехай f(x) - РФ. Тоді існує n∈N таке, що* $\varphi_n = \varphi_{f(n)}$.

Первісне формулювання самого С. Кліні теореми про нерухому точку має такий вигляд:

Теорема 13.5.3 *Нехай h(z, x) - ЧРФ. Тоді існує n∈N таке, що для всіх x маємо* $h(n, x)=\varphi_n(x)$.

Теорему Кліні про нерухому точку краще називати теоремою про псевдонерухому точку. Насправді, умова $\varphi_n=\varphi_{f(n)}$ зовсім не означає, що $n = f(n)$, а свідчить тільки про те, що n та $f(n)$ - індекси однієї і тієї самої ЧРФ.

Теорему про нерухому точку можна переформулювати і так:

Теорема 13.5.4. *Нехай U (n; x) - головна обчислювана універсальна функція для класу обчислюваних функцій одного аргументу. Нехай V (n; x) - довільна обчислювана функція. Тоді функції U і V збігаються на деякому перерізі: знайдеться таке p, що $U_p = V_p$, тобто U (p; n) = V (p; n) для будь-якого n.*

Приклад 2. Існує $n \in N$ таке, що для всіх x маємо $\varphi_n(x)=2n+x^{3n}$.

Візьмемо функцію $h(z, x)=2z+x^{3z}$. За теоремою 7.4.3 існує таке n, що для всіх x $h(n, x)=\varphi_n(x)$. Отже, $\varphi_n(x)=h(n, x)=2n+x^{3n}$ для всіх x.

Приклад 3. Існує $n \in N$ таке, що $D_n=E_n=\{n\}$.

Візьмемо функцію $h(z, x)=\begin{cases} x, & \text{я як } x = z, \\ \text{не визначене} & \text{інакше.} \end{cases}$ Така $h \in$ ЧРФ. За теоремою 7.4.3 існує таке n, що для всіх x маємо $h(n, x)=\varphi_n(x)$. Тоді $\varphi_n(x)=\begin{cases} x, & \text{я як } x = n, \\ \text{не визначене} & \text{інакше.} \end{cases}$ Звідси $D_n=E_n=\{n\}$.

Теорема 13.5.5. *Для кожної РФ f(x) існує строго монотонна РФ α(x) така, що для кожного n ∈ N маємо* $\varphi_{\alpha(n)} =\varphi_{f(\alpha(n))}$.

Наслідок. *Для кожної РФ f(x) та для кожного k ∈ N існує n ∈ N таке, що n>k та* $\varphi_n=\varphi_{f(n)}$.

Класичним прикладом застосування теореми про нерухому точку є такий її *наслідок: існує програма, що друкує свій власний текст.* В самому разі, якщо б такої програми не було, то відображення

p → (програма, що друкує при будь- якому вході p)

не мало б нерухомої точки. Формально цей факт можна виразити теоремою.

Теорема 13.5.6. *Нехай U(n,x) – головна обчислювана універсальна для класу всіх обчислюваних функцій одного аргументу. Тоді існує таке число p, що U(p,x) = p для будь - якого x.*

У термінах програмування на ЕОМ : нехай U(p,x) – результат застосування алгоритмічної програми p (складеній будь-якою мовою програмування) до стандартного входу x. Зрозуміло, що функція U буде головною універсальною функцією, тому, застосовуючи твердження теореми, одержимо програму p, що незалежно від входу на виході видасть p.

Контрольні запитання

1. Чи є $N \smile N^2$ конструктивним простором? Відповідь поясніть.
2. Яке твердження можна надати формулі ∀x(xxS(0) = x), що розглядається в прикладі 13.1 ?
3. Які види нумерацій трапляються в теорії нумерацій алгоритмів.
4. Про що стверджується в теоремі про нерухому точку?
5. Яким чином будується канторова нумерація кортежів?

- Задачі для самостійного розв’язання

Задачі для самостійного розв’язання

1. Знайти канторові номери впорядкованих трійок СЗ(0, 0, 1) ; СЗ(1, 0, 0); СЗ(1, 1, 0).
2. За формулами (13.3), (13.4) знайдіть аналітичні залежності x = l(n) та y = r(n).
3. Використовуючи нумерацію за Гьоделем, знайдіть гьоделівський номер формули $\exists$ x(x+S(0) = 2xx).
4. Доведіть теорему 13.1.
5. Доведіть, що відношення «m – зведення» рефлексивне і транзитивне.

Тема 14

- Складність алгоритмів
  - 14.1. Асимптотичні оцінки складності
  - 14.2. Класи складності

Складність алгоритмів

Створення та реалізація алгоритму відповідно до свого призначення визначає його складність. Проте не існує інтегрованого показника складності алгоритму, хоча існує спеціальний навіть розділ – метрична теорія алгоритмів, що займається саме проблемами складності. Інтуїтивно можна виділити такі основні складові складності алгоритму:

1. Логічна складність - кількість людино-місяців, витрачених на створення алгоритму.
2. Статична складність - довжина опису алгоритмів (кількість операторів).
3. 3. Тимчасова складність - час виконання алгоритму.
4. Ємнісна складність - кількість умовних одиниць пам'яті, необхідних для роботи алгоритму.

Головною метою теорії складності є забезпечення механізму класифікації алгоритмів за складністю. Складність алгоритму дозволяє визначитися з вибором ефективного алгоритму серед існуючих, що побудовані для розв’язання конкретної проблеми. А саме вибір серед уже існуючих алгоритмів дозволяє не розглядати логічну та статичну складність, а оцінювати ті ресурси, що знадобляться під час реалізації обраних алгоритмів.

Означення. Складність алгоритму – *це кількісна характеристика, що відображує споживані алгоритмом ресурси під час свого виконання.*

Основними ресурсами, що оцінюються, є час виконання і простір пам'яті.

Інтуїтивно це поняття досить зрозуміле. В алгоритму є вхід - опис завдання, яке потрібно вирішити. На його розв'язання алгоритм витрачає певний час (тобто кількість операцій). Складність - це функція від довжини входу, значення якої дорівнює максимальному (за будь-якими входами даної довжини) кількості операцій, необхідних алгоритму для отримання відповіді.

Приклад 14.1. Нехай дана послідовність з нулів та одиниць і нам потрібно з'ясувати, чи є там хоч одна одиниця. Яку складність матиме алгоритм розв'язання цієї задачі?

Розв'язання. Нехай n – кількість символів в послідовності. Алгоритм буде послідовно перевіряти, чи немає одиниці в поточному місці заданої послідовності, а потім рухатися далі, поки вхід не скінчиться. Оскільки одиниця дійсно може бути тільки одна, для отримання точної відповіді на це питання в гіршому випадку доведеться перевірити всі *n* символів входу. В результаті отримуємо складність порядку *cn*, де *c* кількість кроків, що потрібна для перевірки поточного символа і переходу до наступного. Оскільки такого роду константи сильно залежать від конкретної реалізації, математичного сенсу вони не мають, і їх зазвичай ховають за символом *O*(О-велике) : в даному випадку фахівець із теорії складності визначив би, що алгоритм має складність

O (n) (про символіку складності див. 14.1); іншими словами, він лінійний.

Існує і продовжує розширюватися клас варіантів поняття складності: складність знизу, зверху й у середньому, алгебраїчна складність, мультиплікативна складність, бітова складність, фундаментальні асимптотичні оцінки складності, оцінка алгоритмів за їх належністю до класів складності самих проблем, що вони розв'язують, і т. д.

14.1. Асимптотичні оцінки складності

Одним зі спрощених видів аналізу складності алгоритмів, що використовують при комп'ютерній їх реалізації, є асимптотичний аналіз алгоритмів. Він використовується з метою порівняння витрат часу та інших ресурсів різноманітними алгоритмами, призначеними для вирішення одного і того самого завдання. Досліджуючи зростання часу роботи алгоритму при вхідних даних досить великих розмірів, ми тим самим вивчаємо асимптотичну ефективність алгоритмів. Це означає, що нас цікавить тільки те, як час роботи алгоритму зростає зі збільшенням розміру вхідних даних, коли цей розмір збільшується до нескінченності. Зазвичай алгоритм, більш ефективний в асимптотичному сенсі, буде більш продуктивним для всіх вхідних даних, за винятком дуже маленьких. Нижче перелічені основні оцінки складності.

Позначення, що вводяться для опису асимптотичної поведінки часу роботи алгоритму, використовують функції, область визначення яких множина невід'ємних цілих чисел N = {0,1,2, ...}. Подібні позначення зручні для опису часу роботи *T(n)* в найгіршому випадку як функції, визначеної тільки для цілих чисел, що становлять розмір вхідних даних.

Означення. *Функція складності алгоритму f(n) має оцінку* Θ(тета) *й записується як f(n) = Θ(g(n)), якщо існує невід'ємна функція g(n) та додатні* n_0, c_1, c_2 *такі, що*

$$c_1 g(n) \le f(n) \le c_2 g(n), \qquad (14.1)$$

при $n > n_0$.

У такому разі говорять, що функція *g(n)* є асимптотично точною оцінкою функції *f(n),* оскільки за визначенням функція *f(n)* не відрізняється від функції *g(n)* з точністю до сталого множника. Важливо розуміти, що

Θ *(g(n))* є не однією функцією, а множиною функцій для опису зростання *f(n)* з точністю до сталого множника. Іншими словами, функція *f(n)* належить множині Θ *(g(n)),* якщо існують додатні c_1 та c_2 , що дозволяють обмежити цю функцію у рамки між функціями $c_1 g(n)$ та $c_2 g(n)$ для достаньо великих значень n.

Наприклад, для методу сортування послідовності чисел алгоритмом heapsort оцінка трудомісткості становить *f(n) = Θ (*$n \log_2 n$*)*, тобто в цьому разі $g(n) = n \log_2 n$.

З означення *f(n) = Θ (g(n))* випливає, що *g(n) = Θ (f(n)).*

Інтуїтивно зрозуміло, що при асимптотично точній оцінці асимптотично невід'ємних функцій, доданками нижчих порядків у них можна знехтувати, оскільки при великих n вони стають неістотними. Навіть невеликої частки доданка найвищого порядку достатньо для того, щоб перевершити доданки нижчих порядків. Таким чином, для виконання нерівностей (14.1), достатньо як c_1вибрати значення, яке дещо менше коефіцієнта при самому старшому доданку, а як c_2 значення, яке дещо більше цього коефіцієнта.

Тому коефіцієнт при старшому доданку можна не враховувати, тому що він лише змінює зазначені константи.

Приклад 14.1. Розглянемо квадратичну функцію $f(n) = an^2 + bn + c$, де *a, b, c* – константи, причому *a > 0.* Відкидаючи доданки нижчих порядків за перший та ігноруючи коефіцієнт a, одержимо $f(n) = \Theta(n^2)$.

Узагалі кажучи, для будь-якого полінома *p(n)* маємо $p(n) = \Theta(n^d)$, де *d* – степінь полінома при $a_d > 0$. Оскільки будь-яка константа – це поліном нульового степеня, то сталу функцію можна записати як $\Theta(n^0)$ або *Θ(1).*

Означення. *Функція складності алгоритму f(n) має оцінку O(«О-велике») й записується як f(n) = O(g(n)), якщо існує невід'ємна функція g(n) та додатні* n_0, *c такі, що*

$$0 \le f(n) \le cg(n), \qquad (14.2)$$

при $n > n_0$.

Означення. *Функція складності алгоритму f(n) має оцінку Ω(«омега-велике») й записується як f(n) = Ω(g(n)), якщо існує невід'ємна функція g(n) та додатні* n_0, *c такі, що 0 ≤, cg(n) ≤ f(n)* (14.3)

при $n > n_0$.

O- позначення застосовуються, коли необхідно вказати верхню межу функції з точністю до сталого множника. В позначеннях теорії множин $\Theta(g(n)) \subset O(g(n))$. Оскільки О-позначення описують верхню межу, то в ході їх використання для обмеження часу роботи алгоритму в найгіршому випадку ми отримуємо верхню межу цієї величини для будь-яких вхідних даних. Таким чином, асимптотична оцінка $O(n^2)$ для часу роботи алгоритму у найгіршому випадку застосовна для часу виконання завдання з будь-якими вхідними даними, чого не можна сказати про Θ-позначення. Наприклад, оцінка в $\Theta(n^2)$ для часу сортування вставками в найгіршому випадку не придатна для довільних вхідних даних. Наприклад, якщо вхідні елементи, що подаються на сортування, вже відсортовані в потрібному порядку, час роботи алгоритму сортування вставкою оцінюється як Θ(n).

Оскільки Ω - позначення використовуються для визначення нижньої межі часу роботи алгоритму в найкращому випадку, то вони визначають нижню межу часу роботи алгоритму при довільних вхідних даних. Наприклад, час роботи алгоритму сортування вставками знаходиться в межах між Ω(n) та $O(n^2)$, тобто між лінійною та квадратичною функціями від n.

На рис. 14.1 графічно подані введені вище позначення.

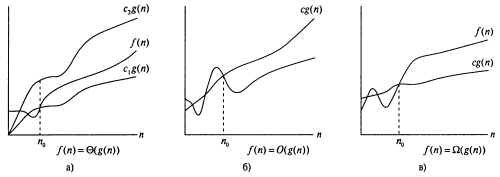


Рисунок 14.1 – Графічні приклади асимптотичних позначень

Теорема 14.1. Для будь-яких двох функцій *f(n)* і *g(n)* співвідношення *f(n) = Θ(g(n))* виконується тоді й тільки тоді, коли *f(n)=O(g(n))* і *f(n)=Ω(g(n)).*

На практиці ця теорема застосовується для визначення асимптотично точної оцінки алгоритму за допомогою асимптотично верхньої та нижньої меж.

Асимптотичний аналіз алгоритмів має не лише практичне, а й теоретичне значення. Так, наприклад, доведено, що всі алгоритми сортування, ґрунтуються на попарному порівнянні елементів, відсортують n елементів за час, не менший $\Omega(n \log_2 n)$.

14.2. Класи складності

На початку 1960-х років, у зв'язку з початком широкого використання обчислювальної техніки для вирішення практичних завдань, виникло питання про межі практичної застосовності цього алгоритму розв'язання задачі в сенсі обмежень на її розмірність. Які завдання можуть бути вирішені на ЕОМ за реальний час? Відповідь на це питання була дана у працях Кобмена (Alan Cobham, 1964) та Едмнодса (Jack Edmonds, 1965), де були введені класи складності задач.

У теорії алгоритмів *класами складності називаються множини обчислювальних задач, приблизно однакових за складністю обчислення*. Говорячи більш вузько, класи складності - це множини предикатів (функцій, які отримують на вхід слово і повертають відповідь 0 або 1), що використовують для обчислення ресурси приблизно однакової кількості.

У рамках класичної теорії здійснюється класифікація завдань за класами складності (P-складні, NP-складні, експоненціально складні та ін.).

Кожен клас складності (у вузькому сенсі) визначається як множина предикатів, що мають деякі властивості. Типове визначення класу складності має такий вигляд.

Означення. *Класом складності X називається множина предикатів P (x), що обчислюються на машинах Тюрінга і використовуються для обчислення*

O (f (n)) ресурсу, де n - довжина слова x.

Як ресурси зазвичай береться час обчислення (кількість робочих тактів машини Тюрінга) або

робоча зона (кількість використаних осередків на рядку під час роботи). Мови, які розпізнаються предикатами з деякого класу (тобто множини слів, на яких предикат повертає 1), також називаються мовами, що належать до того самого класу. Класи прийнято позначати прописними літерами. Доповнення до класу С (тобто клас мов, доповнення яких належать С) позначається со-С.

Для кожного класу існує категорія задач, які є «найскладнішими». Це означає, що будь-яка задача з класу зводиться до такої задачі, і до того ж сама задача лежить у класі. Такі задачі називають повними задачами для даного класу. Найбільш відомими є NP-повні задачі. Повні задачі - хороший інструмент для доведення рівності класів. Достатньо для однієї такої задачі подати алгоритм, що розв'язує її і належить більш маленькому класу, і рівність буде доведено.

Розглянемо найбільш відомі класи складності задач.

1. **Клас P** *(задачі з поліноміальною складністю).*

Формальне визначення. Алгоритм ототожнюється з детермінованою машиною Тюрінга, яка обчислює відповідь при даному на вхідний рядок слові із вхідного алфавіту ∑. Час роботи алгоритму $T_M(x)$ при фіксованому вхідному слові x визначається кількістю робочих тактів машини Тюрінга від початку до зупинки машини.

Складністю функції f: ∑* → ∑*, що обчислюється деякою машиною Тюрінга, називається функція C: N → N, що залежить від довжини вхідного слова і дорівнює максимуму часу роботи машини за всіма вхідними словами фіксованої довжини :

$$C_M(n) = \max_{x:|x|=n} T_M(x) \quad (14.4)$$

Означення. *Якщо для функції f існує машина Тюрінга M така, що*

$C_M(n) < n^c$ *для деякого числа c і досить великих n, то кажуть, що вона належить класу P, або поліноміальна за часом.*

Тобто, задача називається поліноміальною (належить до класу P), якщо існують константа k та алгоритм, що розв'язує задачу з $F_a(n)=O(n^k)$, де n - довжина входу алгоритму в бітах n = |D| .

Згідно з тезою Черча - Тюрінга будь-який мислимий алгоритм можна реалізувати на машині Тюрінга. Для будь-якої мови програмування можна визначити клас P подібним чином (замінивши у визначенні машину Тюрінга на реалізацію мови програмування). Якщо компілятор мови, на якому реалізований алгоритм, уповільнює виконання алгоритму поліноміально (тобто час виконання алгоритму на машині Тюрінга менший деякий за многочлен від часу виконання його на мові програмування), то визначення класів P для цієї мови і для машини Тюрінга збігаються.

Більш вузьке визначення. Іноді під класом P мають на увазі більш вузький клас функцій, а саме клас предикатів (функцій f: ∑* → {0, 1 }). У такому разі мовою L, що розпізнає даний предикат, називається множина слів, на яких предикат дорівнює 1. Мовами класу P називаються мови, для яких існують предикати, що їх розпізнають, класу P. Очевидно, що якщо мови L_1 і L_2 належать класові P, то і їх об'єднання, перетин та доповнення також належать класові P.

Задачі класу P - це, інтуїтивно, задачі, розв'язувані за реальний час. Відзначимо такі переваги алгоритмів із цього класу:

- для більшості задач із класу P константа k менше 6;
- клас P інваріантний за моделлю обчислень (для широкого класу моделей);
- клас P має властивість природної замкненості (сума або добуток поліномів є поліном).

Таким чином, задачі класу P є уточненням визначення «практично розв'язної» задачі. Прикладами завдань із класу P є цілочислове додавання, множення, ділення, одержання залишку від ділення, множення матриць, з'ясування зв'язності графів і деякі інші.

2. **Клас NP** (*поліноміальної перевірки*).

Клас NP складається із задач, розв'язання яких можна перевірити протягом поліноміального часу. Мається на увазі, що якщо ми одержимо якимось чином розв'язок деякої задачі цього класу, то протягом часу, який буде поліноміально залежати від розміру вхідних даних задачі , можна перевірити коректність такого розв'язку.

Означення. *∀ $D \in D_A$, |D| = n поставимо у відповідність сертифікат $S \in S_A$, такий, що | S | = O (n^l), та алгоритм $A_S = A_S(D, S)$, такий, що видає «1», якщо розв'язок правильний, і «0» , якщо розв'язок неправильний. Тоді задача належить до класу NP, якщо F (A_S) = O (n^m).*

Еквівалентне визначення можна отримати, використовуючи поняття недетермінованої машини Тюрінга (тобто такої машини Тюрінга, у програми якої можуть існувати різні рядки з однаковою лівою частиною).

Означення. *Мова $\tilde{L}$ належить до класу NP (недетермінованих поліноміальних), якщо вона розпізнається недетермінованою машиною Тюрінга з поліноміальною часовою складністю T(n).*

Отже, основна відмінність класів P і NP та, що до класу P належить задачі, які можуть бути розв'язані за час, що поліноміально залежить від об'єму початкових даних, за допомогою детермінованої обчислювальної машини (наприклад, машини Тюрінга), а до класу NP — задачі, які можуть бути розв'язані за поліноміально виражений час за допомогою недетермінованої обчислювальної машини, тобто машини, наступний стан якої не завжди однозначно визначається попередніми. Роботу такої машини можна представити як процес, що розгалужується на кожній неоднозначності: задача вважається розв'язаною, якщо хоча б одна гілка процесу прийшла до відповіді. Оскільки кожна детермінована машина Тюрінга може розглядатись як недетермінована, але без вибору варіантів кроків, то клас P міститься в класі NP (P⊆NP). Оскільки клас P міститься в класі NP, приналежність того або іншого завдання до класу NP часто відображає наше поточне уявлення про способи розв'язання даної задачі й носить неостаточний характер.

До задач класу складності NP належать, наприклад, задачі: розв'язність логічного виразу, три кольорове розфарбування графу, побудова кліки з k вершин на неорієнтованому графі, задача покриття множин та інші.

У загальному випадку немає підстав вважати, що для тієї або іншої NP-задачі не може бути знайдене P-розв'язок. Питання про можливу еквівалентність класів P і NP (тобто про можливість знаходження P-розв'язку для будь-якої NP-задачі) вважається багатьма одним із основних питань сучасної теорії складності алгоритмів. Сама постановка питання про еквівалентність класів P і NP можлива завдяки введенню поняття NP-повних задач.

3. **Клас NPC** (*NP - повні задачі*)

Неформально задача належить класу NPC, якщо вона належить класу NP і є такою самою «складною», як і будь-яка задача із класу NP. NP-повні задачі складають підмножину NP-задач і відрізняються тією властивістю, що всі NP-задачі можуть бути тим або іншим способом зведені до них. З цього виходить, що якщо для NP-повної задачі буде знайдений P-розв'язок, то P-розв'язок буде знайдений для всіх задач класу NP.

Поняття NP - повноти було введене на початку 1970-х років і ґрунтується на понятті зведення однієї задачі до іншої.

Зведення може бути представлене у такий спосіб: якщо ми маємо задачу 1 та алгоритм, що розв'язує цю задачу, тобто видає правильну відповідь для всіх конкретних проблем, що становлять задачу, а для задачі 2 алгоритм розв'язання невідомий, але якщо ми можемо переформулювати (звести) задачу 2 у термінах задачі 1, то ми розв'язуємо задачу 2.

Таким чином, якщо задача 1 задана множиною конкретних проблем D_{A1}, а задача 2 – множиною D_{A2}, й існує функція f_s(алгоритм), що зводить конкретну постановку задачі 2 (d_2) до конкретної постановки задачі 1(d_1):

f_s ($d_{(2)} \in D_{A2}$) = $d_{(1)} \in D_{A1}$, то задача 2 зводиться до задачі 1.

Якщо при цьому $F(f_s) = O(n^K)$, тобто алгоритм зведення належить класу P, то говорять, що задача 1 поліноміально зводиться до задачі 2.

Означення. *Задача належить до класу NPC, якщо виконуються такі дві умови: по-перше, задача повинна належати до класу NP (L є NP), і, по-друге, до неї поліноміально повинні зводитися всі задачі із класу NP (Lx=< p, для кожного Lx є NP).*

Виконання цього означення показане на рис. 14.2.

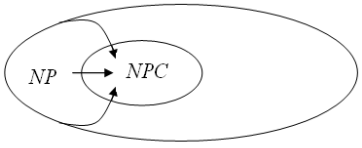


Рисунок 14.2 – Зведення і клас NPC.

Теорема 14.1. *Якщо існує задача, що належить до класу NPC, для якої існує поліноміальний алгоритм розв'язання ($F = O(n^k)$), то клас P збігається із класом NP, тобто P=NP.*

Схема доведення буде складатися зі зведення будь-якої задачі з NP до даної задачі із класу NPC з поліноміальною трудомісткістю й розв'язання цієї задачі за поліноміальний час (за умовою теореми).

У цей час доведено існування сотень NPC задач, але для жодної з них поки не вдалося знайти поліноміального алгоритму розв'язання. У цей час дослідники припускають таке співвідношення класів, що наведене на рис. 14.3, де P ≠NP і задачі із класу NPC не можуть бути розв'язані (сьогодні) з поліноміальною трудомісткістю.

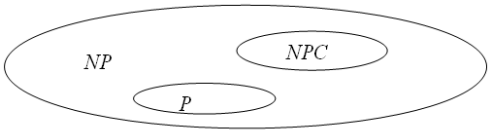


Рисунок 14.3 – Співвідношення класів P, NP, NPC.

Оскільки метод зведення базується на тому, що для деякої задачі відома її NP повнота, то для доведення NP – повноти різних задач, нам знадобиться «перша» NP – повна задача. Як таку використовуємо задачу на здійсненність, у якій задана булева комбінаційна схема, що складається з логічних елементів (або задано КНФ – кон'юнктивну нормальну форму). В задачі запитується, чи існує набір вхідних булевих величин, для яких на виході буде видане значення 1.

Прикладами NP-повних задач є, крім задачі про кон'юнктивну форму, задача комівояжера, розфарбування графа, задача про гамільтонів цикл у графі та інші.

4. Інші класи складності

Дослідження складності алгоритмів дозволили по-новому поглянути на розв'язання багатьох класичних математичних задач і знайти для ряду таких задач (множення многочленів і матриць, розв'язання лінійних систем рівнянь та ін.), розв'язання, що вимагають менше ресурсів, ніж традиційні.

Прості класи складності визначаються такими факторами:

- *типом обчислювальної задачі:* найбільш часто використовуються задачі прийняття рішень.
- *моделлю обчислень:* найбільш поширеною моделлю обчислень є детермінована машина Тюрінга, але багато класів складності визначають на основі недетермінованої машини Тюрінга, логічних схем, квантової машини Тюрінга, монотонних схем і т.д.
- *ресурсами, які мають межі:* вказується як "поліноміальний час", "логарифмічний простір", "стала глибина" тощо.

Усі класи складності знаходяться в ієрархічному відношенні: одні містять у собі інші. Однак про більшість включень невідомо, чи є вони строгими. Одна з найбільш відомих відкритих проблем у цій сфері рівність класів P і NP. На даний момент найбільш поширеною є гіпотеза про невиродженість ієрархії (тобто всі класи різні).

Наведемо приклад побудови однієї з можливих ієрархічних структур класів складності.

Рисунок 14.4 – Ієрархія класів складності

На цій ієрархії представлені, наприклад, такі класи, що побудовані на ймовірнісних машинах Тюрінга (BPP), префіксом *co* позначені доповнення до класів, PSPACE – клас, допустимий детермінованою машиною Тюрінга з поліноміальним обмеженим простором і т. д.

Запитання для самоперевірки

1. Сформулюйте основні асимптотичні оцінки, що застосовуються в теорії алгоритмів.
2. Розташуйте в порядку зростання асимптотичні оцінки $O(n)$, $O(n^2)$, $O(1)$, $O(\log_2 n)$, $O(n\log_2 n)$, $O(\log_2 \log_2 n)$, $O(n^3)$, $O(2^n)$ для $n > 100$.
3. Зазначте основні класи складності алгоритмів та побудуйте діаграму Ейлера про їх взаємозв'язок.
4. Сформулюйте основну проблему теорії складності.
5. Сформулюйте означення терміна «клас складності».

- Задачі для самостійного розв'язання

Задачі для самостійного розв'язання

1. Наведіть приклади рефлективності, транзитивності та симетричності для асимптотичних оцінок.
2. Нехай f(n) і g(n) - невід'ємні функції. Доведіть за допомогою базового означення θ-позначень, що $\max(f(n),g(n)) = \theta(f(n) + g(n))$.
3. Чи є алгоритм динамічного програмування для рішення дискретної задачі про рюкзак алгоритмом з поліноміальним часом? Відповідь обґрунтуйте.
4. Доведіть таке твердження - якщо NP – повна задача розв'язна на протязі поліноміального часу, то P = NP.

Коментарі

В цьому розділі значна частина матеріалу представлена із [13]. Використовується також матеріал Internet- ресурсів (Вікіпедії) та праць [8-11, 16, 28].

Тема 15

- NP - повні, складні та алгоритмічно нерозв'язні проблеми
  - 15.1. Розв'язні та нерозв'язні проблеми, NP- повнота, складність, зведення
  - 15.1.2. Приклади NP-повних задач
  - 15.2. Алгоритмічно нерозв'язні проблеми

NP - повні, складні та алгоритмічно нерозв'язні проблеми

15.1. Розв'язні та нерозв'язні проблеми, NP- повнота, складність, зведення

Теорія алгоритмів розподіляє всі проблеми, що потребують вирішення, на дві широкі категорії : алгоритмічно розв'язні та нерозв'язні. Останні проблеми будуть розглянуті дещо пізніше. Зупинимося на перших.

Бурхливий розвиток комп'ютерної індустрії підсвідомо наводить на думку, що вже не так важливо, які алгоритми застосовувати для розв'язної проблеми, - сучасний комп'ютер може все. Однак, наприклад, алгоритм «грубої сили» для розв'язування задачки на триста-п'ятсот змінних може зажадати такої кількості операцій, що набагато більше, ніж у Всесвіті елементарних частинок. Наприклад, якщо розв'язання задачі на найсучасніших комп'ютерах займе 10^{10} років, то очевидно, такий алгоритм для нас не має сенсу.

У попередньому розділі ми ввели поняття «складність алгоритму». Зрозумілим є бажання аналогічно визначити і складність завдання– наприклад, як складність найефективнішого алгоритму, що розв'язує цю задачу (для даних розміру n). На жаль, це бажання нездійсненне. Доведено, що є завдання, для яких не існує найшвидшого алгоритму, тому що будь-який алгоритм для такого завдання можна «прискорити», побудувавши більш швидкий алгоритм, що розв'язує цю задачу. Це твердження називають теоремою Блюма про прискорення. Якщо відволіктися від технічних деталей, то спрощений варіант теореми Блюма можна сформулювати таким чином.

Теорема 15.1 (теорема Блюма про прискорення). *Існує така алгоритмічно розв'язна проблема Z, що будь-який алгоритм A, що вирішує цю проблему, можна прискорити таким чином: існує інший алгоритм A_1, що також вирішує Z і такий, що $T_{A1}(n) \le \log T_A(n)$ для майже всіх n.*

Зауважимо, що теорема Блюма не стверджує, що прискорення можливе для будь-якої задачі.

Більше того, для деяких задач існує оптимальний (найшвидший) алгоритм. Однак твердження теореми Блюма про існування «незручних» задач не дозволяє визначити універсальне (застосовне до всіх задач) поняття «оптимального алгоритму».

Тому в теорії складності алгоритмічно розв'язних проблем використаний інший підхід через класи складності.

Означення 15.1. *Нехай f (n) - деяка функція, що відображає N в N. Клас складності C (f (n)) - це множина всіх задач, для яких існує хоча б один алгоритм, складність якого не перевищує O (f (n)).*

Це визначення в певному сенсі умовне - позначення *C(f (n))* ніколи не застосовують. Чому? Тому що для завдання реального класу задач необхідно ще уточнити:

- що ми розуміємо під «алгоритмом»;
- яка складність (тимчасова, ємнісна чи ще якась) нас цікавить.

При різних відповідях на ці питання одержимо різні класи завдань, і для кожного класу використовується спеціальне позначення.

Термінологія класів складності з 70-х років значно змінилася й поповнилася новими термінами, розширила рамки вже існуючих. Незмінними аспектами при побудові глобальної картини ієрархії складності залишаються такі.

- У теорії складності під «алгоритмом» зазвичай розуміють той чи інший різновид машини Тюрінга. Винайдення нових різновидів приводить до появи й нових класів складності. Наприклад, окрім уже відомих класів P, NP, NPC, з'явилися класи: EXPTIME (іноді має назву EXP) - задачі, які розв'язуються детермінованою машиною Тюрінга за O ($2^{p(n)}$) часу, де p(n) є поліномом функції n; EXPSPACE - задачі, для розв'язання яких детермінованій машині Тюрінга потрібно O($2^{p(n)}$) ємнісного простору, де p(n) є поліномом функції n; BPP – проблеми, розв'язні за допомогою ймовірнісної машини Тюрінга в поліноміальний час із похибкою ймовірністю не більше 1/3 та інші. Є й екзотичні класи, наприклад, P^{NP}- клас задач, що розв'язуються за поліноміальний час детермінованою машиною Тюрінга з оракулом для NP-завдання.
- Для формальних визначень класів складності зазвичай розглядають не довільні алгоритми, а алгоритми для так званих задач розв'язання (decision problem), у яких потрібно визначити, належить чи ні певний елемент деякій множині. Враховуючи необхідність кодування даних, що подаються на вхід машині Тюрінга, ці задачі абсолютно еквівалентні задачам розпізнавання мов, коли на деякому алфавіті ∑ розглядається підмножина слів L ⊂ ∑* і для довільного слова l ∈∑* потрібно визначити, чи належить воно мові L.
- У сучасній теорії складності обчислень поняття поліноміального алгоритму є адекватним математичним уточненням інтуїтивного поняття «ефективний алгоритм», а клас P становить «клас ефективно розв'язуваних задач». Тобто, прості задачі (розв'язувані) - це задачі, які розв'язування за поліноміальний час. Важкорозв'язні задачі - це задачі, які не розвязуються за поліноміальний час, або алгоритм розв'язання за поліноміальний час не знайдений.
- Побудовано ієрархічне співвідношення класів складності : P⊆NP⊆ BPP⊆ PH⊆ EXPSPACE⊆ EXPTIME, тобто одні містять у собі інші (більшість включень не доведено). Однак, зважаючи на потужності сучасних засобів обчислювальної техніки, задачі інших класів складності розглядаються поки що тільки в теорії, а однією з найбільш відомих відкритих проблем залишається питання про рівність класів P та NP.

У теорії складності обчислень одним із основних інструментаріїв, що застосовуються, є відомий науковий метод зведення - перетворення однієї задачі в іншу. У загальному випадку, якщо у нас є алгоритм, що перетворить екземпляри задачі P_1в екземпляри задачі P_2, які мають ту самому відповідь (так / ні), то говорять, що P_1 зводиться до P_2. Таким чином, зведення - це відношення між двома задачами. За допомогою такого зв'язку може бути доведена приналежність до того або іншого класу складності.

Зведення алгоритмічне - одне з основних понять теорії алгоритмів. Виникло у зв'язку з тим, що нерозв'язність (і розв'язність) багатьох алгоритмічних проблем встановлюється здебільшого не безпосередньо, а шляхом зведення до досліджуваної проблеми такої алгоритмічної проблеми, нерозв'язність якої вже доведена.

Поняття алгоритмічного зведення було уточнено А. Тюрінгом: *якщо, деяка машина Тюрінга переробляє послідовність закодованих значень функції g у послідовність закодованих значень функції f, то функція f зводиться до функції g.* Якщо f зводиться до g за Тюрінгом, $f \le_T g$, а g зводиться до f, $g \le_T f$, то кажуть, що f і g мають один і той степінь нерозв'язності, або $f \equiv_T g$.

Останнім часом частіше застосовуються зведення за Карпом або Куком.

Означення 15.2. *Будь-яка мова програмування L_1 називається зводимою за Карпом до мови L_2, якщо існує функція F:* $\sum^* \to \sum^*$*, що обчислюється за поліноміальний час, де F (x) належить L_2 в тому разі, якщо x належить L_1.*

Розглянемо дві мови L_1 і L_2 над алфавітами $\sum$ і Г. Зведення L_1 до L_2 за Карпом - це функція f: $\sum^* \to \Gamma^*$, обчислювана за поліноміальний час, така, що $\forall$ x ∈ $L_1 \leftrightarrow$ f(x) ∈ L_2. Таким чином, неформально мова L_1 «не складніша» за мову L_2. Якщо така функція існує, кажуть, що L_1зводиться за Карпом до L_2, і пишуть $L_1 \le_K L_2$. Відмітимо, що зведення за Карпом є частковим випадком зведення за Куком.

Означення 15.3. *Зведення задачі R_1до задачі R_2 за Куком – це поліноміальний за часом алгоритм (іншими словами, машина Тюрінга з поліноміальним часом роботи), що розв'язує задачу R_1 за умови, що функція, що знаходить розв'язок задачі R_2, йому дана як оракул, тобто звернення до неї займає один крок.*

Якщо такий алгоритм існує, кажуть, що R_1 зводима за Куком до R_2 і пишуть $R_1 \le_C R_2$.

Поява деяких нових класів складності пов'язана із залученням до процесу їх вивчення так званої «пророчої машини» - машини Тюрінга з оракулом. Пророк, у цьому розрізі, розглядається як сутність, здатна відповідати на набір питань і зазвичай представлена як деяка підмножина натуральних чисел. Тоді, інтуїтивно, пророча машина може виконувати всі звичайні дії машини Тюрінга і також може запитати у пророка: «Чи x належить A?».

Машина Тюрінга взаємодіє з оракулом шляхом запису на свій рядок вхідних даних для оракула і потім запускає оракул на виконання. За один крок оракул обчислює функцію, стирає вхідні дані й пише вихідні дані у рядок.

Клас складності задач, розв'язуваних алгоритмом з класу A з оракулом для задачі класу B, позначають A^B. Наприклад, клас задач, розв'язуваних за поліноміальний час детермінованою машиною Тюрінга з оракулом для NP-завдання, позначають P^{NP}.

Повернемося до одного з основних питань - яке ж практичне значення має вивчення теорії складності й класифікація завдань з точки зору NP-повноти (NPC)? Відповідь очевидна - часто набагато розумніше та ефективніше знайти доказ того, що розглянута задача належить до класу NPC, й відповідно до цього зайнятися пошуком досить точних наближених алгоритмів, ніж безрезультатно витрачати час на відшукання поліноміальних алгоритмів її розв'язання. Зрозуміло, що саме NPC-задачі відіграють тут центральну роль - справа в тому, що поліноміальний час є, хоча й першим, але досить гарним наближенням поняття «практичної можливості розв'язання задачі».

Нагадаємо, що задачі класу NPC відповідають двом умовам: по-перше, вони обов'язково належать класу NP; по-друге, будь-яка довільна задача з класу NP зводиться до цих задач. Якщо опустити першу умову, то одержимо клас задач *NPH(non-deterministic polynomial-time hard)* - клас задач, які "принаймні так складні, як найбільш важкі задачі в NP". Клас NPH містить у собі NPC (див. рис. 15.1), але виходить із меж класу NP(за гіпотези P ≠ NP). Серед задач NPC виділяють також задачі *NPCS – NP- повні в сильному сенсі*, як ті, що належать класу NP, є NP-повними та мають максимальне значення величин, що зустрічаються в цій задачі, обмежені зверху поліномом від довжини входу.

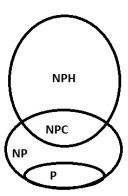


Рисунок 15.1 – Співвідношення між класами P, NP, NPC, NPH

15.1.2. Приклади NP-повних задач

Значення класу NP-повних проблем полягає ще й у тому, що йому належать дуже багато відомих і важливих у прикладному відношенні задач.

Проблемі доведення повноти цих задач присвячені дослідження багатьох видатних вчених з теорії алгоритмів. Будь-які доведення NP-повноти складні, але процедуру доведення можна спростити, застосувавши поняття зведення.

Для цього достатньо показати, що будь-яка з відомих NP-повних задач може бути зведена до цієї задачі.

Наприклад, Річард Карп у своїй праці «Зводимість між комбінаторними задачами» опублікував цілий список, що складається з формулювання та доведення NP-повноти 21 задачі. Перелічемо деякі з них.

1. *Задача розбиття*

Задана множина додатних цілих чисел $x_1, x_2, ..., x_n$. Чи можна розбити її на дві підмножини так, щоб суми чисел в обох підмножинах були однаковими?

2. *Задача ізоморфізму підграфа*

Нехай є два графи G_1 і G_2. Множину вершин першого графа позначимо V_1, а другого V_2. Нехай | V1 |> | V2 |. Потрібно відповісти на запитання: чи знайдеться в графі G_1 підграф H, ізоморфний графу G_2?

3. *Задача виконуваності булевих формул (SAT).*

Задача виконуваності булевих формул (SAT) важлива для теорії обчислювальної складності. Об'єктом задачі SAT є булева формула, що складається тільки з імен змінних, дужок та операцій $\wedge$(І), $\vee$(АБО) і ¬(НІ). Задача полягає в такому: чи можна призначити всім змінним, що зустрічаються у формулі, значення «ІСТИННІСТЬ» і «ХИБНІСТЬ» так, щоб формула набула значення «ІСТИННІСТЬ»?

4. *Задача про кліку.*

Маємо граф G з m вершинами і ціле додатне число n. Граф називається клікою, якщо кожна вершина в ньому зв'язана ребром з кожною. Кількість вершин у кліці назвемо її потужністю. Чи правда, що в даному графі G є кліка потужності не менше ніж n?

5. *Задача про гамільтонів цикл.*

Маємо граф G з n вершинами. Чи існує у графі простий цикл, що проходить через усі вершини графа? Простим називається цикл, у якому вершини не повторюються. Таким чином, гамільтонів цикл це послідовність вершин і ребер (дуг) графа, що містить усі вершини графа G по одному разу, але щодо ребер обмежень не встановлено, то може бути, що містить не всі дуги.

6. *Задача комівояжера.*

Нехай є граф G з n вершинами. Кожному ребру графа приписано додатне ціле число d_i, що задає довжину ребра. Крім цього, задано деяке додатне ціле число L. Потрібно відповісти на запитання: чи знайдеться у графі G маршрут, що проходить через усі вершини графа G такий, що його довжина не перевищує L?

7. *Вершинне покриття.*

Задано граф G з m вершинами і ціле додатне число n. Вершинним покриттям називається підмножина U $\subseteq$ V множини V вершин графа така, що будь-яке ребро графа G інцидентне хоча б однієї з вершин множини U (друга вершина ребра може або належати, або не належати множині U). Чи існує вершинне покриття, що має в собі не більше ніж n вершин?

15.2. Алгоритмічно нерозв'язні проблеми

За час свого існування людство придумало безліч алгоритмів для вирішення різноманітних практичних і наукових проблем. Поставимо запитання: а чи існують такі проблеми, для яких неможливо придумати алгоритми їх вирішення?

Твердження про існування алгоритмічно нерозв'язних проблем є дуже переконливим, адже - ми констатуємо, що ми не знаємо відповідного алгоритму, не тільки зараз а й ми не можемо принципово ніколи його знайти. Виявляється, що існують такі класи задач, для розв'язання яких немає і не може бути єдиного універсального прийому. Проблеми розв'язання такого роду задач називають алгоритмічно нерозв'язними проблемами. Однак алгоритмічна нерозв'язність проблеми розв'язання задач того чи іншого класу зовсім не означає неможливість розв'язання будь-якої конкретної задачі з цього класу. Мова йде про неможливість вирішення всіх завдань даного класу одним і тим самим прийомом. *Алгоритмічна нерозв'язність* - найважливіша властивість деяких класів коректно поставлених завдань, що допускають застосування алгоритмів, яка полягає в тому, що задача кожного з цих класів у принципі не має загального, універсального алгоритму розв'язання, що об'єднує цей клас.

Алгоритмічна нерозв'язність не є проблемою теорії алгоритмів або невдачею, вона є науковим фактом. Популярність досліджень алгоритмічної нерозв'язності не поступається будь-яким іншим дослідженням у теорії алгоритмів і потребує іноді значних зусиль. Наприклад, доведення алгоритмічної нерозв'язності 10-ї проблеми Гільберта, одержане Ю. В. Матіясевичем, відносять до видатних наукових досягнень XX століття.

Знання можливої алгоритмічної нерозв'язності має в галузі комп'ютерних технологій таке саме значення, як для фізика знання про неможливість створення вічного двигуна. Припустимо, що до Білла Гейтса приходить винахідник з пропозицією надати йому грант (грошові інвестиції) для написання програми, яка приймає на вхід будь-яку програму і видає відповідь, потрапить ця програма в нескінченний цикл чи ні за будь-яких вхідних даних. Б. Гейтс йому, звісно, відповідь, оскільки знає, що ця задача є алгоритмічно нерозв'язною.

Означення 15.4. *Алгоритмічною проблемою називається проблема побудови алгоритму, що володіє тими чи іншими властивостями.*

Ми отримаємо вирішення алгоритмічної проблеми, якщо або знайдемо шуканий алгоритм (тобто подано його опис), або доведем, що такого алгоритму не існує.

Однією з найбільш загальних теорем теорії алгоритмів, що пояснює природу багатьох проблем у теорії алгоритмів, є теорема Райса.

Уведемо деякі визначення.

Означення 15.5. *Характеристичною функцією множини A будемо називати функцію*

$$\chi_A(x) = \begin{cases} 1, якщо\ x \in A, \\ 0, якщо\ x \notin A \end{cases} \qquad (15.1)$$

Множина A називається рекурсивною, якщо її характеристична функція рекурсивна.

Означення 15.6. *Нехай Q – деяка властивість одномісних частково-рекурсивних функцій. Властивість Q називається нетривіальною, якщо існують функції, що володіють цією властивістю, так і функції, що не володіють*.

Усі властивості, що розглядаються в теорії алгоритмів, зазвичай є нетривіальними. Наприклад, властивості функцій : усюди визначеність, взаємна однозначність, тотожна рівна нулю та інші.

Ураховуючи те, що частково-рекурсивні функції можна задати програмою їх обчислення, виникає питання: чи можливо за програмою визначити, чи має відповідна функція певну нетривіальну властивість?

Відповідно до тез Черча і Тюрінга задача є алгоритмічно розв'язуваною тоді і тільки тоді, коли існує деяка машина Тюрінга M, що розв'язує цю задачу. За поставленим завданням необхідно визначити, чи характерна функції $f_M(x)$, що реалізується машиною M, властивість Q. Функцію $f_M(x)$ будемо розуміти як функцію, що відповідає програмі з номером Геделя *x*.

Теорема Райса. *Якою б не була нетривіальна властивість Q одномісних частково-рекурсивних функцій, задача розпізнання цієї властивості алгоритмічно нерозв'язна, тобто не існує машини Тюрінга M,що розв'язує цю задачу*.

Наслідки з теореми Райса. *Неможливо за допомогою універсального алгоритму перевірити всюди визначеність функції $f_M(x)$; не існує алгоритму перевірки того, що програма обчислить нульову функцію; питання про те, обчислюють чи ні дві програми одну й ту саму функцію, масово нерозв'язна; не існує алгоритму, що визначає за текстом програми, чи буде ця програма обчислювати деяку конкретну обчислювальну функцію.*

Теорема Райса є однією з найбільш загальних теорем теорії алгоритмів, що пояснює природу багатьох проблем нерозв'язності у практиці програмування.

Першою фундаментальною теоретичною працею, пов'язаною з доведенням алгоритмічної нерозв'язності, була робота Курта Геделя його відома теорема про неповноту символічних логік. Це була строго сформульована математична проблема, для якої не існує розв'язного її алгоритму. Зусиллями різних дослідників список алгоритмічно нерозв'язних проблем був значно розширений. Сьогодні прийнято під час доведення алгоритмічної нерозв'язності деякої проблеми зводити її до класичної задачі «задачі зупину».

Теорема 15.3. *Не існує алгоритму (машини Тюрінга), що дозволяє за описом довільного алгоритму і його вихідних даних (і алгоритм, і дані задані символами на рядку машини Тюрінга) визначити, зупиняється цей алгоритм на цих даних чи працює нескінченно.*

Таким чином, фундаментально алгоритмічна нерозв'язність пов'язана з нескінченністю виконуваних алгоритмом дій, тобто неможливістю передбачити, що для будь-яких вихідних даних розв'язок буде отриманий за кінцеву кількість кроків. Тим не менше, можна спробувати сформулювати причини, що призводять до алгоритмічної нерозв'язності, ці причини достатньо умовні, оскільки всі вони зводяться до проблеми зупину, однак такий підхід дозволяє більш глибоко зрозуміти природу алгоритмічної нерозв'язності.

Приклад 15.1. *Розподіл дев'яток у запису числа π*

Визначимо функцію f (n) = i, де n - кількість дев'яток поспіль у десятковому записі числа π, а i - номер найлівішої дев'ятки з n дев'яток поспіль:

π = 3,141592 ... f (1) = 5.

Завдання полягає в обчисленні функції f (n) для довільно заданого n.

Оскільки число π є ірраціональним і трансцендентним, то ми не знаємо жодної інформації про розподіл дев'яток (так само як і будь-яких інших цифр) у десятковому записі числа. Обчислення f

(n) пов'язане з обчисленням наступних цифр у розкладанні, доти, поки ми не виявимо n дев'яток поспіль, однак у нас немає загального методу обчислення f (n), тому для деяких n обчислення можуть тривати нескінченно - ми навіть не знаємо в принципі (за природою числа π), чи існує розв'язок для всіх n.

Приклад 15.2. *Десята проблема Гільберта*

Вона полягає у знаходженні універсального методу цілочислового розв'язання довільного алгебраїчного діофантового рівняння. Доведення алгоритмічної нерозв'язності цієї задачі зайняло близько двадцяти років і було завершене Ю. В. Матіясевичем у 1970 році.

Формально мова йде про цілочислове розв'язання рівнянь вигляду

$P(x_1, x_2, \ldots, x_n) = 0$, де P - многочлен із цілими коефіцієнтами й цілими показниками степенів. Доведено, що такого алгоритму не існує, тобто відсутній загальний метод визначення цілих коренів цього рівняння.

- Запитання для самоперевірки:

Запитання для самоперевірки:

1. У чому суть теореми Блюма про прискорення?
2. Алгоритмічні зведення за Карпом, Куком, Тюрінгом. Яка між ними різниця і що є однаковим?
3. Як довести, що завдання є NP-повним?
4. Що означає, що одна задача зводиться до іншої за поліноміальний час?
5. Як довести, що завдання A зводиться (за поліноміальний час) до задачі B?
6. Дайте означення алгоритмічної нерозв'язності.
7. Що називається характеристичною функцією множини A?
8. Сформулюйте теорему Райса та її наслідки.

- Задачі для самостійного розв'язання

Задачі для самостійного розв'язання

1. Якщо задача $A \in NP$ і $A \le_P B$, то чи можна стверджувати, що $B \in NP$?
2. Якщо задача $A \in NP$ і $B \le_P A$, то чи можна стверджувати, що $B \in NP$?
3. Якщо $A \in P$, то чи можна стверджувати, що $A \le_P B$ для будь-якої задачі B? Доведіть.

Коментарі.

Основні відомості, щодо класів складності та проблеми зведення представлені в [13], алгоритмічна нерозв'язність викладена в працях [5,9,11,24,28].