МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Оксана Безущак
Олександр Ганюшкін

МАТЕМАТИЧНА ЛОГІКА

Навчальний посібник

КИЇВСЬКИЙ УНІВЕРСИТЕТ 2023

УДК 510.6:164(075.8)
Б40

Рецензенти:
д-р фіз.-мат. наук, проф., чл.-кор. НАН України Олександр Бойчук,
д-р фіз.-мат. наук, проф. Олександр Станжицький

Рекомендовано до друку
вченою радою механіко-математичного факультету
(*протокол № 2 від 15 вересня 2022 року*)

Ухвалено науково-методичною радою
Київського національного університету імені Тараса Шевченка
(протокол № 3-23 від 30 березня 2023 року)

Безущак Оксана

Б40 Математична логіка : навч. посіб. / Оксана Безущак, Олександр Ганюшкін. – К. : ВПЦ "Київський університет". – 2023. – 143 с.

ISBN 978-966-933-248-6

Викладено основний матеріал із дисципліни "Математична логіка" в передбаченому робочими програмами обсязі. В основу посібника покладено лекції, які читають на механіко-математичному факультеті. Проте видання не можна ототожнювати з конспектом лекцій. Його функції значно ширші.

Для студентів і викладачів закладів вищої освіти, а також усіх, кого цікавить математична логіка.

УДК 510.6:164(075.8)

ISBN 978-966-933-248-6

Зміст

Вступ

Коли говорять про предмет логіки, то часто кажуть, що це *наука про закони мислення*. Означення гарне, лишається тільки з'ясувати, що таке наука, що таке закони і що таке мислення.

Наведене означення майже нічого не прояснює і його слід сприймати дуже обережно — межі компетенції логіки насправді значно вужчі. За цими межами лишаються, наприклад, історичні, психологічні й інші подібні закономірності процесу мислення. Логіку цікавлять лише *формальні, структурні* властивості цього процесу, які, можливо, відображають деякі властивості реального мислення. І не всього мислення (напр., за межами логіки лишається образне, асоціативне мислення митців), а лише так званих "правильних міркувань", адже нам часто доводиться переконувати співрозмовника в істинності того чи іншого твердження. Логіка виникла з вивчення тих сторін мови, які суттєві для цієї мети. Виявилося — і в цьому й полягає сила логіки — що про коректність того чи іншого міркування (зокрема математичного доведення) часто можна судити виключно на підставі форми тверджень, що входять до цих міркувань. Тому логіка цікавиться лише формою міркувань, тобто способами переходу від одних тверджень (які називають *засновками*) до інших (які називають *висновками*). "Правильність" міркувань означає, що з правильних засновків мають одержувати правильні висновки[1].

Зокрема, одним із завдань логіки є систематизація і формалізація тих способів міркувань, які гарантують істинність висновку, якщо всі засновки — істинні. А самі по собі істинність чи хибність окремих засновків і висновків, їхній конкретний зміст логіку не цікавлять.

Якщо ж у першу чергу досліджують математичні міркування (трохи ширше — міркування, що використовують у точних науках), а в ході досліджень застосовують математичний апарат, то таку науку називають *математичною логікою*.

Оскільки наш курс невеликий, то ми звузимо предмет математичної логіки ще більше. Нашою головною метою буде намагання з'ясувати точний зміст двох центральних математичних понять: що таке "доведення" (як стверджують Бурбакі, ще із часів давніх греків говорити "математика" означає говорити "доведення") і що таке "алгоритм". У повному обсязі ці поняття належать математиці не більше, ніж, скажімо, психології. Адже доведення, на-

[1] Зауважимо однак, що схильність до правильних міркувань не дуже властива людям. Як писав данський вчений Х. І. Ульдалль, "Логічне мислення ... схоже швидше на танці коней, тобто на трюк, якому декого таки можна навчити, але далеко не всіх, причому цей трюк може виконуватися лише з великими зусиллями і з різним рівнем майстерності, і навіть кращі представники не у змозі повторити його багато разів підряд".

приклад, це просто міркування, здатне переконати нас настільки, що ми готові за його допомогою переконувати інших. Але інтуїтивного розуміння понять доведення й алгоритму виявилося для математики не досить. Ще в XIX ст. після двохтисячолітніх марних спроб виникла підозра про неможливість доведення певних тверджень (напр., знаменитого п'ятого постулату Евкліда про паралельність) чи неіснування певних алгоритмів (напр., трисекції кута за допомогою циркуля та лінійки). І хоча для цих двох конкретних проблем неіснування доведення/алгоритму зрештою вдалося довести[2], список подібних проблем швидко зростав.

Серед математиків ще довго панувало переконання, що кожну таку проблему рано чи пізно вдасться розв'язати. Найафористичніше це переконання сформулював Гільберт: "Ми хочемо це знати — ми будемо це знати". Однак поступово прийшло розуміння одного принципового моменту. Переконливим свідченням існування певного доведення (алгоритму) служить демонстрація самого цього доведення (алгоритму). У той же час неможливість пред'явити конкретне доведення чи алгоритм може свідчити не про їхнє неіснування, а лише про нашу нездатність знайти їх (через обмеженість наших здібностей, надзвичайну складність доведення тощо). Для доведення неіснування чогось спочатку треба чітко і недвозначно вказати, неіснування чого ми хочемо довести. Указані міркування привели до потреби у строгому означенні понять доведення й алгоритму.

Щодо самої структури навчального посібника, то він розбитий на розділи, кожен з яких містить підрозділи, які, своєю чергою, — пункти. Для посилання на теорему (твердження, лему, наслідок, означення, вправу) використовується подвійна нумерація: перше число є номером розділу, до якого ця теорема (твердження, лема, наслідок, означення, вправа) належить, а друге — її наскрізним номером у цьому розділі незалежно від підрозділу. Важливо звернути вашу увагу на деякі особливості тексту. Частини матеріалу представлено меншим шрифтом, оскільки містять додаткову інформацію, що сприяє глибшому розумінню відповідної теми. Ці відомості можуть бути корисними для більш високого рівня математичних концепцій і застосувань. При першому читанні посібника ви можете вирішити пропустити цю інформацію. Проте ми рекомендуємо повертатися до неї, коли ви будете готові для глибшого розуміння відповідної теми. Якщо додаткова інформація винесена як окремий підрозділ, то назва цього підрозділу позначена зірочкою.

[2] У першому випадку була побудована нова геометрія, в якій п'ятий постулат *a priori* не виконувавсь і яка була настільки ж несуперечливою, наскільки несуперечливою є класична евклідова геометрія. У другому випадку вдалося в алгебричних термінах описати всі геометричні конструкції, які можна побудувати за допомогою циркуля та лінійки, і трисекції кута серед них не виявилося.

Текст навчального посібника містить невелику кількість вправ, які є обов'язковими (зазвичай вони прості й даються для закріплення наведених перед ними понять) та розраховані на активну участь читача в опановуванні матеріалу.

Список літератури не претендує на повноту і містить лише деякі доступні підручники, що можна використати для глибшого розуміння матеріалу.

Нехай цей посібник стане вашим надійним провідником у світ математичної логіки. Бажаємо успіхів у навчанні та відкритті нових горизонтів математичних знань!

1. Логіка висловлювань

1.1. Висловлювання

Висловлювання — це певні твердження (тобто розповідні речення) про певні об'єкти, за змістом яких доречно ставити питання про їхню істинність чи хибність, причому відповідь на це питання має бути однозначною (хоча, можливо, і невідомою нам). Зокрема, більшість речень, що зустрічаються в математичних текстах, є висловлюваннями. Не є ними означення, запитання, наказові й окличні речення тощо[3].

Сказане не слід сприймати за означення висловлювання. *Логіку висловлювань* будують таким же чином, як і багато інших розділів математики (алгебра, геометрія, теорія ймовірностей тощо). В основу кладуть деякий клас об'єктів (у нашому випадку — *прості висловлювання*) разом із деяким набором властивостей і відношень між ними. Зокрема, ми дотримуємося так званого *принципу двозначності суджень*:

Кожне висловлювання, в якому сформульовано певну чітко виражену думку, є або ***істинним****, або* ***хибним****, незалежно від того, чи стосується воно речей і подій відомих, чи невідомих, минулих, майбутніх чи теперішніх. Жодне висловлювання не може бути одночасно і хибним, й істинним*[4].

Наведені поняття є вихідними, первісними[5] і всередині цього розділу математики не вимагають подальших означень (можуть лише роз'яснюватися на прикладах). Звичайно, вихідні властивості і відношення вибирають так, щоб

[3] Точніше, висловлюваннями є не самі розповідні речення, а їхній зміст. Те саме висловлювання може бути виражене різними розповідними реченнями.

[4] Так є в математиці, принаймні, у класичній. Однак у повсякденному житті з поділом висловлювань на істинні й хибні трохи гірше. Яка Ваша думка з приводу істинності висловлювання "Зараз на вулиці хороша погода"?

[5] Ми лишимо філософам з'ясовувати глибинну суть таких понять, як істина і хибність. Для нас досить, що вони можуть бути значеннями істинності висловлювань.

вони відповідали змістовній практиці, яку ця математична теорія описуватиме. Зокрема і логіка створювалася для дослідження висловлювань (і певних маніпуляцій із ними — так званих міркувань) у живих, неформальних мовах. Але у процесі формалізації довелося абстрагуватися від конкретного змісту висловлювань і перейти до формальних мов, в яких висловлювання ототожнюються з певними послідовностями символів (так званими *правильно побудованими формулами*) і визначаються, таким чином, синтаксисом цих мов.

Якщо висловлювання істинне, то кажуть, що *значенням істинності* (або *логічним значенням*) цього висловлювання є "істина" (і позначають це значення символом "**i**", або "**t**" — від англійського *true*, або "1"). Аналогічно значення істинності хибного висловлювання позначають символом "**x**" (або "**f**" — від англійського *false*, або "0").

Логіка висловлювань вивчає лише ті їхні властивості, які можуть бути сформульовані лише в термінах їхньої істинності чи хибності (і не апелюють до змісту висловлювань). Переваги такого підходу ми зможемо оцінити вже в наступному підрозділі, вивчаючи дії над висловлюваннями (особливо добре це буде видно на прикладі імплікації).

Зауваження. *Спочатку ідея позначення значень істинності висловлювань числами може видатися досить безглуздою: із числами можна виконувати різні дії, а який зміст можуть мати ці дії для значень істинності? Виявляється, що можуть. Частково ми в цьому переконаємося пізніше.*

За текстом самого висловлювання, взагалі кажучи, ще не можна встановити його істинності. Треба ще врахувати контекст (або точніше: *зафіксувати ситуацію*). Наприклад, щоб визначити істинність висловлювання "на вулиці йде дощ", треба знати час і місце події. Не завжди допомагає і контекст: наївне переконання, що кожному розповідному реченню можна розумним (принаймні, несуперечливим) чином приписати певне значення істинності, спростовується так званим "парадоксом брехуна". Якщо хтось говорить: "Розповідне речення, яке я зараз виголошую, є хибним висловлюванням", то спроба приписати цьому реченню певне значення істинності негайно приводить до суперечності. Справді, якщо припустити, що воно істинне, то внаслідок свого власного змісту воно повинно бути хибним, і навпаки[6]. Тому такі речення ми не вважаємо висловлюваннями.

[6] "Парадокс брехуна" лежить в основі кількох знаменитих теорем математичної логіки (зокрема і теореми Ґеделя про неповноту арифметики, і теореми Тарського про невиразність поняття істини).

Навіть коли ми переконані, що маємо справу з висловлюванням, ми не завжди можемо встановити його значення істинності. Встановлення цього значення може вимагати від нас нереальних матеріальних витрат чи інтелектуальних зусиль. До перших належать висловлювання типу "На найближчій планетній системі існують органічні форми життя", до других — багато математичних гіпотез. Лише кілька прикладів:
— існують непарні досконалі числа;
— існує нескінченно багато простих "чисел-близнюків";
— кожне більше за 2 парне число є сумою двох простих чисел;
— "$\mathcal{P} \neq \mathcal{NP}$".

Більше того, для багатьох математичних теорій, зокрема і для арифметики, доведено неіснування алгоритмів, які дозволяли б установлювати логічні значення довільних висловлювань цих теорій (навіть якщо знехтувати будь-якими обмеженнями на час роботи, об'ємом пам'яті тощо)[7].

Висловлювання, які істинні в усіх можливих ситуаціях, називають *абсолютно істинними*. Аналогічно визначають *абсолютно хибні* висловлювання. Абсолютно істинні й абсолютно хибні висловлювання називають ще *логічними константами*.

1.2. Дії над висловлюваннями (логічні зв'язки)

Розповідні речення бувають прості й складні. Прості речення з'єднуються у складні за допомогою різних сполучників і мовних конструкцій. Найуживанішим із таких конструкцій у логіці висловлювань відповідають *дії над висловлюваннями* (їх ще називають *логічними операціями*, *логічними зв'язками* або *логічними функторами*). За допомогою цих дій із простих висловлювань утворюють *складні*[8].

Але конструкції живої мови неоднозначні, їх тлумачення залежить від контексту. Досить порівняти вживання сполучника "або" в реченнях "Студент Логіченко має видатні математичні здібності або він уже розв'язував подібні задачі раніше" і "Студент Логіченко отримає на екзамені з математичної логіки "добре" або "відмінно". Тому, визначаючи логічну дію, з усіх можливих варіантів і відтінків уживання певної граматичної конструкції вибирають і фіксують один. З іншого боку, у логіці висловлювань дозволено будь-які

[7] Однак встановлення значення істинності конкретних висловлювань і не входить у компетенцію логіки висловлювань.

[8] Під простими висловлюваннями ми розумітимемо такі висловлювання, внутрішня структура яких нас зовсім не цікавить (принаймні, в межах логіки висловлювань). Нам треба лише вміти розпізнавати і розрізняти їх. Тому далі прості висловлювання ми позначатимемо малими буквами латинського алфавіту (зазвичай з його першої половини).

граматично правильні способи творення складних речень. Вимагається лише, щоб за формою відповідної конструкції і логічними значеннями простих висловлювань-компонент логічне значення утвореного складного висловлювання встановлювалося однозначно. Причому це значення повинно залежати саме від логічних значень простих компонент, а не від їхнього змісту[9].

Отже, логічна зв'язка повністю визначається своєю *таблицею істинності* (ця таблиця вказує, яких логічних значень набуває складне висловлювання для різних наборів логічних значень простих висловлювань, що до нього входять).

Розглянемо найуживаніші логічні операції. Найпростішими з них є такі.

Заперечення. Ця операція утворена на основі звороту "неправильно, що ..." і позначається символом $\neg$. Логічне значення $\neg a$ (читають "не a") характеризується такою таблицею істинності:

a	0	1
$\neg a$	1	0

.

Кон'юнкція. Ця операція утворена на основі сполучника "і"; позначається символом $\wedge$. Логічне значення виразу $a \wedge b$ (читають "a і b") визначається такою таблицею істинності:

a	0	0	1	1
b	0	1	0	1
$a \wedge b$	0	0	0	1

.

У живій мові кон'юнкція може виражатися й іншими сполучниками та конструкціями: "а" ("На городі бузина, а в Києві дядько"), "але", "хоча" ("Він зайшов у річку, хоча й не вмів плавати"), "незважаючи на те, що", ..., і навіть взагалі без сполучника: "На городі бузина, у Києві дядько". У живій мові ці сполучники не є тотожними, вони мають різні смислові відтінки, які в реальних життєвих обставинах можуть бути дуже важливими. Однак у логіці висловлювань ми ці відтінки не розрізняємо.

Диз'юнкція та альтернатива. Обидві ці операції утворені на основі сполучника "або": *диз'юнкція* відповідає з'єднувальному "або" і позначається символом $\vee$ (вираз $a \vee b$ читають "a або b")[10]; *альтернатива* відповідає розділовому "або" і позначається символом $\oplus$ (вираз $a \oplus b$ читається "або a, або b"). Ці операції характеризуються такими таблицями істинності:

[9] У цьому сенсі логіку висловлювань можна вважати теорією логічних зв'язок.

[10] Символ $\vee$ походить від першої літери латинського слова *vel*, яке якраз і має значення з'єднувального *або*.

a	0	0	1	1
b	0	1	0	1
$a \vee b$	0	1	1	1

,

a	0	0	1	1
b	0	1	0	1
$a \oplus b$	0	1	1	0

.

Еквіваленція. Ця операція утворена на основі мовних конструкцій: "... тоді й лише тоді, коли ...", "... є необхідною і достатньою умовою для ...", "... рівносильне ..." тощо. Еквіваленцію позначають символом $\leftrightarrow$ (вираз $a \leftrightarrow b$ читають як "a еквівалентне b") і задають такою таблицею істинності:

a	0	0	1	1
b	0	1	0	1
$a \leftrightarrow b$	1	0	0	1

.

Імплікація. Ця операція утворена на основі звороту "якщо ..., то ..." і позначається символом $\rightarrow$. Логічне значення виразу $a \rightarrow b$ (читають як "a імплікує b" або "якщо a, то b") визначається такою таблицею істинності:

a	0	0	1	1
b	0	1	0	1
$a \rightarrow b$	1	1	0	1

.

Висловлювання a називають *умовою, засновком* або *антецедентом* імплікації, а висловлювання b — її *наслідком* або *консеквентом*.

Проте відповідність між уживанням у живій мові звороту "якщо ... , то ..." та імплікацією як дією над висловлюваннями дуже неповна. У першу чергу це пов'язано з тим, що в мовній практиці зазвичай не використовують складні речення вигляду "якщо a, то b" тоді, коли про речення a напевно відомо, що воно є хибним. Вважають, що такі речення безглузді. І справді, чому ми повинні вважати речення вигляду "якщо $2 \times 2 = 5$, то $3 \times 7 = 9$" або "якщо на Північному полюсі ростуть ананаси, то $2 \times 2 = 2$" не тільки не позбавленими глузду, а навіть істинними?

Але така позиція не може бути прийнятною в математиці. Наприклад, один із найпоширеніших методів доведення — від супротивного — ґрунтується якраз на виводі наслідків із твердження, в хибності якого ми впевнені. З іншого боку, багато важливих теорем теорії чисел мають вигляд "якщо гіпотеза Рімана правильна, то справедливе твердження b", тобто вигляд імплікацій $a \rightarrow b$, де a — гіпотеза Рімана. Однак досі не відомо, чи справді ця гіпотеза є правильною. Зустрічається в математиці й ситуація, коли ту саму аксіому в

одній теорії вважають істинною, а в іншій — хибною (досить згадати аксіому паралельності в геометрії або аксіому вибору в теорії множин).

Ще два аргументи на користь загальноприйнятої таблиці істинності для імплікації:

a) умовні обіцянки: "якщо складеш екзамен з логіки, то вийду за тебе заміж". Обіцянку можна вважати порушеною (хибною) лише в тому випадку, коли умова виконана (істинна), а друга частина — ні;

b) математичні міркування типу "якщо n ділиться на 6, то n ділиться на 3". Оскільки ми вважаємо таке міркування істинним, то маємо вважати його істинним і для конкретних n. Якщо взяти, наприклад, $n = 9, 10, 12$, то виходить, що для кожного з наборів значень істинності $(0,1)$, $(0,0)$ і $(1,1)$ висловлювань $a =$ "n ділиться на 6" і $b =$ "n ділиться на 3" імплікація $a \to b$ повинна бути істинною.

Отже, якщо ми хочемо, щоб імплікація $\mathbf{t} \to \mathbf{f}$ була хибною і щоб математичні міркування типу "якщо дане число ділиться на 6, то воно ділиться на 3" вважалися законними, то таблицю істинності для імплікації слід визначити однозначно.

У зв'язку із цим у жодному разі не можна розглядати імплікацію як твердження про те, що засновок є причиною наслідку (у тому сенсі, як це прийнято у природничих науках). Дотримуючись нашого означення, імплікацію "якщо в зайця куций хвіст, то головною посадовою особою у структурі є її керівник" слід визнати істинною, бо істинним є висновок імплікації: "головною посадовою особою у структурі є її керівник". Але при розумінні засновку як причини (як це розуміють у природничих науках чи — більш широко — у повсякденній практиці) такий висновок жодним чином не випливає із засновку (хоча останній також істинний). Адже між ними немає жодного причинного зв'язку. Розуміння імплікації як причинно-наслідкового зв'язку між засновком і висновком не може бути реалізоване засобами логіки висловлювань, тому що його не можна сформулювати лише в термінах істинності-хибності.

У звичайній мові зворот "якщо ... , то ..." може вживатися і в інших значеннях: для позначення послідовності подій у часі або просторі, для позначення зв'язку мети і засобів ("якщо хочеш рибки, то лізь у воду"), для позначення умовної домовленості ("якщо складеш екзамен, то отримаєш стипендію") тощо, у кожному з яких він має свою специфіку. Однак в імплікації $a \to b$ ми абстрагуємося від природи зв'язку між a та b, і надаємо імплікації тільки того змісту, який виражається таблицею істинності.

Кожну логічну зв'язку ми характеризували певною таблицею істинності. Але кожну таку таблицю можна розглядати як табличне задання деякої функції, визначеної на множині логічних значень $B = \{0, 1\}$ (як у випадку заперечення) або на декартовому квадраті $\{0, 1\} \times \{0, 1\}$ цієї множини (як в інших

випадках) і яка набуває значень із цієї ж множини. Такий підхід природно узагальнюється на довільну кількість аргументів і дозволяє ототожнювати логічні операції з функціями вигляду $\{0,1\}^n \to \{0,1\}$. Зокрема, він дозволяє вводити нові операції, вже не спираючись безпосередньо на мовну практику. Приклади таких операцій наведемо згодом.

Число n називають *арністю* операції. При $n = 1$ операцію називають *унарною*, при $n = 2$ — *бінарною*, при $n = 3$ — *тернарною*.

Для тих логічних зв'язок, що ми розглянули, в літературі зустрічаються й інші позначення. Наведемо найпоширеніші з них:

для заперечення — N, $^-$, $\sim$, C;

для диз'юнкції — D, $+$;

для кон'юнкції — C, $\&$, $\cdot$;

для імплікації — $\supset$, $\Rightarrow$;

для еквіваленції — $\sim$, $\equiv$, $\Leftrightarrow$.

Символи $\Rightarrow$, $\Leftrightarrow$ і $\equiv$ ми також використовуватимемо, але не для позначення операцій, а з іншою метою.

Підсумовуючи, наведемо таблицю всіх тих бінарних зв'язок, в яких результат залежить від обох аргументів. Для тих зв'язок, які ще не зустрічалися, наводимо також їхні найпоширеніші назви і позначення:

a	b	$a \wedge b$	$a \not\to b$	$a \not\leftarrow b$	$a \downarrow b$	$a \leftrightarrow b$	$a \oplus b$	$a \vee b$	$a \leftarrow b$	$a \to b$	$a \mid b$
0	0	0	0	0	1	1	0	0	1	1	1
0	1	0	0	1	0	0	1	1	0	1	1
1	0	0	1	0	0	0	1	1	1	0	1
1	1	1	0	0	0	1	0	1	1	1	0

$a \leftarrow b$ — *зворотна* (або *обернена*) *імплікація*; $a \not\leftarrow b$ — *зворотна* (або *обернена*) *антиімплікація*; $a \mid b$ (або $a \uparrow b$) — *антикон'юнкція* (або *штрих Шефера*, або *несумісність*); $a \not\to b$ — *антиімплікація*; $a \downarrow b$ — *антидиз'юнкція* (або *стрілка Пірса*, або *стрілка Лукасевича*).

1.3. Формули

1.3.1. Поняття формули

За допомогою логічних операцій далі можемо будувати як завгодно складні висловлювання. Записувати їх будемо у вигляді *формул* логіки висловлювань. Інтуїтивно зрозуміло, що таке формули, але щоб їх можна було вивчати, корисно мати строге означення. Для цього перш за все фіксуємо *алфавіт*[11],

[11] *Алфавітом* називається довільна множина попарно різних символів. Елементи алфавіту часто називають *буквами*.

за допомогою якого записуватимемо формули. Алфавіт логіки висловлювань складається із символів трьох типів:

a) малі літери латинського алфавіту (можливо, з індексами), які ми називатимемо *пропозиційними змінними* (або просто *змінними*), і символи $0, 1$ — для позначення простих висловлювань;

b) символи $\neg$, $\vee$, $\wedge$, $\oplus$, $\to$, $\leftrightarrow$ логічних операцій;

c) допоміжні символи — ліва дужка "(" і права дужка ")" — для фіксації порядку виконання дій.

Формулами логіки висловлювань будуть певні слова (тобто послідовності символів) у цьому алфавіті. Точніше:

a) кожен символ для позначення простих висловлювань є формулою;

b) якщо A і B — формули, то слова $(\neg A)$, $(A \vee B)$, $(A \wedge B)$, $(A \oplus B)$, $(A \to B)$, $(A \leftrightarrow B)$ також будуть формулами;

c) слово буде формулою тоді й лише тоді, коли його можна одержати з простих висловлювань, застосовуючи скінченну кількість разів правила з пункту b).

Отже, строге означення формули має *рекурсивний* характер: спочатку вказують деякі вихідні формули (пункт a), а потім формулюють правила, які дозволяють з уже побудованих формул будувати нові (пункт b).

Рекурсивні означення досить поширені в математиці. Зокрема, вони неодноразово зустрічатимуться і в нас. Такі означення конструктивні (напр., неважко побудувати алгоритм, який для кожної послідовності символів розпізнає, чи є вона формулою логіки висловлювань, а потім запрограмувати його і передовірити перевірку властивості "бути формулою" комп'ютеру). Крім того, вони добре пристосовані для доведень за індукцією (напр., щоб довести, що всі формули мають певну властивість, досить перевірити цю властивість для найпростіших формул, а потім показати, що кожне правило побудови нових формул цю властивість зберігає).

Логічну зв'язку, яка для побудови формули використовувалася останньою, називають *головною зв'язкою* формули. Кількість зв'язок у формулі називають її *довжиною*. Якщо формула A має вигляд $A = A_1 A_2 A_3$, де слово A_2 також є формулою, то A_2 називають *підформулою* формули A.

Нагромадження дужок у великих формулах часто робить їх важкими для сприйняття. Уникнути цього нагромадження можна за рахунок домовленості про порядок виконання дій (подібно тому, як це роблять в арифметиці). Для цього кожній зв'язці приписують певний ранг:

$$\neg, \quad \wedge, \quad \vee, \quad \oplus, \quad \to, \quad \leftrightarrow$$

(зв'язки виписано в порядку убування їхніх рангів), і операції більшого рангу виконують першими. Операції однакового рангу виконують зліва направо. Дужки використовують головно для того, щоб змінити обумовлений цими домовленностями порядок виконання дій. Зовнішні дужки зазвичай також вилучають[12].

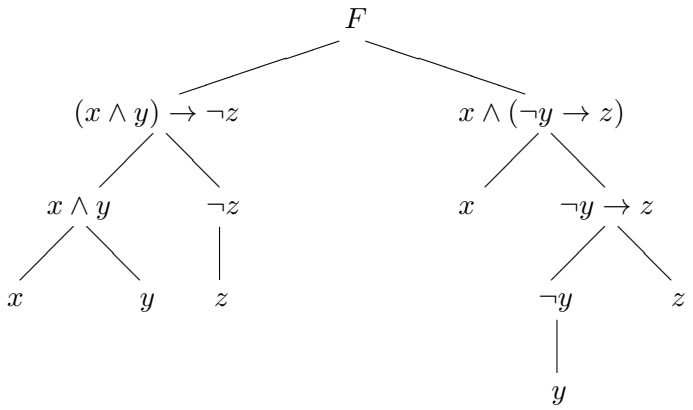
Тому надалі вживатимемо, взагалі кажучи, не формули, а вирази, які відрізнятимуться від формул деякими вольностями (напр., у них може бути пропущена частина дужок). За потреби строго дотримуватися означення ми говоритимемо про *правильно побудовані формули* (скорочено: ппф).

Якщо F_1 і F_2 — дві підформули формули F, то або вони не перетинаються, або одна з них міститься в іншій. Звідси випливає, що підформули відносно включення утворюють дерево (*дерево підформул*) і це дерево визначене однозначно. Дерево підформул формули F є кореневим із коренем F, а висячі вершини відповідають найкоротшим підформулам, тобто простим висловлюванням, що зустрічаються в F.

Для прикладу розглянемо дерево підформул для формули

$$F = ((x \wedge y) \to \neg z) \vee (x \wedge (\neg y \to z))$$

(зовнішні дужки у підформулах вилучаємо):



Зауваження. 1. *Сама по собі формула логіки висловлювань не є істинною чи хибною: це лише побудований за певними правилами рядок символів. Ми одержимо хибне чи істинне висловлювання лише після того, як підставимо*

[12] Не слід зловживати цими домовленостями. Формули, які містять замало дужок, також важкі для сприйняття.

у формулу замість змінних конкретні висловлювання і виконаємо над ними відповідні операції.

2. *Запис $F = F(x_1, \ldots, x_n)$ (або просто $F(x_1, \ldots, x_n)$) означає, що F розглядається як формула від змінних $x_1, \ldots, x_n$. Однак у цьому разі деякі змінні можуть бути фіктивними, тобто явно у формулі не зустрічатися. До того ж у формулі F трапляються й інші змінні, окрім $x_1, \ldots, x_n$. Тому, наприклад, запис $x_1 \vee x_2 = F(x_2, x_3)$ є коректним. Коли пишемо $F = F(x_1, \ldots, x_n)$, то це означає, що в даний момент нас цікавить залежність формули F саме від змінних $x_1, \ldots, x_n$.*

1.3.2. Таблиці істинності

Таблицю істинності можна будувати не тільки для зв'язок, а й для довільних формул логіки висловлювань. Розглянемо, наприклад, побудову таблиці істинності для формули $F = ((x \wedge y) \to \neg z) \vee (x \wedge (\neg y \to z))$. У верхньому рядку таблиці виписуємо всі підформули даної формули у порядку зростання їхньої складності. Зокрема, спочатку виписуються усі прості висловлювання (змінні), а останньою — сама формула. Для змінних розглядаємо всі можливі варіанти значень. Стовпці зручно заповнювати послідовно зліва направо, використовуючи для побудови стовпця з даною підформулою таблицю істинності головної зв'язки цієї підформули.

x	y	z	$x \wedge y$	$\neg z$	$\neg y$	$\neg y \to z$	$(x \wedge y) \to \neg z$	$x \wedge (\neg y \to z)$	F
0	0	0	0	1	1	0	1	0	1
0	0	1	0	0	1	1	1	0	1
0	1	0	0	1	0	1	1	0	1
0	1	1	0	0	0	1	1	0	1
1	0	0	0	1	1	0	1	0	1
1	0	1	0	0	1	1	1	1	1
1	1	0	1	1	0	1	1	1	1
1	1	1	1	0	0	1	0	1	1

Часто таблицю істинності записують значно компактніше: виписують тільки саму формулу і для кожної із змінних, що входять у формулу, виділяють одне з її входжень (напр., перше). Логічні значення змінних виписують під їхніми виділеними входженнями, а логічні значення підформул — під їхніми головними зв'язками (оскільки головна зв'язка підформули визначена однозначно і різні підформули мають різні головні зв'язки, то жодних непорозумінь такий запис не викликає). Таблиці істинності у такій компактній формі введе-

но Постом 1921 р. і названо *таблицями Поста*. Для даної формули F таблиця Поста має вигляд

$((\boldsymbol{x} \wedge \boldsymbol{y}) \to \neg \boldsymbol{z}) \vee (x \wedge (\neg y \to z))$										
0	0	0	1	1	0	1	0	1	0
0	0	0	1	0	1	1	0	1	1
0	0	1	1	1	0	1	0	0	1
0	0	1	1	0	1	1	0	0	1
1	0	0	1	1	0	1	0	1	0
1	0	0	1	0	1	1	1	1	1
1	1	1	1	1	0	1	1	0	1
1	1	1	0	0	1	1	1	0	1

Інколи за побудови таблиці істинності формули F доводиться враховувати не тільки всі ті змінні, які у ній зустрічаються явно, але й деякі інші (у цьому випадку говорять, що додаткові змінні зустрічаються у формулі F неявно). Така необхідність, наприклад, виникне у нас пізніше під час доведення рівносильності формул.

Зауваження. *На таблиці істинності логічних операцій можна дивитися двояко. Можна вважати зв'язки* $\wedge, \vee, \neg, \to, \leftrightarrow, \oplus$ *скороченими позначеннями відомих конструкцій і будувати таблиці, уже маючи перед собою цю інтерпретацію зв'язок (вище ми так і робили). А з другого боку, можна вважати таблиці істинності означеннями операцій* $\wedge, \ldots, \leftrightarrow, \oplus$ *(не спираючись на яку-небудь інтерпретацію цих символів). Цей другий підхід дуже важливий надалі.*

1.3.3. Труднощі перекладу

Як для перекладу з однієї живої мови на іншу, так і для перекладу із звичної розмовної мови на мову формул логіки висловлювань і навпаки немає формальних процедур. Такий переклад є неоднозначним, вимагає аналізу висловлювань не тільки за формою, а й за змістом, і є творчим процесом. Пояснюється це тим, що в живих мовах немає взаємно однозначної відповідності між змістом і формою його вираження: та сама думка може бути висловлена різними способами (синонімія), а залежно від контексту те саме речення може виражати різні думки (омонімія).

Наприклад, залежно від контексту сполучник "або" може означати як диз'юнкцію $\vee$, так і альтернативу $\oplus$. Зворот "якщо .., то ..." може означати як імплікацію $\to$, так і еквіваленцію $\leftrightarrow$ (зокрема, в означеннях: "якщо послідовність має границю, то вона — збіжна") або кон'юнкцію $\wedge$ ("якщо в

планіметрії вивчають плоскі фігури, то в стереометрії — просторові тіла"). Навпаки, імплікація $\to$ в українській мові може виражатися зворотами "якщо ... , то ...", "оскільки ... , то ...", "... , бо ...", "коли ... , тоді ...", "з ... випливає ...", "... тягне за собою ...", "... лише тоді, коли ...", "як (тільки) ... , зараз ..." і т. д. Кон'юнкція $\wedge$ може виражатися зворотами "і", "а", "але", "якщо, то", "хоч", "незважаючи на", ... ; еквіваленція $\leftrightarrow$ може передаватися зворотами "... тоді й лише (тільки) тоді, коли ...", "для ... необхідно й достатньо, щоб ...", "якщо ... , то ...", "... рівносильне (еквівалентне) ...", "... якщо і тільки якщо ..." і т. д.

Схожі приклади можна навести для інших зв'язок.

Для прикладу, як здійснюється переклад на мову формул логіки висловлювань, розглянемо таке речення з Остапа Вишні: "Собак вони своїми іклами одним ударом січуть на бефстроганов, а охотник, як побачить сікача, зараз бере на мушку або дуба, або грушу і сидить там тихий, як горличка".

Спочатку виділимо всі прості висловлювання, які входять до складу даного складного висловлювання. У цьому випадку вважатимемо простими такі висловлювання: "Собак вони (тобто сікачі) своїми іклами одним ударом січуть на бефстроганов", "Охотник побачить сікача", "[Охотник] бере на мушку дуба", "[Охотник] бере на мушку грушу", "[Охотник] сидить тихий, як горличка". Позначимо їх буквами a, b, c, d, e.

Далі потрібно виявити логічні зв'язки, за допомогою яких будується дане складне висловлювання. Сполучник "а" вживають тут у значенні кон'юнкції. Зворот "як ..., зараз ..." можна, очевидно, інтерпретувати як імплікацію. Сполучник "або" тут вживають у розділовому значенні, тому йому відповідає альтернатива. Нарешті, сполучник "і" в кінці речення вживають у значенні кон'юнкції.

Насамкінець треба проаналізувати порядок, за яким дане складне висловлювання будується з простих.

Лише після такого, досить суб'єктивного, аналізу тексту ми можемо написати відповідну формулу:

$$a \wedge (b \to ((c \oplus d) \wedge e)) .$$

1.3.4. Класифікація формул

Упорядковані набори $(\varepsilon_1, \ldots, \varepsilon_n)$ логічних значень називатимемо *булевими*[13] *векторами*, компоненти $\varepsilon_1, \ldots, \varepsilon_n$ — *координатами* булевого вектора, а число n — його *довжиною* або *розмірністю*. Два булеві

[13] На честь англійського логіка Дж. Буля (1815–1864).

вектори $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)$ і $\delta = (\delta_1, \ldots, \delta_n)$ вважають рівними тоді й лише тоді, коли рівні їхні відповідні координати: $\varepsilon_1 = \delta_1, \ldots, \varepsilon_n = \delta_n$.

Нехай $F(a_1, \ldots, a_n)$ — формула логіки висловлювань, яка не містить інших простих висловлювань, крім $a_1, \ldots, a_n$. Множину $\mathcal{O}(F)$ усіх тих наборів $(\varepsilon_1, \ldots, \varepsilon_n)$ логічних значень, для яких $F(\varepsilon_1, \ldots, \varepsilon_n) = 1$, називають *областю істинності* формули F. Її доповнення до множини всіх булевих векторів довжини n називають *областю хибності* формули F (тобто це множина всіх таких наборів $(\varepsilon_1, \ldots, \varepsilon_n)$, що $F(\varepsilon_1, \ldots, \varepsilon_n) = 0$).

Формулу логіки висловлювань називають *виконливою*, якщо її область істинності не є порожньою. Виконливу формулу називають *тавтологією* (або *тотожно істинною*), якщо її область істинності містить усі булеві вектори відповідної довжини (або, що те саме, область хибності є порожньою). Формули з непорожньою областю хибності називають *спростовуваними*. Спростовувану формулу називають *суперечністю* (або *тотожно хибною)*, якщо її область істинності — порожня множина. Формули, для яких і область істинності, і область хибності — непорожні, називають *нейтральними*.

Однією із центральних задач у логіці висловлювань (до неї зводиться багато інших) є задача встановлення типу формули (тавтологія, тотожно хибна чи нейтральна). Легко зрозуміти, що досить уміти перевіряти формули лише на тавтологічність. Справді, формула F є тотожно хибною, якщо її заперечення $\neg F$ є тавтологією, і нейтральною, якщо ні F, ні $\neg F$ не є тавтологіями. Найпростішим, хоч і громіздким, методом перевірки формули на тавтологічність є побудова її таблиці істинності.

Ми вкажемо ще два методи перевірки формули на тавтологічність, які інколи бувають зручнішими.

А. Міркування від супротивного. Цей метод полягає в пошуку такого набору значень простих висловлювань, за яких формула стає хибною. Він особливо ефективний, коли аналізована формула включає багато імплікацій.

Для прикладу розглянемо формулу

$$F = ((p \wedge q) \to r) \to (p \to (q \to r)).$$

Ця формула буде хибною лише в тому випадку, коли $(p \wedge q) \to r$ є істинною, а $p \to (q \to r)$ — хибною. Друга формула буде хибною лише тоді, коли $p = 1$, а формула $q \to r$ є хибною (що можливо лише тоді, коли $q = 1$ і $r = 0$). Отже, формула F може бути хибною лише у випадку $p = 1$, $q = 1$, $r = 0$.

Безпосередньо перевіряємо, що на наборі $p = 1$, $q = 1$, $r = 0$ формула F набуває значення 1. Тому F є тавтологією.

Зауважимо, що хоча нам і довелося використати певні додаткові міркування, зате замість 8 можливих наборів значень для p, q і r ми обчислювали значення формули F лише на одному наборі.

В. Арифметизація формул. Інколи зручно вважати, що змінні з формул логіки висловлювань набувають не логічних, а числових значень 0 і 1 (дійсних чи з поля $\mathbb{Z}_2$). Тоді логічним операціям відповідатимуть певні арифметичні операції:

	над $\mathbb{R}$	над $\mathbb{Z}_2$
$\neg p$	$1-p$	$1+p$
$p \vee q$	$p+q-pq$	$p+q+pq$
$p \wedge q$	pq	pq
$p \to q$	$1-p+pq$	$1+p+pq$
$p \leftrightarrow q$	$1-p-q+2pq$	$1+p+q$
$p \oplus q$	$p+q-2pq$	$p+q$

Алгоритм перевірки формули на тавтологічність виглядає таким чином: спочатку замінюємо всі логічні операції відповідними арифметичними, а потім, користуючись арифметичними тотожностями, спрощуємо отриманий вираз. Зауважимо, що наші змінні набувають лише значень 0 і 1, а тому для них виконується ще й тотожність $x^2 = x$. Крім того, над полем $\mathbb{Z}_2$ маємо ще й тотожність $x + x = 0$. Ці тотожності значно полегшують обчислення.

Якщо після всіх спрощень отримаємо 1, формула є тавтологією. У протилежному разі формула є спростовуваною.

Розглянемо два приклади.

1. Для перевірки на тавтологічність формули

$$(q \to r) \to ((p \vee q) \to (p \vee r))$$

застосуємо арифметизацію над полем $\mathbb{Z}_2$:

$$\begin{gathered}
F = 1+(1+q+qr)+(1+q+qr)\big(1+(p+q+pq)+(p+q+pq)(p+r+pr)\big) = \\
= q+qr+(1+q+qr)(1+p+q+pq+ \\
+p^2+pr+p^2r+qp+qr+qpr+p^2q+pqr+p^2qr) = \\
= q+qr+(1+q+qr)(1+q+qr+pq+pqr) = \\
= q+qr+(1+q+qr+pq+pqr+q+q^2+q^2r+ \\
+pq^2+pq^2r+qr+q^2r+q^2r^2+pq^2r+pq^2r^2) = \\
= q+qr+(1+q+qr) = 1.
\end{gathered}$$

Отже, формула є тавтологією.

2. Для перевірки на тавтологічність формули

$$(q \to (p \wedge r)) \wedge \neg((p \vee r) \to q)$$

застосуємо арифметизацію над полем $\mathbb{R}$:

$$\begin{gathered} F = (1 - q + qpr)\big(1 - (1 - (p + r - pr) + (p + r - pr)q)\big) = \\ = (1 - q + qpr)(p + r - pr - pq - rq + pqr) = \\ = p + r - pr - pq - rq + pqr - pq - rq + prq + pq^2 + rq^2 + pq^2r = \\ = p + r - pr - pq - rq + pqr. \end{gathered}$$

Останній вираз не дорівнює 1, тому формула не є тавтологією. З отриманого виразу, видно, зокрема, що $F = 0$ при $p = q = r = 0$ і при $p = q = r = 1$.

Зауваження. *Пізніше розглянемо ще один метод перевірки формул на тавтологічність — за допомогою рівносильних перетворень.*

Оскільки є суто алгоритмічна процедура перевірки формули на тавтологічність за допомогою таблиць істинності, то надалі доведення того, що дана формула логіки висловлювань є тавтологією, зазвичай вилучаємо.

1.4. Алгебра формул

1.4.1. Інтерпретації

Множина $\mathfrak{F}$ всіх формул логіки висловлювань, записаних із використанням логічних зв'язок $\neg$, $\vee$, $\wedge$, $\to$, $\leftrightarrow$, $\oplus$, природно перетворюється на універсальну алгебру сигнатури $(0, 1, \neg, \vee, \wedge, \to, \leftrightarrow, \oplus)$ (напр., результатом застосування бінарної операції $*$ до формул A і B є формула $(A * B)$).

Як алгебра $\mathfrak{F}$ породжується множиною $\mathcal{A} = \{a, a_1, a_2, \ldots, b, b_1, \ldots\}$ символів змінних. Застосовуючи до елементів цієї множини операції даної сигнатури, ми зможемо одержати довільну формулу алгебри висловлювань. Навіть більше, з означення формули випливає, що таким шляхом кожну формулу можна одержати лише одним способом[14].

Універсальна алгебра такої ж сигнатури визначається і на множині $B = \{0, 1\}$ логічних значень (результат застосування операції до відповідних

[14] Якщо універсальна алгебра має систему твірних із такою властивістю, то її називають *вільною*.

логічних значень визначається таблицею істинності цієї операції). Алгебру $\langle\{0,1\};0,1,\neg,\vee,\wedge,\to,\leftrightarrow,\oplus\rangle$ позначатимемо $\mathfrak{B}$.

Оскільки алгебри $\mathfrak{F}$ і $\mathfrak{B}$ мають однаковий набір операцій, то можна розглядати гомоморфізми $\varphi:\mathfrak{F}\to\mathfrak{B}$. Такі гомоморфізми називатимемо *інтерпретаціями* алгебри формул логіки висловлювань. Зокрема, образ $\varphi(F)$ формули F при деякому гомоморфізмі φ називатимемо *інтерпретацією* формули F.

Зрозуміло, що гомоморфізм алгебри повністю задається своїми значеннями на множині твірних. У нашому випадку гомоморфізм $\varphi:\mathfrak{F}\to\mathfrak{B}$ повністю задається своїми значеннями на множині $\mathcal{A}$ символів змінних. А оскільки кожну формулу отримують із цих символів тільки одним способом, то образи змінних можна задавати довільно. Іншими словами, кожне відображення $\widetilde{\varphi}:\mathcal{A}\to\mathfrak{B}$ однозначно продовжується до гомоморфізму $\varphi:\mathfrak{F}\to\mathfrak{B}$.

Саме відображення $\widetilde{\varphi}:\mathcal{A}\to\mathfrak{B}$ часто називають *оцінкою*. Оцінку $\widetilde{\varphi}$ можна розглядати як присвоєння символам простих висловлювань певних значень істинності. За цієї оцінки кожна формула F отримує значення істинності $\varphi(F)$.

Інтерпретацію $\varphi:\mathfrak{F}\to\mathfrak{B}$ (і її обмеження $\widetilde{\varphi}$ на множину $\mathcal{A}$ символів змінних) називатимемо *моделлю* формули F, якщо $\varphi(F)=1$. Зокрема, формула F буде *виконливою*, якщо вона має модель, і буде *тавтологією*, якщо кожна інтерпретація буде її моделлю F.

Будемо говорити, що формули A і B *рівносильні* (і позначати $A\equiv B$), якщо за кожної інтерпретації образи цих формул збігаються (тобто $\varphi(A)=\varphi(B)$ для кожного гомоморфізму $\varphi:\mathfrak{F}\to\mathfrak{B}$). Іншими словами, якщо на кожному наборі логічних значень змінних, що зустрічаються хоча б в одній із формул A і B, ці формули набувають однакових логічних значень. У термінах моделей це означає, що вони мають ті самі моделі.

Два висловлювання називають *рівносильними*, якщо їх можна одержати з рівносильних формул A і B за допомогою заміни всіх змінних, що входять до цих формул, конкретними висловлюваннями.

Вправа 1.1. *Доведіть, що формули A і B будуть рівносильними тоді й лише тоді, коли формула $A\leftrightarrow B$ буде тавтологією.*

Вправа 1.2. *Доведіть такі рівносильності:*

a) для штриха Шефера: $a\mid b\equiv\neg(a\wedge b)$ *;*

b) для стрілки Пірса (стрілки Лукасевича) : $a\downarrow b\equiv\neg(a\vee b)$.

1.4.2. Алгебра Лінденбаума*

Нагадаємо, що конгруенцією на алгебрі $\mathcal{A}$ називається відношення еквівалентності $\sim$ на цій алгебрі, узгоджене з усіма операціями в $\mathcal{A}$. Тобто для кожної унарної операції $\widehat{\ }$ із $a \sim a_1$ має випливати $\widehat{a} \sim \widehat{a_1}$, а для кожної бінарної операції $*$ із $a \sim a_1$ і $b \sim b_1$ має випливати $a * b \sim a_1 * b_1$ і т. д.

Для конгруенції $\sim$ через $\overline{a}$ позначимо той клас еквівалентності, що містить a. На множині $\mathcal{A}/\!\sim$ усіх класів еквівалентності конгруенції $\sim$ можна визначити всі операції, які є в алгебрі $\mathcal{A}$:

$$\widehat{\overline{a}} := \overline{\widehat{a}}, \quad \overline{a} * \overline{b} := \overline{a * b}, \quad \ldots$$

Коректність визначення цих операції, тобто незалежність результату $\widehat{\overline{a}}$, $\overline{a} * \overline{b}$ тощо від вибору конкретних представників a, b, ... відповідних класів еквівалентності, випливає з узгодженості відношення $\sim$ з операціями в алгебрі $\mathcal{A}$. Отриману таким чином алгебру $\mathcal{A}/\!\sim$ називають *факторалгеброю* алгебри $\mathcal{A}$ за конгруенцією $\sim$.

Теорема 1.1. *Відношення рівносильності $\equiv$ є конгруенцією на алгебрі $\mathfrak{F}$ всіх формул логіки висловлювань.*

Доведення. З означення відношення $\equiv$ одразу випливає, що воно є відношенням еквівалентності. Тому треба перевірити лише узгодженість відношення $\equiv$ з операціями.

Очевидно, що з $A \equiv B$ випливає $\neg A \equiv \neg B$. Нехай тепер $A_1 \equiv A_2$, $B_1 \equiv B_2$, $\varphi : \mathfrak{F} \to \mathfrak{B}$ — довільна інтерпретація, а $*$ — якась із бінарних операцій. Враховуючи, що $\varphi(A_1) = \varphi(A_2)$ і $\varphi(B_1) = \varphi(B_2)$, то

$$\varphi(A_1 * B_1) = \varphi(A_1) * \varphi(B_1) = \varphi(A_2) * \varphi(B_2) = \varphi(A_2 * B_2).$$

Оскільки інтерпретація φ — довільна, то $A_1 * A_1 \equiv A_2 * B_2$. □

Тому можна розглянути факторалгебру $\mathfrak{L} = \mathfrak{F}/\!\equiv$ алгебри $\mathfrak{F}$ за відношенням рівносильності. Її елементами є *класи рівносильних формул*. Дії у факторалгебрі $\mathfrak{L}$ визначають стандартно: якщо $\overline{A}$ — клас рівносильності, що містить формулу A, то

$$\neg\overline{A} := \overline{\neg A} \quad \text{і} \quad \overline{A} * \overline{B} := \overline{A * B} \quad \text{для довільної бінарної дії } *.$$

Факторалгебру $\mathfrak{L}$ називають *алгеброю Лінденбаума* (інші назви: *алгебра висловлювань*, *алгебра класів рівносильних формул*).

Оскільки

$$A \to B \equiv \neg A \vee B\,;$$
$$A \leftrightarrow B \equiv (A \to B) \wedge (B \to A)\,;$$
$$A \oplus B \equiv \neg(A \leftrightarrow B)\,,$$

то в алгебрі Лінденбаума операції $\to$, $\leftrightarrow$, $\oplus$ можна розглядати як похідні від операцій $\vee$, $\wedge$ та $\neg$. Аналогічне зауваження стосується й алгебри $\mathfrak{B}$. Тому далі ми розглядатимемо $\mathfrak{L}$ і $\mathfrak{B}$ як алгебри сигнатури $(0, 1, \neg, \vee, \wedge)$.

Булевою алгеброю назвемо довільну універсальну алгебру сигнатури $(0, 1, \neg, \vee, \wedge)$, в якій виконуються всі ті закони (= тотожності), що виконуються в алгебрі $\mathfrak{B} = \langle\{0, 1\}; 0, 1, \neg, \vee, \wedge\rangle$. Таке задання класу універсальних алгебр є трохи незвичним, зазвичай це роблять за допомогою явно вказаного списку аксіом[15]. Далі ми доведемо, що це можна зробити і для булевих алгебр; і навіть укажемо одну з можливих аксіоматик. Але при цьому ми вже розумітимемо, звідки взявся саме такий набір аксіом і чому алгебри, які задовольняють ці аксіоми, є важливими.

Теорема 1.2. *Алгебра Лінденбаума $\mathfrak{L}$ є булевою.*

Доведення. Нехай

$$F_1(x_1, \ldots, x_n) = F_2(x_1, \ldots, x_n) \tag{1.1}$$

— закон алгебри $\mathfrak{B}$, $\overline{A_1}, \ldots, \overline{A_n}$ — довільні елементи алгебри $\mathfrak{L}$, $A_1, \ldots, A_n$ — представники відповідних класів рівносильних формул. Розглянемо довільний гомоморфізм $\varphi : \mathfrak{F} \to \mathfrak{B}$. З означення гомоморфізму та (1.1) випливає, що

$$\varphi(F_1(A_1, \ldots, A_n)) = F_1(\varphi(A_1), \ldots, \varphi(A_n)) =$$
$$= F_2(\varphi(A_1), \ldots, \varphi(A_n)) = \varphi(F_2(A_1, \ldots, A_n)).$$

Отже,

$$F_1(A_1, \ldots, A_n) \equiv F_2(A_1, \ldots, A_n)\,,$$

тобто

$$\overline{F_1(A_1, \ldots, A_n)} = \overline{F_2(A_1, \ldots, A_n)}\,.$$

Але, за означенням дій у факторалгебрі,

$$\overline{F_1(A_1, \ldots, A_n)} = F_1(\overline{A_1}, \ldots, \overline{A_n}), \quad \overline{F_2(A_1, \ldots, A_n)} = F_2(\overline{A_1}, \ldots, \overline{A_n})\,.$$

[15] Наприклад, задають групи, кільця, векторні простори і багато інших класів алгебр.

Тому

$$F_1(\overline{A_1}, \ldots, \overline{A_n}) = F_2(\overline{A_1}, \ldots, \overline{A_n})\,.$$

Оскільки $\overline{A_1}, \ldots, \overline{A_n}$ — довільні, то закон (1.1) виконується і в алгебрі $\mathfrak{L}$. $\square$

Нехай M — деяка множина. Множину $\mathfrak{B}(M)$ усіх її підмножин можна розглядати як алгебру сигнатури $(\varnothing, M, {}^{-}, \cup, \cap)$. Очевидно, що алгебра $\mathfrak{B}(M)$ однотипна алгебрі $\mathfrak{B} = \langle \{0,1\}; 0, 1, \neg, \vee, \wedge \rangle$.

Теорема 1.3. *Алгебра $\mathfrak{B}(M)$ усіх підмножин даної множини M є булевою.*

Доведення. Для кожного $a \in M$ розглянемо відображення $\varphi_a \colon \mathfrak{B}(M) \to \mathfrak{B}$, визначене правилом:

$$\varphi_a(N) = 1 \;\text{ тоді й лише тоді, коли }\; a \in N\,.$$

Легко перевірити, що відображення φ_a є гомоморфізмом алгебр. Зрозуміло також, що для довільних підмножин $N_1, N_2 \subseteq M$ рівність $N_1 = N_2$ виконується тоді й лише тоді, коли $\varphi_a(N_1) = \varphi_a(N_2)$ для всіх a.

Нехай $F_1(x_1, \ldots, x_n) = F_2(x_1, \ldots, x_n)$ — закон алгебри $\mathfrak{B}$, а $N_1, \ldots, N_n$ — довільні підмножини з M. Тоді для довільного $a \in M$

$$\begin{aligned}\varphi_a(F_1(N_1, \ldots, N_n)) &= F_1(\varphi_a(N_1), \ldots, \varphi_a(N_n)) = \\ &= F_2(\varphi_a(N_1), \ldots, \varphi_a(N_n)) = \varphi_a(F_2(N_1, \ldots, N_n)).\end{aligned}$$

Отже, $F_1(N_1, \ldots, N_n) = F_2(N_1, \ldots, N_n)$. $\square$

Підалгебри алгебр $\mathfrak{B}(M)$ називають *алгебрами множин*.

Теорема 1.4. *Алгебра Лінденбаума $\mathfrak{L}$ ізоморфна деякій алгебрі множин.*

Доведення. Нехай M — множина всіх інтерпретацій алгебри формул $\mathfrak{F}$. Кожному класу $\overline{A}$ рівносильних формул поставимо у відповідність множину N_A тих інтерпретацій, які на формулах із цього класу набувають значення 1: $N_A = \{\varphi \in M \mid \varphi(A) = 1\}$. Розглянемо відображення

$$\psi : \mathfrak{L} \to \mathfrak{B}(M)\,, \quad \overline{A} \mapsto N_A\,.$$

Покажемо, що ψ є ін'єктивним гомоморфізмом.

Ін'єктивність ψ. Якщо $\overline{A} \neq \overline{B}$, то $A \not\equiv B$ й існує інтерпретація φ, за якої $\varphi(A) \neq \varphi(B)$. Скажімо, $\varphi(A) = 1$, $\varphi(B) = 0$. Але тоді $\varphi \in N_A$ і $\varphi \notin N_B$. Отже, $\psi(\overline{A}) \neq \psi(\overline{B})$.

Гомоморфність ψ. Очевидно, що $\psi(0) = \varnothing$, $\psi(1) = M$. Далі маємо

$$\varphi \in \psi(\overline{A} \vee \overline{B}) \;\Leftrightarrow\; \varphi(A \vee B) = 1 \;\Leftrightarrow\; \varphi(A) = 1 \text{ або } \varphi(B) = 1 \;\Leftrightarrow$$
$$\Leftrightarrow\; \varphi \in N_A \text{ або } \varphi \in N_B \;\Leftrightarrow\; \varphi \in N_A \cup N_B \;\Leftrightarrow\; \varphi \in \psi(\overline{A}) \cup \psi(\overline{B})\,.$$

Отже, $\psi(\overline{A} \vee \overline{B}) = \psi(\overline{A}) \cup \psi(\overline{B})$. Узгодженість ψ з іншими діями перевіряють аналогічно.

Враховуючи, що ψ є гомоморфізмом, то його образ $\psi(\mathfrak{L})$ буде підалгеброю в $\mathfrak{B}(M)$, тобто алгеброю множин. А з ін'єктивності ψ випливає, що відображення $\psi : \mathfrak{L} \to \psi(\mathfrak{L})$ є ізоморфізмом. □

Зауваження. *Теорему* 1.4 *легко узагальнити: кожна булева алгебра ізоморфна деякій алгебрі множин.*

1.4.3. Властивості рівносильних формул

В алгебрі висловлювань використання рівносильностей відіграє приблизно таку ж роль, як у шкільній алгебрі використання тотожностей. Тому зупинимося на властивостях рівносильних формул детальніше.

Теорема 1.5 (про рівносильну заміну)**.** *Нехай A_B — формула A з виділеним входженням підформули B, а $A_{B'}$ — формула, яку одержують з A заміною виділеного входження B в A на формулу B'. Тоді якщо $B \equiv B'$, то $A_B \equiv A_{B'}$.*

Доведення. Вилучимо в таблицях істинності (див. пункт 1.3.2) для формул A_B та $A_{B'}$ ті стовпці, які відповідають власним підформулам виділеної формули B (відповідно формули B'). Далі поставимо у відповідність кожному стовпцю таблиці для A_B, який відповідає підформулі C, що містить B, той стовпець таблиці для $A_{B'}$, який одержується із C заміною B на B'. З рівносильності формул B і B' випливає, що отримані таблиці будуть однаковими. Зокрема, однаковими будуть і ті стовпці цих таблиць, що відповідають формулам A_B і $A_{B'}$. Тому $A_B \equiv A_{B'}$. □

Нехай A і B — деякі формули. *Оператор підстановки* $\prod_A^B$ — це правило перетворення формул логіки висловлювань, за яким у довільній формулі F усі підформули A одночасно замінюють на підформули B. Оператор підстановки найчастіше вживають тоді, коли $A = x$ — символ змінної. Крім того, у випадку формули $F = F(x_1, \ldots, x_n)$ замість

$$\prod_{x_1}^{B_1} \cdots \prod_{x_n}^{B_n} F$$

зазвичай пишуть просто $F(B_1, \ldots, B_n)$.

Із теореми 1.5 про рівносильну заміну одразу випливає наслідок.

Наслідок 1.1. *Якщо $A \equiv B$, то для довільної формули F буде $\prod_A^B F \equiv F$.*

Наступна теорема є очевидною.

Теорема 1.6 (про підстановку)**.** *Якщо $F = F(x_1, \ldots, x_n)$ — тавтологія, то для довільних формул $B_1, \ldots, B_n$ формула $F = F(B_1, \ldots, B_n)$ також буде тавтологією.*

Якщо $B \equiv B'$, то перехід від формули A_B до $A_{B'}$ часто називають *рівносильним перетворенням* формули A. Рівносильні перетворення часто використовують для зведення формули A до зручнішого або простішого вигляду або для доведення рівносильності двох формул.

Наведемо список основних рівносильностей, які використовують у рівносильних перетвореннях формул логіки висловлювань:

1. *Закон подвійного заперечення*: $\neg\neg A \equiv A$.
2. *Комутативні закони*: $A \vee B \equiv B \vee A$, $A \wedge B \equiv B \wedge A$, $A \leftrightarrow B \equiv B \leftrightarrow A$, $A \oplus B \equiv B \oplus A$.
3. *Асоціативні закони*: $(A \vee B) \vee C \equiv A \vee (B \vee C)$, $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$, $(A \leftrightarrow B) \leftrightarrow C \equiv A \leftrightarrow (B \leftrightarrow C)$, $(A \oplus B) \oplus C \equiv A \oplus (B \oplus C)$.
4. *Дистрибутивні закони*: $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$, $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$.
5. *Закони ідемпотентності*: $A \vee A \equiv A$, $A \wedge A \equiv A$.
6. *Закони де Моргана*: $\neg(A \vee B) \equiv \neg A \wedge \neg B$, $\neg(A \wedge B) \equiv \neg A \vee \neg B$.
7. *Закони поглинання* (адсорбції): $A \vee (A \wedge B) \equiv A$, $A \wedge (A \vee B) \equiv A$.
8. *Закони поглинання константами*: $A \vee 1 \equiv 1$, $A \wedge 1 \equiv A$, $A \vee 0 \equiv A$, $A \wedge 0 \equiv 0$.
9. *Закон виключення третьої можливості* (*tertium non datur*): $A \vee \neg A \equiv 1$.
10. *Закон суперечності*: $A \wedge \neg A \equiv 0$.

До найважливіших рівносильностей логіки висловлювань належать і так звані *закони виключення логічних зв'язок*:

11. $A \oplus B \equiv \neg(A \leftrightarrow B)$;
12. $A \leftrightarrow B \equiv (A \to B) \wedge (B \to A)$;
13. $A \to B \equiv \neg A \vee B$;
14. $A \vee B \equiv \neg(\neg A \wedge \neg B)$;
15. $A \wedge B \equiv \neg(\neg A \vee \neg B)$.

1.5. Логічний наслідок

Нехай $\Delta = \{A_1, \ldots, A_n\}$ — набір формул, B — формула, $x_1, \ldots, x_k$ — список усіх змінних, які зустрічаються принаймні в одній із цих формул. Кажуть, що формула B є *логічним наслідком* із формул $A_1, \ldots, A_n$, якщо для кожного набору логічних значень змінних $x_1, \ldots, x_k$, за яких усі формули $A_1, \ldots, A_n$ є істинними, формула B також буде істинною (тобто, якщо перетин областей істинності формул $A_1, \ldots, A_n$ міститься в області істинності формули B). Інколи зручніше використовувати рівносильне означення: якщо на якомусь наборі логічних значень змінних $x_1, \ldots, x_k$ формула B хибна, то на цьому наборі принаймні одна з формул $A_1, \ldots, A_n$ також буде хибною.

Той факт, що B є логічним наслідком набору формул $\Delta = \{A_1, \ldots, A_n\}$, позначають $A_1, \ldots, A_n \models B$ або $\Delta \models B$. Тут B є *висновком*, а формули $A_1, \ldots, A_n$ — *засновками* логічного наслідку.

Поняття логічного наслідку легко узагальнюють на довільні множини формул: формула F є *логічним наслідком* множини формул Δ, якщо кожна модель для Δ буде моделлю і для F.

Зокрема, множина Δ може бути порожньою. Тоді з означення логічного наслідку випливає, що B буде істинною на кожному наборі логічних значень змінних $x_1, \ldots, x_k$. Отже, запис $\models B$ означає, що формула B є тавтологією.

Вправа 1.3. *Переформулюйте означення логічного наслідку в термінах інтерпретацій.*

Твердження 1.1. *$A \models B$ тоді й лише тоді, коли формула $A \to B$ є тавтологією.*

Доведення. Співвідношення $A \models B$ виконується тоді й лише тоді, коли з істинності A випливає істинність B, тобто коли формула $A \to B$ є тотожно істинною. □

Наслідок 1.2.

$$A_1, \ldots, A_n \models B \Leftrightarrow \models A_1 \to (A_2 \to (\cdots \to (A_n \to B) \cdots)).$$

Формули $A_1 \to (A_2 \to (\cdots \to (A_n \to B) \cdots))$ і $(A_1 \wedge \cdots \wedge A_n) \to B$ рівносильні. Тому

$$A_1, \ldots, A_n \models B \Leftrightarrow \models (A_1 \wedge \cdots \wedge A_n) \to B. \tag{1.2}$$

Нарешті, із співвідношення (1.2) і твердження 1.1 випливає, що

$$A_1, \ldots, A_n \models B \;\Leftrightarrow\; A_1 \wedge \cdots \wedge A_n \models B. \tag{1.3}$$

Отже, поняття логічного наслідку й тавтології виявляються тісно пов'язаними: питання про правильність певного висновку зводиться до питання про тавтологічність деякої формули. З іншого боку, поняття логічного наслідку можна вважати дуже широким узагальненням поняття імплікації: якщо останнє пов'язує окремі висловлювання, то логічний наслідок — цілі множини висловлювань.

Поняття логічного наслідку дозволяє дати строге означення *логічності міркування*: міркування є логічним, якщо під час його формалізації у вигляді послідовності формул засновки і висновок виявляються пов'язаними відношенням логічного наслідку. Тобто це міркування, що мають цілком певну форму.

Приклад. *З'ясуємо, чи є логічними такі міркування:*

Кава смачна, а ліки не є смачними. Якщо людина хвора, то вона має приймати ліки або дотримуватися режиму. Дотримуватися режиму і не приймати ліки неможливо. Отже, якщо людина п'є каву, то вона не хвора.

Насамперед виділимо прості висловлювання:

a — *"кава смачна";*
b — *"ліки смачні";*
c — *"людина хвора";*
d — *"людина дотримується режиму";*
e — *"людина приймає ліки";*
f — *"людина п'є каву".*

Після цього формалізуємо висловлювання, з яких складається дане міркування:

$a \wedge \neg b$ — *"кава смачна, а ліки не є смачними";*

$c \to (d \vee e)$ — *"якщо людина хвора, то вона має приймати ліки або дотримуватися режиму";*

$\neg(d \wedge \neg e)$ — *"дотримуватися режиму і не приймати ліки неможливо";*

$f \to \neg c$ — *"якщо людина п'є каву, то вона не хвора".*

Тепер питання про логічність міркувань можна переформулювати таким чином: чи правильно стоїть знак $\models$ у співвідношенні

$$a \wedge \neg b,\ c \to (d \vee e),\ \neg(d \wedge \neg e) \models f \to \neg c.$$

Висновок $f \to \neg c$ буде хибним, коли $f = c = 1$. Спробуємо підібрати такі значення істинності решти a, b, d, e простих висловлювань, щоб усі формули

ліворуч від знака $\models$ стали істинними. Очевидно, що це буде тоді, коли $a = 1$, $b = 0$, а формули $d \vee e$ і $d \wedge \neg e$ набувають відповідно значень 1 і 0. Останнє буде, зокрема, коли $e = 1$.

Отже, ми змогли підібрати такі значення істинності простих висловлювань, за яких усі формули ліворуч від знака $\models$ істинні, а висновок хибний. Тому знак $\models$ стоїть неправильно і міркування не є логічними.

У загальному випадку формалізація ланцюжка міркувань є процедурою творчою, зокрема, тому, що в живій мові деякі висловлювання можуть бути присутніми неявно і вгадуватися лише з контексту. Особливо часто таке трапляється в математичних текстах. Пропущені фрагменти міркувань можуть бути позначені (однак не завжди) словами "очевидно", "легко бачити" тощо.

Крім логічних міркувань часто, особливо у природничих науках, вживаються ще так звані "фактичні міркування" — в яких правильність висновку випливає не тільки з логічної структури засновків, але і з певних причинно-наслідкових зв'язків. Наприклад: "якщо по провіднику проходить струм, то він нагрівається". Аналіз правильності таких міркувань виходить поза межі математичної логіки.

Одним з основних завдань логіки є класифікація усіх можливих форм логічних міркувань. Першим кроком у цьому напрямку є виділення найпоширеніших типів логічних наслідків.

Теорема 1.7 (основні типи логічних наслідків)**.**
(1) *modus ponens*[16]: $A, A \to B \models B$;
(2) *modus tollens*[17]: $A \to B, \neg B \models \neg A$;
(3) *Правило силогізму*: $A \to B, B \to C \models A \to C$;
(4) *Введення диз'юнкції та кон'юнкції*: $A \models A \vee B$, $A, B \models A \wedge B$;
(5) *Вилучення диз'юнкції та кон'юнкції*: $A \wedge B \models A$, $A \vee B, \neg B \models A$;
(6) *Правило контрапозиції*: $A \to B \models \neg B \to \neg A$;
(7) *Закони Клавіуса*: $\neg A \to A \models A$, $A \to \neg A \models \neg A$;
(8) *Закон Дунса Скотта*: $A \wedge \neg A \models B$;
(9) *ex falsoquod libet*[18]: $\neg A \models A \to B$;
(10) *Метод зведення до абсурду*: $\neg A \to B, \neg A \to \neg B \models A$;
(11) *Метод вичерпування можливостей для умови*:
$A_1 \vee \cdots \vee A_n, A_1 \to B, \ldots, A_n \to B \models B$;
(12) *Метод вичерпування можливостей для наслідку*:
$B \to (A_1 \vee \cdots \vee A_n), \neg(B \to A_2), \ldots, \neg(B \to A_n), \models B \to A_1$.

[16] Modus ponens (лат.) — правило ствердження.
[17] Modus tollens (лат.) — правило заперечення.
[18] Ex falsoquod libet (лат.) — з брехні що завгодно (принцип вибуху).

Доведення. Перші десять співвідношень легко перевіряються за допомогою таблиць істинності.

(11) Припустимо, що всі формули ліворуч від символу $\models$ є істинними. Тоді має бути істинною і якась із формул A_i. Але з істинності формул A_i та $A_i \to B$ випливає істинність формули B.

(12) Знову припустимо, що всі формули ліворуч від символу $\models$ є істинними. Якщо $\neg(B \to A_i)$ істинна, то $B \to A_i$ хибна, що можливо лише, коли B істинна, а A_i хибна. З істинності формул B і $B \to (A_1 \vee \cdots \vee A_n)$ випливає істинність $A_1 \vee \cdots \vee A_n$. Але A_2, …, A_n — хибні. Тому A_1 має бути істинною. □

Перевірити безпосередньо правильність співвідношення $\Delta \models B$ інколи буває складно. Полегшити таку перевірку може очевидне і корисне твердження.

Твердження 1.2. *a)* $A_1, \ldots, A_n \models A_i$ *для довільного* $1 \le i \le n$;

b) якщо для довільного $1 \le j \le m$ $A_1, \ldots, A_n \models C_j$ *і* $C_1, \ldots, C_m \models B$, *то* $A_1, \ldots, A_n \models B$.

Користуючись цим твердженням, доведення правильності співвідношення $A_1, \ldots, A_n \models B$ можна подати у вигляді ланцюжка формул D_1, D_2, …, D_k, де $D_k = B$, а присутність у ланцюжку кожної з формул D_j обґрунтовується одним із правил:

(1) формула D_j збігається з одним із засновків A_i;

(2) знайдуться такі формули D_{j_1}, …, D_{j_m}, які передують D_j (тобто $j_1, \ldots, j_m < j$), що $D_{j_1}, \ldots, D_{j_m} \models D_j$.

Зауваження. 1. *Логічні наслідки, які використовуються у правилі* (2), *можуть бути спеціального простого вигляду (напр., із теореми* 1.7*).*

2. *У ланцюжок* D_1, …, D_k *можна включати будь-яку тавтологію (для тавтології* D *і довільних формул* D_1, …, D_r *завжди матимемо* $D_1, \ldots, D_r \models D$*).*

3. *Зі схожими правилами ми ще зіткнемося пізніше, коли розглядатимемо формальні теорії.*

Множину Δ формул (можливо, нескінченну) називають *суперечливою* або *несумісною*, якщо $\Delta \models F$ для кожної формули F. Оскільки F може бути тотожно хибною, то це означає, що для кожного набору логічних значень змінних, що входять у формули з множини Δ, принаймні одна із цих формул буде хибною. У протилежному разі множину Δ називають *несуперечливою* (або *сумісною* чи *виконливою*).

У термінах моделей суперечливість множини формул означає, що ця множина не має моделі.

Твердження 1.3. *Множина формул Δ буде суперечливою тоді й лише тоді, коли тотожно хибна формула буде логічним наслідком із Δ.*

Доведення. Необхідність умови випливає безпосередньо з означення суперечливої множини. З другого боку, якщо для якогось набору логічних значень змінних усі формули з Δ будуть істинними, то тотожно хибна формула не буде логічним наслідком з Δ. □

Твердження 1.4 (доведення від супротивного (reductio ad absurdum))**.** $\Delta \models B$ *тоді й лише тоді, коли множина формул $\Delta \cup \{\neg B\}$ є суперечливою.*

Доведення. Нехай $\Delta \models B$. Тоді для кожного набору логічних значень змінних, для якого всі формули з Δ істинні, істинною буде і формула B, а формула $\neg B$ — хибною. Отже, для кожного набору логічних значень змінних буде хибною або принаймні одна з формул із Δ, або $\neg B$. Тому множина $\Delta \cup \{\neg B\}$ є суперечливою.

Навпаки, якщо множина $\Delta \cup \{\neg B\}$ є суперечливою, то для кожного набору логічних значень змінних, коли всі формули з Δ істинні, формула $\neg B$ має бути хибною. Але тоді формула B буде істинною. Тому $\Delta \models B$. □

1.6. Двоїстість

Нехай формула F утворена тільки за допомогою зв'язок $\neg$, $\vee$, $\wedge$, $\leftrightarrow$, $\oplus$, 0, 1. Формулу F^* називають *двоїстою* до F, якщо вона отримується з F заміною кожного входження зв'язки $\vee$ на $\wedge$ і навпаки, а також кожного входження $\oplus$ на $\leftrightarrow$ і навпаки та 0 на 1 і навпаки. Легко бачити, що $(F^*)^* = F$.

Лема 1.1. *Нехай формула F утворена тільки за допомогою зв'язок $\neg$, $\vee$, $\wedge$, $\leftrightarrow$, $\oplus$, 0, 1. Позначимо через F' формулу, що одержується з F заміною кожного символу змінної його запереченням* (тобто, якщо $F = F(x_1, \ldots, x_n)$, то $F' = F(\neg x_1, \ldots, \neg x_n)$) *. Тоді* $F' \equiv \neg F^*$.

Доведення. Застосуємо індукцію за довжиною формули F. Для формул довжини 0 твердження очевидне.

Нехай тепер для формул, коротших ніж F, лему вже доведено. Далі можливі два випадки:

1. $F = \neg G$. У цьому випадку $F^* = \neg G^*$ і за припущенням індукції $G' \equiv \neg G^*$. Тому

$$F' = \neg G' \equiv \neg\neg G^* = \neg F^*.$$

2. $F = G_1 \circ G_2$, де $\circ$ — якась із бінарних зв'язок $\vee$, $\wedge$, $\leftrightarrow$, $\oplus$. У цьому випадку $F^* = G_1^* \overset{*}{\circ} G_2^*$, де $\overset{*}{\circ}$ — зв'язка, двоїста до $\circ$. Враховуючи закони де

Моргана і рівносильності

$$A \leftrightarrow B \equiv \neg A \leftrightarrow \neg B, \quad A \oplus B \equiv \neg A \oplus \neg B,$$

отримуємо

$$F' = G_1' \circ G_2' \equiv \neg G_1^* \circ \neg G_2^* \equiv \neg(G_1^* \overset{*}{\circ} G_2^*) = \neg F^*. \qquad \square$$

Теорема 1.8 (принцип двоїстості)**.** *Якщо $F \equiv G$, то $F^* \equiv G^*$.*

Доведення. Використовуючи теорему про підстановку (теорема 1.6) і лему 1.1, отримуємо такий ланцюжок імплікацій:

$$F \equiv G \;\Rightarrow\; F' \equiv G' \;\Rightarrow\; \neg F^* \equiv \neg G^* \;\Rightarrow\; F^* \equiv G^*. \qquad \square$$

2. Булеві функції і нормальні форми

2.1. Булеві функції

Нехай $F(x_1, \ldots, x_n)$ — формула логіки висловлювань, яка не містить інших змінних, окрім $x_1, \ldots, x_n$. При інтерпретаціях цим змінним надають певні логічні значення. На кожному наборі $(\varepsilon_1, \ldots, \varepsilon_n)$ таких значень формула F набуває логічного значення $F(\varepsilon_1, \ldots, \varepsilon_n)$. Тим самим формула $F(x_1, \ldots, x_n)$ визначає певну функцію вигляду

$$F : \{0,1\}^n \to \{0,1\}, \quad (\varepsilon_1, \ldots, \varepsilon_n) \mapsto F(\varepsilon_1, \ldots, \varepsilon_n). \tag{2.1}$$

Впорядковані набори $(\varepsilon_1, \ldots, \varepsilon_n)$ логічних значень називають *булевими векторами довжини* (або *розмірності*) n, а довільні функції вигляду (2.1) — *булевими* функціями арності n[19].

Отже, кожна формула логіки висловлювань визначає деяку булеву функцію.

Теорема 2.1. *Існує 2^n булевих векторів довжини n і 2^{2^n} різних булевих функцій від n аргументів.*

Доведення. Перша частина твердження випливає з того, що кожна з n координат булевого вектора може набувати будь-якого з двох значень 0 і 1, причому різні координати набувають цих значень незалежно. Другу частину доводять аналогічно: на кожному з 2^n булевих векторів функція може набувати будь-якого з двох значень 0 і 1, причому ці значення на різних векторах можна вибирати незалежно. $\square$

[19] Названі так на честь англійського логіка Дж. Буля (1815–1864).

Подібно тому, як це було для формул, можна говорити про *область істинності*

$$O(f) = \{(\varepsilon_1, \ldots, \varepsilon_n) \mid f(\varepsilon_1, \ldots, \varepsilon_n) = 1\}$$

та *область хибності*

$$\overline{O(f)} = \{(\varepsilon_1, \ldots, \varepsilon_n) \mid f(\varepsilon_1, \ldots, \varepsilon_n) = 0\}$$

булевої функції f від n аргументів.

Найпростішим способом задання булевих функцій є таблиці. У такому заданні зручно виписувати булеві вектори у певному фіксованому порядку. Зазвичай користуються лексикографічним порядком (який збігається з порядком зростання булевих векторів, якщо їх розглядати як записи натуральних чисел у двійковій системі числення).

Незабаром ми побачимо, що кожну булеву функцію можна задати відповідною формулою логіки висловлювань. Однак різних булевих функцій від n аргументів скінченна кількість, а різних формул від цих же аргументів — нескінченно багато. Тому та сама булева функція може задаватися багатьма формулами (взагалі кажучи, нескінченною кількістю). Зрозуміло, що дві формули від тих самих змінних визначатимуть ту саму булеву функцію тоді й лише тоді, коли вони рівносильні[20].

Зауваження. *Кожну булеву функцію від n аргументів можна розглядати як n-арну операцію на множині $B = \{0, 1\}$ логічних значень. Алгебру, утворену множиною B разом з усіма можливими операціями на ній, називають алгеброю логіки.*

2.2. Повні системи зв'язок

Якщо на множині B визначена деяка операція (напр., бінарна операція $*$), то для кожної множини A ця операція природно — поточково — переноситься на функції з A у B:

$$(f * g)(x) := f(x) * g(x) \quad \text{для всіх} \quad x \in A\,.$$

[20] Уточнення "від тих самих змінних" важливе. Рівносильні формули формально можуть залежати від різних наборів змінних. Напр., формули $x \vee \neg x$ і $y \vee \neg y$ рівносильні, однак формально перша визначає тотожно істинну функцію від змінної x, а друга — від змінної y. Щоб можна було говорити про ту саму булеву функцію, її треба розглядати вже як функцію двох змінних — x та y. Так з'являються неявні (фіктивні) змінні.

Це дозволяє всі операції, визначені на множині $B = \{0, 1\}$ значень істинності, застосовувати й до булевих функцій. Але на множинах функцій (зокрема, й булевих) є і свої операції, пов'язані саме з функціональною природою елементів цих множин. Найважливішою з таких операцій є *суперпозиція*.

Нехай $f_1(x_1, \ldots, x_n)$, $\ldots$, $f_m(x_1, \ldots, x_n)$ і $g(y_1, \ldots, y_m)$ — булеві функції (тут x_1, $\ldots$, x_n — список усіх змінних, від яких залежить хоча б одна з функцій f_1, $\ldots$, f_m. Тому якісь із цих функцій можуть залежати від частини змінних фіктивно). *Суперпозицією* вказаних функцій (або *підстановкою* функцій $f_1, \ldots, f_m$ у функцію g) називають функцію

$$h(x_1, \ldots, x_n) = g\big(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)\big).$$

Замиканням $[S]$ системи S булевих функцій називають сукупність усіх булевих функцій, які можна одержати з S за допомогою суперпозиції.

Систему S булевих функцій називають *повною*, якщо її замикання $[S]$ збігається з множиною всіх булевих функцій, тобто, якщо будь-яку булеву функцію можна подати у вигляді суперпозиції функцій із S.

Аналогічно систему логічних зв'язок називають *повною*, якщо кожну булеву функцію можна задати формулою, записаною за допомогою зв'язок тільки із цієї системи.

Зрозуміло, що кожна система булевих функцій (логічних зв'язок), яка містить повну систему, сама є повною, а кожна система, яка міститься в неповній — сама є неповною.

Теорема 2.2. *Система зв'язок $\{\vee, \wedge, \neg\}$ є повною. Іншими словами, кожну булеву функцію $f(x_1, \ldots, x_n)$ можна задати формулою $F(x_1, \ldots, x_n)$, записаною за допомогою лише зв'язок $\{\vee, \wedge, \neg\}$. Зокрема, для кожної формули логіки висловлювань існує рівносильна їй формула, записана лише за допомогою зв'язок $\{\vee, \wedge, \neg\}$.*

Доведення. Застосуємо індукцію за кількістю n змінних. Безпосередньо перевіряємо, що кожна булева функція $f(x)$ від змінної x задається однією з формул

$$x, \quad \neg x, \quad x \vee \neg x, \quad x \wedge \neg x\,.$$

Припустимо тепер, що для функцій від n змінних теорему вже доведено. Для функції $f(x_1, \ldots, x_n, x_{n+1})$ розглянемо функції

$$g(x_1, \ldots, x_n) = f(x_1, \ldots, x_n, 0) \quad \text{і} \quad h(x_1, \ldots, x_n) = f(x_1, \ldots, x_n, 1).$$

За припущенням індукції їх можна задати формулами $G(x_1, \ldots, x_n)$ і $H(x_1, \ldots, x_n)$ відповідно. Тоді $f(x_1, \ldots, x_n, x_{n+1})$ задають формулою

$$F(x_1, \ldots, x_n, x_{n+1}) = \big(G(x_1, \ldots, x_n) \wedge \neg x_{n+1}\big) \vee \big(H(x_1, \ldots, x_n) \wedge x_{n+1}\big)\,.$$

Справді, для довільного набору $(\varepsilon_1, \ldots, \varepsilon_n)$ булевих значень

$$F(\varepsilon_1, \ldots, \varepsilon_n, 0) = \big(G(\varepsilon_1, \ldots, \varepsilon_n) \wedge \neg 0\big) \vee \big(H(\varepsilon_1, \ldots, \varepsilon_n) \wedge 0\big) \equiv$$
$$\equiv \big(G(\varepsilon_1, \ldots, \varepsilon_n) \wedge 1\big) \vee 0 \equiv G(\varepsilon_1, \ldots, \varepsilon_n) \equiv g(\varepsilon_1, \ldots, \varepsilon_n) = f(\varepsilon_1, \ldots, \varepsilon_n, 0).$$

Аналогічно доводять, що $F(\varepsilon_1, \ldots, \varepsilon_n, 1) \equiv f(\varepsilon_1, \ldots, \varepsilon_n, 1)$.

Нарешті, нехай A — довільна формула логіки висловлювань, а f — булева функція, яку вона задає. За доведеним вище f можна задати формулою F, записаною лише за допомогою зв'язок $\{\vee, \wedge, \neg\}$. Тоді $A \equiv F$. □

Зв'язки $\{\vee, \wedge, \neg\}$ називають *булевими*. Дуже часто, особливо в застосуваннях алгебри логіки в техніці, користуються заданням булевих функцій лише за допомогою цих зв'язок (повнота системи булевих зв'язок дозволяє це робити)[21]. Узагалі замість класичного набору

$$\{\neg, \vee, \wedge, \to, \leftrightarrow, \oplus\}$$

можна обмежитися будь-якою повною системою зв'язок (навіть з однієї зв'язки), і часто це зручно. Однак формули у цьому випадку стають значно громіздкішими. Та й змістовно класичні логічні дії, особливо імплікація, відіграють важливу роль.

Щоб довести повноту якоїсь системи зв'язок, досить показати, що кожна зв'язка із системи, про яку вже відомо, що вона повна, виражається через зв'язки з даної системи.

Твердження 2.1. *Кожна із систем зв'язок*: $\{\vee, \neg\}$, $\{\wedge, \neg\}$, $\{\neg, \to\}$, $\{\to, \oplus\}$, $\{0, \to\}$ *є повною.*

Доведення. Із теореми 2.2 і законів де Моргана (с. 27) випливає повнота систем $\{\vee, \neg\}$ і $\{\wedge, \neg\}$. Далі з рівносильності $x \vee y \equiv \neg x \to y$ випливає повнота системи $\{\neg, \to\}$. У свою чергу, з рівносильності $\neg x \equiv x \to (x \oplus x)$ слідує повнота системи $\{\to, \oplus\}$. Нарешті, з рівносильності $\neg x \equiv x \to (x \to 0)$ випливає повнота системи $\{0, \to\}$. □

Зауваження. *Системи* $\{\neg, \to\}$ *і* $\{0, \to\}$ *часто використовують для побудови різних варіантів числення висловлювань.*

Твердження 2.2. *Система зв'язок* $\{\vee, \wedge, \to, \leftrightarrow\}$ *є неповною.*

[21] Це викликано тим, що зв'язки $\{\vee, \wedge, \neg\}$ природно реалізуються "в залізі" за допомогою паралельного та послідовного з'єднання відповідних схем і логічних інверторів.

Доведення. Якщо формула $F(x_1, \ldots, x_n)$ побудована за допомогою зв'язок із $\{\vee, \wedge, \to, \leftrightarrow\}$, то $F(1, \ldots, 1) = 1$. Тому задати такою формулою функцію, яка на наборі $(1, \ldots, 1)$ набуває значення 0, не можна. □

Теорема 2.3. *Кожна із систем* $\{|\}$ (штрих Шефера) *і* $\{\uparrow\}$ (стрілка Пірса) *є повною. Інших повних систем, які складалися б з лише з однієї бінарної зв'язки, не існує.*

Доведення. Повнота штриху Шефера:

$$\neg a \equiv a \mid a\,, \qquad a \wedge b \equiv \neg(a \mid b) \equiv (a \mid b) \mid (a \mid b)\,.$$

Повнота стрілки Пірса:

$$\neg a \equiv a \uparrow a\,, \qquad a \vee b \equiv \neg(a \uparrow b) \equiv (a \uparrow b) \mid (a \uparrow b)\,.$$

Якщо бінарна зв'язка $x * y$ явно не залежить від змінних, то вона набуває або лише значення 1, або лише значення 0. Якщо ж вона явно залежить лише від однієї змінної, то вона або дорівнює цій змінній, або є її запереченням (і тоді не можна отримати, наприклад, функцію, яка тотожно дорівнює 1). Отже, якщо система $\{*\}$ повна, то формула $x * y$ явно залежить від обох змінних. Крім того, із міркувань, подібних до тих, що використовувалися при доведенні твердження 2.2, випливає, що мають виконуватися рівності $1*1 = 0$ і $0*0 = 1$. З таблиці на с. 13 видно, що ці умови задовольняють лише штрих Шефера і стрілка Пірса. □

2.3. Нормальні форми

Раніше вже зазначено, що ту саму булеву функцію можна задавати багатьма різними формулами логіки висловлювань. Тому природно виникає питання про вибір серед цих формул у певному сенсі найпростіших, певною мірою "канонічних представників".

Далі (і не тільки в обговоренні булевих функцій) буде корисним таке позначення: для довільних $\alpha \in \{0, 1\}$ та пропозиційної змінної a

$$a^{\alpha} := \begin{cases} a, & \text{якщо } \alpha = 1; \\ \neg a, & \text{якщо } \alpha = 0. \end{cases}$$

Вираз a^{α} називають *літералом* змінної a.

Кон'юнктивним членом (або просто *кон'юнктом*) від змінних a_1, a_2, …, a_n називають формулу вигляду

$$a_{i_1}^{\alpha_{i_1}} \wedge a_{i_2}^{\alpha_{i_2}} \wedge \cdots \wedge a_{i_k}^{\alpha_{i_k}}\,,$$

в якій зустрічаються змінні лише зі списку a_1, a_2, …, a_n, причому жодна змінна не зустрічається двічі.

Аналогічно визначають *диз'юнктивний член* (або просто *диз'юнкт*)

$$a_{i_1}^{\alpha_{i_1}} \vee a_{i_2}^{\alpha_{i_2}} \vee \cdots \vee a_{i_k}^{\alpha_{i_k}}.$$

Кон'юнктивний (диз'юнктивний) член від змінних $a_1, a_2, \ldots, a_n$ називають *елементарним*, якщо він містить усі змінні із цього списку.

Зрозуміло, що два кон'юнктивні (диз'юнктивні) члени будуть рівносильними тоді й лише тоді, коли вони містять однакові літерали і розрізняються лише порядком цих літералів.

Кажуть, що формула логіки висловлювань має *нормальну форму* (коротко: НФ), якщо вона містить лише булеві зв'язки, причому знак заперечення $\neg$ зустрічається лише біля змінних. Із доведення теореми 2.2 випливає, що для кожної формули існує рівносильна їй НФ.

Частковими випадками нормальної форми є так звані *диз'юнктивна нормальна форма* (коротко: ДНФ), яка є диз'юнкцією кон'юнктивних членів, причому серед цих членів немає рівносильних, і *кон'юнктивна нормальна форма* (коротко: КНФ), яка є кон'юнкцією диз'юнктивних членів, причому серед цих членів немає рівносильних.

Диз'юнктивну (відповідно кон'юнктивну) нормальну форму називають *досконалою*, якщо всі її кон'юнктивні (відповідно диз'юнктивні) члени є елементарними. Коротко писатимемо ДДНФ (відповідно ДКНФ).

Якщо додатково домовитися, що диз'юнкцією порожньої множини кон'юнктивних членів є тотожно хибна формула 0, а кон'юнкцією порожньої множини диз'юнктивних членів є тотожно істинна формула 1, то легко доводиться наступна теорема.

Теорема 2.4. *Для кожної формули логіки висловлювань існують рівносильні їй ДДНФ і ДКНФ.*

Доведення. Нехай $F(x_1, \ldots, x_n)$ — довільна формула, а $\mathcal{O}(F)$ — її область істинності. Елементарна кон'юнкція $x_1^{\alpha_1} \wedge x_2^{\alpha_2} \wedge \cdots \wedge x_n^{\alpha_n}$ набуває значення 1 лише на одному булевому векторі $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, а елементарна диз'юнкція $x_1^{1-\alpha_1} \vee x_2^{1-\alpha_2} \vee \cdots \vee x_n^{1-\alpha_n}$ лише на цьому векторі набуває значення 0. Тому

$$F(x_1, \ldots, x_n) \equiv \bigvee_{(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathcal{O}(F)} x_1^{\alpha_1} \wedge x_2^{\alpha_2} \wedge \cdots \wedge x_n^{\alpha_n} \equiv$$

$$\equiv \bigwedge_{(\alpha_1, \alpha_2, \ldots, \alpha_n) \notin \mathcal{O}(F)} x_1^{1-\alpha_1} \vee x_2^{1-\alpha_2} \vee \cdots \vee x_n^{1-\alpha_n}.$$

□

З доведення теореми 2.4 випливає, що ДДНФ (ДКНФ) даної формули повністю визначається її областю істинності (хибності). Точніше, виконується твердження 2.3.

Твердження 2.3. *Із точністю до порядку елементарних кон'юнкцій (диз'юнкцій) та порядку літералів у цих елементарних кон'юнкціях (диз'юнкціях) ДДНФ (ДКНФ) даної формули визначена однозначно.*

Зауваження. *Є певна аналогія між ДДНФ і ДКНФ формули з одного боку, й записом полінома у вигляді суми одночленів і добутку незвідних множників — із другого (особливо у випадку алгебрично замкненого поля).*

Знаходження нормальної форми, рівносильної формулі F, називають *зведенням* F до нормальної форми. Аналогічно визначають зведення до ДНФ, КНФ, ДДНФ і ДКНФ. Доведення теореми 2.4 дає метод зведення до ДДНФ і ДКНФ за допомогою обчислення областей істинності та хибності формули. Інший підхід до зведення формул логіки висловлювань до нормальної форми використовує рівносильні перетворення і ґрунтується на теоремі 1.5.

Алгоритм зведення формули логіки висловлювань до нормальної форми.

1. Замінюємо кожну підформулу вигляду $A \oplus B$ на підформулу

$$(A \vee B) \wedge (\neg A \vee \neg B).$$

2. Замінюємо кожну підформулу вигляду $A \leftrightarrow B$ на підформулу

$$(A \wedge B) \vee (\neg A \wedge \neg B).$$

3. Замінюємо кожну підформулу вигляду $A \to B$ на підформулу $\neg A \vee B$.

4. Замінюємо кожну підформулу вигляду $\neg(A \wedge B)$ на підформулу $\neg A \vee \neg B$.

5. Замінюємо кожну підформулу вигляду $\neg(A \vee B)$ на підформулу $\neg A \wedge \neg B$.

6. Замінюємо кожну підформулу вигляду $\neg\neg A$ на підформулу A.

7. Якщо отримана формула має нормальну форму, то закінчуємо роботу; у протилежному разі повертаємось до п. 4.

Зауваження. *Якщо одна з підформул вигляду $A \oplus B$ є частиною іншої, то застосовувати перший пункт починаємо з внутрішньої підформули. Аналогічні зауваження стосуються і решти пунктів.*

Після виконання перших трьох кроків алгоритму формула міститиме лише булеві зв'язки. А після виконання наступних кроків знаки заперечення опустяться до змінних. Тому на виході алгоритму ми одержимо нормальну форму. Теорема 1.5 гарантує, що отримана НФ рівносильна початковій формулі.

Алгоритм зведення нормальної форми до диз'юнктивної нормальної форми.

1. За допомогою дистрибутивного закону $A\wedge(B\vee C)\equiv(A\wedge B)\vee(A\wedge C)$ зводимо формулу до диз'юнкції членів, кожен з яких є кон'юнкцією літералів.

2. За допомогою рівносильностей $A\wedge A\equiv A,\ \ A\wedge\neg A\equiv 0$ видаляємо лишні літерали в кон'юнктивних членах.

3. За допомогою рівносильностей $A\vee A\equiv A,\ \ A\vee\neg A\equiv 1$ видаляємо лишні кон'юнктивні члени.

4. За допомогою законів поглинання константами $A\vee 1\equiv 1,\ \ A\wedge 1\equiv A,$ $A\vee 0\equiv A,\ \ A\wedge 0\equiv 0$ видаляємо константи.

Зауваження. *ДНФ уже легко звести до ДДНФ. Для цього досить "розщепити" ті кон'юнктивні члени, які містять не всі змінні:*

$$A\equiv A\wedge 1\equiv A\wedge(x\vee\neg x)\equiv(A\wedge x)\vee(A\wedge\neg x)\,,$$

а потім з отриманої формули видалити лишні елементарні кон'юнкції.

Вправа 2.1. *Запропонуйте аналогічні алгоритми зведення нормальної форми до КНФ і до ДКНФ.*

Теорема 2.5. *Для довільних двох рівносильних формул F_1 та F_2 логіки висловлювань можна перейти від F_1 до F_2 за допомогою рівносильних перетворень, що спираються на такі рівносильності*:

1) $A\oplus B\equiv(A\vee B)\wedge(\neg A\vee\neg B)$;

2) $A\leftrightarrow B\equiv(A\wedge B)\vee(\neg A\wedge\neg B)$;

3) $A\to B\equiv\neg A\vee B$;

4) $\neg\neg A\equiv A$ (*закон подвійного заперечення*);

5) $\neg(A\vee B)\equiv\neg A\wedge\neg B,\ \ \neg(A\wedge B)\equiv\neg A\vee\neg B$ (*закони де Моргана*);

6) $A\wedge(B\vee C)\equiv(A\wedge B)\vee(A\wedge C),\ \ A\vee(B\wedge C)\equiv(A\vee B)\wedge(A\vee C)$ (*дистрибутивні закони*);

7) $(A\vee B)\vee C\equiv A\vee(B\vee C),\ \ (A\wedge B)\wedge C\equiv A\wedge(B\wedge C)$ (*асоціативні закони*);

8) $A\vee B\equiv B\vee A,\ \ A\wedge B\equiv B\wedge A$ (*комутативні закони*);

9) $A\vee A\equiv A,\ \ A\wedge A\equiv A$ (*закони ідемпотентності*);

10) $A\wedge\neg A\equiv 0$ (*закон суперечності*);

11) $A\vee\neg A\equiv 1$ (*закон виключення третьої можливості*);

12) $A\vee 1\equiv 1,\ \ A\wedge 1\equiv A,\ \ A\vee 0\equiv A,\ \ A\wedge 0\equiv 0$ (*закони для констант*).

Доведення. З описаних вище алгоритмів випливає, що за допомогою вказаних рівносильностей кожну формулу можна звести до рівносильної їй ДДНФ.

Зведемо тепер F_1 та F_2 до ДДНФ. Оскільки $F_1 \equiv F_2$, то з доведення теореми 2.4 і твердження 2.3 випливає, що їхні ДДНФ відрізняються щонайбільше порядком елементарних кон'юнкцій і порядком літералів у цих елементарних кон'юнкціях. Тому за допомогою комутативних законів для кон'юнкції та диз'юнкції можна зробити ці ДДНФ однаковими. Розвертаючи у протилежний бік перетворення, які використовувалися при зведенні до ДДНФ формули F_2, одержимо ланцюжок рівносильних перетворень від F_1 до F_2. □

Теорема 2.6. *Булеву алгебру задають скінченним списком аксіом.*

Доведення. Теорема 2.5 стверджує, що кожну рівносильність логіки висловлювань виводять з наведеного в цій теоремі скінченного списку рівносильностей. Їх і можна взяти за аксіоми булевої алгебри. □

Набір з усіх аксіом із теореми 2.5 є дуже надлишковим. Можна, наприклад, лишити лише по 2 асоціативних, комутативних і дистрибутивних закони (із двох дистрибутивних насправді один також можна вилучити, але це вже не зовсім тривіально), а крім них ще

7) $\neg\neg x = x$, 8) $\neg(x \vee y) = \neg x \wedge \neg y$, 9) $x \wedge \neg x = 0$, 10) $x \vee \neg x = 1$, 11) $0 \vee x = x$.

Решта аксіом із них виводиться:

a) $\neg(x \wedge y) = \neg(\neg\neg x \wedge \neg\neg y) = \neg(\neg(\neg x \vee \neg y)) = \neg\neg(\neg x \vee \neg y) = \neg x \vee \neg y$;

b) $\neg 1 = \neg(x \vee \neg x) = \neg x \wedge \neg\neg x = \neg x \wedge x = 0$;
$\neg 0 = \neg(x \wedge \neg x) = \neg x \vee \neg\neg x = \neg x \vee x = 1$;

c) $1 \wedge x = \neg\neg(1 \wedge x) = \neg(\neg 1 \vee \neg x) = \neg(0 \vee \neg x) = \neg\neg x = x$;

d) $x = x \wedge 1 = x \wedge (x \vee \neg x) = (x \wedge x) \vee (x \wedge \neg x) = (x \wedge x) \vee 0 = x \wedge x$;

e) $x = x \vee 0 = x \vee (x \wedge \neg x) = (x \vee x) \wedge (x \vee \neg x) = (x \vee x) \wedge 1 = x \vee x$;

f) $0 \wedge x = (\neg x \wedge x) \wedge x = \neg x \wedge (x \wedge x) = \neg x \wedge x = 0$;

g) $1 \vee x = (\neg x \vee x) \vee x = \neg x \vee (x \vee x) = \neg x \vee x = 1$.

Зауваження. *У літературі можна знайти багато інших варіантів аксіом булевої алгебри. Зазвичай вони також не є незалежними. Але насамперед звертають увагу на зручність аксіом для користування, що набагато переважає недоліки, пов'язані з їхньою надлишковістю*[22].

Булеві функції є зручним апаратом для розв'язування багатьох прикладних задач (напр., у теорії проєктування ЕОМ). Зокрема, у проєктуванні тих же ЕОМ виникає задача реалізації конкретних булевих функцій (часто — від

[22] Загальноприйняті в алгебрі аксіоми групи чи векторного простору також не є незалежними.

великої кількості змінних) "у залізі". Зазвичай у таких випадках булеві функції задають нормальними формами того чи іншого типу. Але природне для математика задання функцій досконалими формами з погляду інженера є дуже неекономним: для переважної кількості булевих функцій існують значно коротші ДНФ або КНФ, які їх реалізують. Тому виникає задача пошуку короткої (бажано — найкоротшої) нормальної форми, яка реалізує дану булеву функцію — так звана *задача мінімізації* булевої функції.

Приклади. 1. *Однією з ДНФ для формули* $F = (x \wedge \neg y) \leftrightarrow (x \vee z)$ *є* $D = (x \wedge \neg y) \vee (\neg x \wedge \neg z)$ *(для кожної із цих формул область істинності складається з векторів* $(0,0,0)$, $(0,1,0)$, $(1,0,0)$ *і* $(1,0,1)$*). Покажемо, що* D *є мінімальною ДНФ, тобто такою, що містить найменшу можливу кількість літералів. Справді,* F *явно залежить від кожної змінної, тому її ДНФ не може містити менше 3 літералів. ДНФ від змінних* u, v, w*, яка містить усі змінні й лише* 3 *літерали, має вигляд* $u^\alpha \wedge v^\beta \wedge w^\gamma$*,* $(u^\alpha \wedge v^\beta) \vee w^\gamma$ *або* $u^\alpha \vee v^\beta \vee w^\gamma$*. Але в першому випадку область істинності містить* 1 *вектор, у другому —* 5*, а в третьому —* 7*. А область істинності формули* F *містить* 4 *вектори.*

Зауважимо, що мінімальна ДНФ для формули F із прикладу 1 містить 4 літерали, тоді як її ДДНФ містить 12 літералів. Але бувають випадки, коли нічого коротшого за ДДНФ не існує.

2. *Для формули* $F = (x \leftrightarrow y) \leftrightarrow z$ *мінімальною ДНФ буде її ДДНФ* $L = (x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (x \wedge y \wedge z$*. Справді, область істинності формули* F *складається з векторів* $(0,0,1)$, $(0,1,0)$, $(1,0,0)$ *і* $(1,1,1)$*), будь-які два з яких розрізняються у двох позиціях. Коротша ДНФ повинна містити кон'юнкт з одного або двох літералів. Але за наявності такого кон'юнкта область істинності міститиме два вектори, які розрізняються лише в одній позиції.*

Свого часу (у 50—60-х рр. минулого століття), коли електронні схеми будувалися з ламп, транзисторів, реле і подібних громіздких і дорогих компонентів, питанням мінімізації булевих функцій надавали величезне значення. Із цього приводу опубліковано багато книг і ледь не тисячі статей. Але технічний прогрес перевершив усі сподівання, електронні схеми стрімко зменшувалися в розмірах і не менш стрімко дешевшали. Тому поступово проблема мінімізації булевих функцій втратила свою колишню актуальність.

Поруч із нормальними формами для зображення булевих функцій інколи використовують так звані *поліноми Жегалкіна*. Вони визначаються так.

Одночленом від змінних $x_1, \ldots, x_n$ називають логічну константу або вираз вигляду $x_{i_1} \wedge x_{i_2} \wedge \cdots \wedge x_{i_k}$, де $1 \le i_1 < \cdots < i_k \le n$. *Поліномом Жегалкіна* називають альтернативу попарно різних одночленів:

$$f(x_1, \ldots, x_n) \equiv \bigoplus_{1 \le i_1 < \cdots < i_k \le n} \alpha_{i_1 \ldots i_k} x_{i_1} \wedge \cdots \wedge x_{i_k},$$

де $0 \le k \le n$, $\alpha_{i_1 \ldots i_k} \in \{0, 1\}$.

Теорема 2.7. *Кожну булеву функцію однозначно зображують поліномом Жегалкіна.*

Доведення. Оскільки $1 \oplus x \equiv \neg x$, то $1, \wedge, \oplus$ — повна система логічних зв'язок.

З асоціативності й комутативності зв'язок $\wedge$ і $\oplus$ та рівносильностей

$$a \wedge (b \oplus c) \equiv a \wedge b \oplus a \wedge c, \quad a \oplus a \equiv 0, \quad 0 \oplus a \equiv a, \quad 0 \wedge a \equiv 0$$

випливає існування полінома Жегалкіна.

Єдиність випливає з того, що кількість булевих функцій і поліномів Жегалкіна від n змінних однакова і дорівнює 2^{2^n}. □

2.4. Критерій повноти системи зв'язок

Щоб описати всі повні системи булевих функцій, нам знадобляться такі п'ять класів функцій:

a) клас T_0 функцій, що зберігають 0 (тобто $f(0, \ldots, 0) = 0$);

b) клас T_1 функцій, що зберігають 1 (тобто $f(1, \ldots, 1) = 1$);

c) клас L лінійних функцій (функція називається *лінійною*, якщо вона зображується поліном Жегалкіна вигляду $c_0 \oplus x_{i_1} \oplus \cdots \oplus x_{i_k}$, $1 \le i_1 < \cdots < i_k \le n$);

d) клас D самодвоїстих функцій (функція f називається *самодвоїстою*, якщо $f^* = f$);

e) клас M монотонних функцій.

Щоб описати цей клас, для кожного n визначимо на множині булевих векторів довжини n такий частковий порядок:

$(\alpha_1, \ldots, \alpha_n) \ge (\beta_1, \ldots, \beta_n)$, якщо для всіх i з $\alpha_i = 0$ випливає $\beta_i = 0$.

Булева функція $f(x_1, \ldots, x_n)$ буде *монотонною*, якщо із

$$(\alpha_1, \ldots, \alpha_n) \ge (\beta_1, \ldots, \beta_n) \quad \text{випливає} \quad f(\alpha_1, \ldots, \alpha_n) \ge f(\beta_1, \ldots, \beta_n).$$

Твердження 2.4. *Кожен із класів T_0, T_1, L, D і M є замкненим відносно суперпозиції.*

Доведення. Нехай $h(x_1,\dots,x_n) = g\big(f_1(x_1,\dots,x_n),\dots,f_m(x_1,\dots,x_n)\big)$.

Замкненість відносно суперпозиції класів T_0 і T_1 очевидна.

c) Замкненість класу L випливає з асоціативності та комутативності зв'язки $\oplus$ і рівносильностей $x \oplus x \equiv 0$ та $0 \oplus 1 \equiv 1$.

d) Нехай функції f_1, …, f_m та g — самодвоїсті. Згідно з лемою 1.1 це означає, що

$$f_i \equiv \neg f_1(\neg x_1,\dots,\neg x_n), \quad i = 1,\dots,m;$$
$$g \equiv \neg g(\neg y_1,\dots,\neg y_m).$$

Але тоді

$$\neg h(\neg x_1,\dots,\neg x_n) \equiv \neg g\big(f_1(\neg x_1,\dots,\neg x_n),\dots,f_m(\neg x_1,\dots,\neg x_n)\big) \equiv$$
$$\equiv \neg g\big(\neg f_1(x_1,\dots,x_n),\dots,\neg f_m(x_1,\dots,x_n)\big) \equiv$$
$$\equiv g\big(f_1(x_1,\dots,x_n),\dots,f_m(x_1,\dots,x_n)\big) \equiv h(x_1,\dots,x_n).$$

Отже, згідно з лемою 1.1, функція $h(x_1,\dots,x_n)$ також самодвоїста.

e) Нехай функції f_1, …, f_m та g монотонні і $(\alpha_1,\dots,\alpha_n) \geq (\beta_1,\dots,\beta_n)$. Тоді $f_k(\alpha_1,\dots,\alpha_n) \geq f_k(\beta_1,\dots,\beta_n)$ для всіх $k = 1,\dots,m$, а тому

$$h(\alpha_1,\dots,\alpha_n) = g\big(f_1(\alpha_1,\dots,\alpha_n),\dots,f_m(\alpha_1,\dots,\alpha_n)\big) \geq$$
$$\geq g\big(f_1(\beta_1,\dots,\beta_n),\dots,f_m(\beta_1,\dots,\beta_n)\big) = h(\beta_1,\dots,\beta_n).$$

Отже, функція h також монотонна. □

Зауважимо, що для функцій від n аргументів приналежність функції f до класу T_0 або T_1 накладає обмеження на значення функції f лише в одній точці. Тому

$$|T_0| = |T_1| = 2^{2^n-1}.$$

У лінійної функції $c_0 \oplus x_{i_1} \oplus \dots \oplus x_{i_k}$ параметр c_0 може набувати двох значень 0 і 1, а набір змінних $x_{i_1},\dots,x_{i_k}$ може бути довільною підмножиною (зокрема, й порожньою) із $\{x_1,\dots,x_n\}$. Тому

$$|L| = 2 \cdot 2^n = 2^{n+1}.$$

Згідно з лемою 1.1 самодвоїста функція на векторах $(0,\alpha_2,\dots,\alpha_n)$ і $(1,\neg\alpha_2,\dots,\neg\alpha_n)$ повинна набувати протилежних значень. Оскільки таких пар векторів 2^{n-1}, то $|D| = 2^{2^{n-1}}$. Підрахунок кількості монотонних функцій

від n аргументів — досі нерозв'язана задача[23]. Відомі лише різні оцінки згори і знизу.

Теорема 2.8 (критерій Поста)**.** *Система A булевих функцій буде повною тоді й лише тоді, коли A не міститься в жодному з класів T_0, T_1, L, D і M* (тобто система A повинна містити хоча б одну функцію, яка не зберігає 0, хоча б одну функцію, яка не є лінійною, і т. д.).

Доведення. Необхідність умови випливає з твердження 2.4.

Нехай тепер $f_0 \notin T_0,\ f_1 \notin T_1,\ f_l \notin L,\ f_d \notin D,\ f_m \notin M$ (функції $f_0, \ldots, f_m$ — не обов'язково різні).

a) Спочатку будуємо константи. Тут можливі два випадки. Якщо $f_0(1, \ldots, 1) = 1$, то $f_0(x, \ldots, x) \equiv 1$ і далі за допомогою суперпозиції будуємо $0 = f_1(1, \ldots, 1)$.

Якщо ж $f_0(1, \ldots, 1) = 0$, то $f_0(x, \ldots, x) \equiv \neg x$. Розглянемо функцію f_d. Вона несамодвоїста, тому існує такий набір $(\alpha_1, \ldots, \alpha_n)$, що

$$f_d(\neg\alpha_1, \ldots, \neg\alpha_n) = f_d(\alpha_1, \ldots, \alpha_n).$$

Тоді функція $g(x) = f_d(x^{\alpha_1}, \ldots, x^{\alpha_n})$ є константою. Іншу константу дає функція $f_0(g(x), \ldots, g(x)) \equiv \neg g(x)$.

b) Далі за допомогою констант 0, 1 і функції f_m будуємо заперечення $\neg x$. Для цього візьмемо такі набори $\overline{\alpha} = (\alpha_1, \ldots, \alpha_n)$ і $\overline{\beta} = (\beta_1, \ldots, \beta_n)$, що $\overline{\alpha} > \overline{\beta}$, але $f_m(\overline{\alpha}) < f_m(\overline{\beta})$. Від $\overline{\alpha}$ до $\overline{\beta}$ можна перейти, кожного разу змінюючи один 0 на 1. Тому можна вважати, що $\overline{\alpha}$ і $\overline{\beta}$ розрізняються лише k-ю компонентою: $\overline{\alpha} = (\ldots, 1, \ldots)$ і $\overline{\beta} = (\ldots, 0, \ldots)$. Тоді $f_m(\ldots, x, \ldots) \equiv \neg x$.

c) Нарешті за допомогою 0, 1, заперечення і нелінійної функції f_l будуємо кон'юнкцію $x \wedge y$. Нехай поліном Жегалкіна для f_l має вигляд

$$f_l = x_1 \wedge x_2 \wedge g_1(x_3, \ldots, x_n) \oplus x_1 \wedge g_2(x_3, \ldots, x_n) \oplus \\ \oplus x_2 \wedge g_3(x_3, \ldots, x_n) \oplus g_4(x_3, \ldots, x_n),$$

причому $g_1(x_3, \ldots, x_n) \not\equiv 0$. Тому існує такий набір $(\alpha_3, \ldots, \alpha_n)$, що

$$g_1(\alpha_3, \ldots, \alpha_n) = 1.$$

Тоді для елементів $a = g_2(\alpha_3, \ldots, \alpha_n)$, $b = g_3(\alpha_3, \ldots, \alpha_n)$, $c = g_4(\alpha_3, \ldots, \alpha_n)$ маємо

$$f_l(x_1, x_2, \alpha_3, \ldots, \alpha_n) = x_1 \wedge x_2 \oplus ax_1 \oplus bx_2 \oplus c \equiv \\ \equiv (x_1 \oplus b) \wedge (x_2 \oplus a) \oplus (a \wedge b \oplus c).$$

[23] Кількість монотонних булевих функцій від n змінних називають *числом Дедекінда* D_n. Ці числа з'являються і в інших розділах алгебри і комбінаторного аналізу як, наприклад, кількість односторонніх ідеалів у симетричній інверсній напівгрупі степеня n або порядок вільної дистрибутивної ґратки рангу n.

Оскільки $x \oplus 0 \equiv x$ і $x \oplus 1 \equiv \neg x$, то ми можемо отримати $x_1 \oplus b$, $x_2 \oplus a$ і $f_l \oplus (a \wedge b \oplus c)$. Після відповідних підстановок у попередню рівність отримуємо

$$f_l(x_1 \oplus b, x_2 \oplus a, \alpha_3, \ldots, \alpha_n) \oplus (a \wedge b \oplus c) \equiv$$
$$\equiv (x_1 \oplus b \oplus b) \wedge (x_2 \oplus a \oplus a) \oplus (a \wedge b \oplus c) \oplus (a \wedge b \oplus c) \equiv x_1 \wedge x_2.$$

Врахувавши, що система зв'язок $\{\wedge, \neg\}$ є повною, теорему доведено. □

Повну систему зв'язок називають *базою*, якщо після вилучення будь-якої зв'язки вона перестає бути повною. Зокрема, такими є системи

$$\{\vee, \neg\},\ \{\wedge, \neg\},\ \{\neg, \rightarrow\},\ \{|\}\ i\ \{\uparrow\}.$$

Твердження 2.5. *Кожна база містить не більше чотирьох функцій.*

Доведення. Згідно з критерієм Поста повна система повинна містити п'ять функцій: $f_0 \notin T_0,\ f_1 \notin T_1,\ f_l \notin L,\ f_d \notin D,\ f_m \notin M$.

Розглянемо f_0. Можливі два випадки:
1. $f_0 \in T_1$. Тоді $f_0 \notin D$ і система (f_0, f_1, f_m, f_l) є повною.
2. $f_0 \notin T_1$. Тоді $f_0 \notin M, T_1$ і система (f_0, f_d, f_l) є повною. □

Бази із чотирьох функцій існують. Наприклад, такою є система зв'язок $0 \notin T_1, D$; $1 \notin T_0;\ x \wedge y \notin L;\ x \oplus y \oplus z \notin M$.

Вправа 2.2. *Доведіть, що є рівно* 17 *баз із функцій арності* ≤ 2.

3. Логіка предикатів

3.1. Висловлювальні функції та предикати

За допомогою логіки висловлювань можна обґрунтувати далеко не всі види правильних міркувань. Наведемо типове для математики міркування.

Кожна монотонна й обмежена послідовність має границю. Послідовність $x_n = \left(1 + \frac{1}{n}\right)^n$ *є монотонною й обмеженою. Отже, ця послідовність має границю.*

Воно правильне. Але за допомогою логіки висловлювань його обґрунтувати не можна. Причина полягає в тому, що для обґрунтування правильності цього міркування треба вже враховувати внутрішню будову простих речень, з яких воно складається. Проте логіку висловлювань цікавить лише значення істинності простих речень, будова цих речень її не цікавить.

Дослідженням правильності різних форм міркувань з урахуванням і внутрішньої структури простих речень займається логіка предикатів.

Основним поняттям, яке дозволяє враховувати внутрішню структуру речень, є поняття *предиката*. У граматиці під терміном "предикат" розуміють таке слово (чи зворот) у реченні, який виражає те, що говориться про суб'єкт (підмет). Наприклад: "літає", "є парним числом", "має зелений колір", "ділить число 100" тощо (тобто зміст цього терміна близький до змісту терміна "присудок", хоча й дещо ширший).

Вправа 3.1. *Що буде предикатом у реченні "Усі студенти час від часу пропускають лекції"?*

У логіці термін "предикат" розуміють у трохи ширшому значенні. Тут предикат включає і змінну, яка позначає предмет, про який ідеться: "u літає", "x є парним числом", "y має зелений колір", "z ділить число 100" ... Підставляючи замість змінної той чи інший елемент із відповідної області визначення A, одержуємо висловлювання, яке може бути істинним або хибним. Тим самим одержуємо функцію, яка кожному елементу з A ставить у відповідність певне висловлювання — так звану *висловлювальну функцію*. Якщо нас цікавить лише значення істинності цих висловлювань, то одержуємо функцію з A у множину логічних значень $\{0, 1\}$. Цю останню функцію і називають *предикатом*[24].

[24] Поняття висловлювальної функції і предиката трохи розрізняються: значеннями висловлювальної функції є *висловлювання*, а значеннями відповідного предиката — *значення істинності* цих висловлювань. Однак зараз ми ці поняття трактуватимемо як майже синоніми, а пізніше нас цікавитимуть виключно предикати.

Одразу ж напрошується узагальнення цього поняття на висловлювальні функції від кількох змінних: "прямі a і b паралельні", "z є сумою x і y" тощо. Це приводить до поняття n-місної висловлювальної функції. З певних причин, які стануть зрозумілими трохи пізніше, до можливих значень n зручно долучити і 0: 0-місна висловлювальна функція не містить аргументів, а тому є просто висловлюванням.

Отже, висловлювальні функції від однієї змінної описують властивості елементів множини A, а від кількох змінних — властивості наборів елементів, тобто відношення між елементами множини A.

Відомому польсько-американському логіку А.Тарському належить дуже вдале порівняння висловлювальної функції з незаповненим бланком документа, порожні місця в якому відповідають змінним. Щоб бланк став документом, порожні місця треба заповнити. Аналогічно висловлювальна функція стає висловлюванням, якщо всім її змінним присвоєно конкретні значення[25].

Виявляється, що за допомогою поняття висловлювальної функції (точніше — її формального відповідника — предиката) вже можна описати й дослідити практично всі форми правильних міркувань[26]. Тому перейдемо до строгих означень.

n-місним (або *n-арним*) *предикатом* на непорожній множині A називається довільне відображення множини $A^n = \underbrace{A \times \cdots \times A}_{n}$ у множину $\{0, 1\}$ логічних значень.

Зручно для кожної непорожньої множини A покласти $A^0 = \{\varnothing\}$. Тоді $|A^0| = 1$. Відповідно до цієї домовленості кожна 0-арна функція $A^0 \to N$ фактично вибирає певний елемент з N. Тому її можна ототожнити із цим елементом. Отже, кожен 0-арний предикат на A — це певний фіксований елемент з $\{0, 1\}$, тобто певна логічна константа.

Зрозуміло, що кожна висловлювальна функція $F(x_1, \ldots, x_n)$ від n змінних на множині A визначає деякий n-місний предикат $P_F : A^n \to \{0, 1\}$ на A. Проте різні висловлювальні функції можуть визначати один і той же предикат.

З кожним n-місним предикатом P цілком природно пов'язується множина

$$O_P = \{(a_1, \ldots, a_n) \in A^n \mid P(a_1, \ldots, a_n) = 1\},$$

[25] Аналогія посилюється тим, що висловлювальні функції ще називають *висловлювальними формами*, а порожні бланки також часто називають *формами*.

[26] Звичайно, лише одного поняття предиката ще не досить. Потрібен ще цілий ряд пов'язаних із предикатами понять. Це насамперед *терми* — імена суб'єктів, властивості яких позначають предикати, та *квантори* — операції над предикатами.

яку називають *областю істинності* предиката P. На O_P можна дивитися як на n-арне відношення на множині A. Тим самим між n-арними предикатами на A і n-арними відношеннями на A встановлюють взаємно однозначну відповідність.

Зазначимо, що n-арний предикат P на множині A називають *тотожно істинним*, якщо $O_P = A^n$, і *тотожно хибним*, якщо $O_P = \varnothing$. Предикат, який не є тотожно хибним, називають *виконливим*.

Дії над висловлюваннями легко поширюються на довільні висловлювальні функції і предикати, визначені на тій самій множині A.

Запереченням визначеної на множині A висловлювальної функції $P(x_1, \ldots, x_n)$ називають визначену також на A висловлювальну функцію такої ж арності (позначається $\neg P(x_1, \ldots, x_n)$), значенням якої на довільному наборі $(a_1, \ldots, a_n)$ є заперечення висловлювання $P(a_1, \ldots, a_n)$.

Нехай $P(x_1, \ldots, x_n)$ і $Q(x_1, \ldots, x_n)$ — дві висловлювальні функції на множині A (без обмеження загальності можна вважати, що вони залежать від тих самих змінних), а $\circ$ — одна з бінарних зв'язок $\vee$, $\wedge$, $\oplus$, $\to$ або $\leftrightarrow$. Тоді через $P(x_1, \ldots, x_n) \circ Q(x_1, \ldots, x_n)$ позначають визначену на A висловлювальну функцію арності n, значенням якої на довільному наборі $(a_1, \ldots, a_n)$ є висловлювання $P(a_1, \ldots, a_n) \circ Q(a_1, \ldots, a_n)$.

Цілком аналогічно визначають відповідні дії і над предикатами. Назви дій — диз'юнкція, кон'юнкція тощо — також зберігають.

Подібним чином (поточково) на висловлювальні функції і предикати можна переносити й інші операції над висловлюваннями.

Якби все обмежувалося лише такими діями, то логіка предикатів була б не набагато змістовнішою за логіку висловлювань. Але над предикатами є дві принципово нові операції, хоча й дуже природні. Це операції *навішування кванторів*[27], які є формальними відповідниками мовних зворотів "для всіх ..." та "існує ...".

Зауважимо, що коли $P(x, x_1, \ldots, x_n)$ є предикатом арності $n+1$ на множині A, то кожен фіксований набір $(a_1, \ldots, a_n)$ елементів з A визначає предикат $P(x, a_1, \ldots, a_n)$ арності 1 від змінної x.

Формальним відповідником звороту "для всіх ..." (і подібних до нього) є *квантор загальності* $\forall$. Результатом навішування на предикат $P(x, x_1, \ldots, x_n)$

[27] Хоча неявно квантори загальності й існування використовували в математиці віддавна, першим формалізував ці поняття в логіці німецький логік Фреге 1879 р. А термін "квантор" увів 1885 р. американський логік Пірс.

квантора $\forall$ за змінною x є предикат $\forall x\, P(x, x_1, \ldots, x_n)$ від змінних $x_1, \ldots, x_n$, значення якого на наборі $(a_1, \ldots, a_n)$ визначається так:

$$\forall x\, P(x, a_1, \ldots, a_n) = \begin{cases} 1, & \text{якщо предикат } P(x, a_1, \ldots, a_n) \text{ є} \\ & \text{тотожно істинним;} \\ 0, & \text{в інших випадках.} \end{cases} \tag{3.1}$$

Зокрема, при $n = 0$ отримуємо предикат $\forall x\, P(x)$ арності 0, тобто деяку логічну константу.

Якщо x не зустрічається серед змінних предиката P, то навішування на P квантора загальності за змінною x цей предикат не змінює. Тобто в цьому випадку P і $\forall x\, P$ — це та ж функція $A^n \to \{0, 1\}$ (де n — арність предиката P).

Формальним відповідником звороту "існує ..." є *квантор існування* $\exists$. Операцію навішування цього квантора визначають схожим чином: результатом його навішування за змінною x на предикат $P(x, x_1, \ldots, x_n)$ є предикат $\exists x\, P(x, x_1, \ldots, x_n)$ від змінних $x_1, \ldots, x_n$, значення якого на наборі $(a_1, \ldots, a_n)$ визначається так:

$$\exists x\, P(x, a_1, \ldots, a_n) = \begin{cases} 0, & \text{якщо предикат } P(x, a_1, \ldots, a_n) \text{ є} \\ & \text{тотожно хибним;} \\ 1, & \text{в інших випадках.} \end{cases} \tag{3.2}$$

Подібно як для квантора $\forall$, при $n = 0$ отримуємо предикат $\exists x\, P(x)$ арності 0, тобто логічну константу, а якщо x не зустрічається серед змінних предиката P, то навішування на P квантора $\exists$ за цією змінною предикат P не змінює.

Легко бачити, що для скінченної множини $A = \{m_1, \ldots, m_k\}$

$$\forall x\, P(x, a_1, \ldots, a_n) \equiv P(m_1, a_1, \ldots, a_n) \wedge \cdots \wedge P(m_k, a_1, \ldots, a_n)$$

і

$$\exists x\, P(x, a_1, \ldots, a_n) \equiv P(m_1, a_1, \ldots, a_n) \vee \cdots \vee P(m_k, a_1, \ldots, a_n).$$

Тому для скінченних множин квантори $\forall$ і $\exists$ є лише зручними позначеннями для кон'юнкцій і диз'юнкцій спеціального вигляду[28]. Проте в нескінченному випадку звести навішування кванторів до простіших операцій уже не можна.

[28] На перший погляд для скінченних множин нема нічого нового. Однак використання зручних позначень уже саме собою тут може мати велике значення. Згадаймо, що нинішній розквіт математики був би неможливим без буквенної символіки, яка фактично почала створюватися лише з кінця XVI ст.

3.2. Мови першого порядку

Одним з основних етапів дослідження будь-якої мови є її граматичний аналіз: розклад елементів мови (слів, речень, текстів тощо) на складові компоненти і дослідження способів поєднання цих компонентів. У дослідженні живих мов ідуть згори донизу: від речень і слів до складів і букв. У дослідженні мов, які використовують у логіці, зручно рухатися навпаки.

Серед елементів мови виділимо найпростіші (тобто такі, подальший розклад яких на граматично змістовні компоненти вже не є можливим) — *букви*. Під буквами ми розуміємо тут певні абстрактні символи, а не їхні конкретні графічні зображення, наприклад, на папері чи камені. Сукупність усіх букв даної мови називають її *алфавітом*. Під час дослідження живої мови, наприклад, української чи англійської, її алфавіт можна описати лише після глибокого аналізу мови, а побудову *мов першого порядку* зручніше почати одразу з алфавіту.

Мови, які ми розглядатимемо, мають далі слугувати для опису та дослідження різних математичних теорій. Предметом останніх є множини певних елементів (предметів, індивідів тощо), наприклад, натуральні числа в арифметиці, дійсні числа в аналізі чи елементи конкретної групи в теорії груп. Деякі з цих елементів виділяються, оскільки відіграють особливу роль (як нейтральний елемент у групі або числа 0 і 1 в арифметиці). Окрім елементів можуть розглядатися також певні відношення між ними (як-от відношення порядку в арифметиці чи аналізі, або властивість "бути точками загального положення" в геометрії) і певні відображення та функції (напр., різні дії над числами — додавання, множення тощо). Зручно, щоб різним об'єктам — елементам множин, виділеним елементам, відношенням, функціям — відповідали букви різних типів. Окрім того, зручно ввести деякі допоміжні символи, які відіграватимуть роль розділових знаків звичайної мови. Нарешті, потрібні логічні символи — на роль сполучників звичайної мови. Оскільки далі часто використовуватиметься індукція за довжиною слів (а фактично — за кількістю логічних символів у слові), то зручно обмежитися мінімально необхідною кількістю різних логічних символів.

Отже, *алфавіт мови* (або *логіки*) *першого порядку* складається із символів таких шести груп.

1) Нескінченна множина символів *предметних* (або *індивідних*) *змінних*. Зазвичай позначатимемо їх малими латинськими буквами з кінця алфавіту (можливо, з індексами): $x, y, z, \ldots, x_1, y_1, z_1, \ldots$ і називатимемо просто *змінними*.

2) Деяка множина (можливо, порожня) символів *предметних* (або *індивідних*) *констант*. Зазвичай позначатимемо їх малими латинськими буквами з

початку алфавіту (можливо, з індексами): $a, b, \ldots, a_1, b_1, \ldots$ і називатимемо просто *константами*.

3) Непорожня скінченна або нескінченна множина *предикатних* символів. Позначатимемо їх великими латинськими буквами із середини алфавіту (можливо, з індексами). Кожному предикатному символу P ставиться у відповідність ціле число $\omega(P) \geq 0$, яке називають *арністю* цього символу, причому хоча б один предикатний символ мусить мати додатну арність. Якщо $\omega(P) = n$, то P називають *n-арним* предикатним символом (при $n = 1, 2$ або 3 говорять також про *унарні*, *бінарні* та *тернарні* символи). Якщо арність предикатного символу потрібно вказати явно, то її записують як верхній індекс (напр., запис P_3^2 означає, що предикатний символ P_3 має арність 2).

4) Деяка (можливо, порожня) множина *функціональних* символів. Позначатимемо їх малими латинськими буквами із середини алфавіту (можливо, з індексами). Кожному функціональному символу f ставиться у відповідність ціле додатне число $\omega(f)$, яке називають його *арністю*. Щодо арностей функціональних символів можна повторити всі ті зауваження, що стосувалися предикатних символів.

5) Символи $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, $\oplus$, $\neg$ *логічних операцій* (або *логічних зв'язок*) і два *квантори*: (квантор *загальності*) $\forall$ і (квантор *існування*) $\exists$.

6) *Службові* (допоміжні) символи: *ліва дужка* "(", *права дужка* ")" і *кома* ",".

Зауваження. 1. *Зліченний алфавіт (а в переважній більшості змістовних прикладів він є саме таким) формально легко можна замінити скінченним. Для цього досить занумерувати символи і далі використовувати лише їхні номери, записані у певній системі числення. Більше того, на практиці майже завжди так і роблять. Однак "ієрогліфи" вигляду x_{1341}, P_{13}^2 чи f_{179}^{12} зручно розглядати все-таки як окремі символи (відповідно змінної, предикатний і функціональний), а не як слова у відповідних алфавітах. Наприклад, тоді, коли використовують індукцію за складністю формули.*

2. *Інколи предметні константи розглядають як функціональні символи арності 0 і не виділяють в окрему групу символів.*

3. *У мовах, пристосованих до конкретних математичних теорій, для предикатних і функціональних символів і констант часто використовують традиційні позначення (напр., 0 і 1 — для відповідних констант, $<$ і $=$ — для предикатних символів, $+$ і $\cdot$ — для функціональних тощо).*

4. *Предикатні символи арності* 0 *називають ще логічними константами, гіпотезами або висловлюваннями*[29].

[29] У конкретних математичних теоріях вони зустрічаються нечасто і зазвичай саме у вигляді гіпотез: континуум-гіпотеза, гіпотеза Рімана тощо.

Перша й шоста групи символів в усіх мовах першого порядку однакові. П'ята група може варіюватися, але в не дуже широких межах: набір логічних зв'язок має бути повним. Зате інші три групи можуть варіюватися дуже сильно.

Пару $\Omega = (\Omega_1, \Omega_2)$, де Ω_1 — множина предикатних, а Ω_2 — множина констант і функціональних символів мови L, називають *сигнатурою* цієї мови.

Довільну скінченну послідовність символів алфавіту називають *словом* у цьому алфавіті.

Приклад. *Слово*

$$(x + y = z) \wedge (\sin y > 3) \to \forall x(y \cdot z > 0) \tag{3.3}$$

належить мові, яка містить предметні константи 3 *і* 0*, предикати арності* 2 *"=" і ">", функціональний символ* "$\sin$" *арності* 1 *і функціональні символи "+" та "·" арності* 2.

Сукупність правил, які визначають граматику мови, тобто вказують, які слова і тексти (= послідовності слів) мови є 'правильними', утворює *синтаксис* мови. Розглянемо ці правила для мови першого порядку:

Термами називають слова мови L, для яких використовують такі правила:

(1) кожна змінна і кожна константа є термом;
(2) для кожного функціонального символу f арності n і термів $t_1, \ldots, t_n$ слово $f(t_1, \ldots, t_n)$ також є термом;
(3) інших термів нема (тобто слово мови L буде термом тоді й лише тоді, коли його можна побудувати, застосовуючи скінченну кількість разів правила (1) і (2)).

Множину всіх термів мови L позначатимемо $\mathcal{T}(L)$.

Якщо P — предикатний символ арності n і $t_1, \ldots, t_n$ — терми, то слово $P(t_1, \ldots, t_n)$ називають *атомарною* формулою. Зокрема, атомарними формулами будуть предикатні символи арності 0.

Серед усіх слів мови L найважливішими для нас будуть так звані *правильно побудовані формули* (скорочено *ппф*). Вони визначаються так:

(1) кожна атомарна формула є ппф;
(2) для довільних ппф A і B вирази $(\neg A)$, $(A \vee b)$, $(A \wedge B)$, $(A \to B)$, $(A \leftrightarrow B)$ і $(A \oplus B)$ також є ппф;

(3) для довільних ппф A та змінної x вирази $\big(\forall x A(x)\big)$ і $\big(\exists x A(x)\big)$ також є ппф;

(4) інших правильно побудованих формул нема (тобто слово мови L буде ппф тоді й лише тоді, коли його можна побудувати, застосовуючи скінченну кількість разів правил (1)—(3)).

Множину всіх правильно побудованих формул мови L позначатимемо $\mathcal{F}(L)$.

Зауваження. *Дужки у формулах відіграють суто службову роль і потрібні насамперед для того, щоб можна було однозначно відновити послідовність застосування правил* (1)—(3). *Друга їхня функція — полегшити сприйняття формул. Однак, коли дужок забагато, то друга функція не виконується. Тому зазвичай для полегшення сприйняття частину дужок у формулі вилучають. Зокрема, вилучаються зовнішні дужки. Правила, які саме дужки відкидати, можна формалізувати, але простіше в цих речах керуватися здоровим глуздом.*

Ще одна вольність стосується однойменних кванторів: замість $\forall\, x_1 \forall\, x_2 \ldots \forall\, x_k$ часто пишуть $\forall\, x_1, x_2, \ldots, x_k$, а замість $\exists\, x_1 \exists\, x_2 \ldots \exists\, x_k$ — $\exists\, x_1, x_2, \ldots, x_k$.

Крім того у випадку предикатних і функціональних символів арності 2 відповідний символ часто пишуть не перед аргументами, а між ними. Наприклад, як у слові (3.3), $x > y$ замість $>(x, y)$ або $x + y$ замість $+(x, y)$.

Підформулою формули A називається довільне підслово формули A, яке саме є ппф.

Під *областю дії квантора* розуміється підформула, яка починається з лівої дужки, що стоїть перед даним квантором, і закінчується відповідною їй правою дужкою.

Входження змінної x у формулу A називають *пов'язаним*, якщо ця змінна входить в область дії деякого квантора із змінною x (тобто, якщо воно є входженням у підформулу вигляду $(\mathsf{K}x\, B)$ де K — квантор). У протилежному разі входження змінної x називають *вільним*.

Пов'язані кванторами змінні ще називають *фіктивними*. Вони лише вказують, як саме побудоване висловлювання $\forall x A(x)$ або $\exists x A(x)$. Зокрема, замість пов'язаних змінних не можна підставляти конкретні значення чи назви конкретних об'єктів. Зате, з невеликими застереженнями, пов'язану змінну можна замінити довільною іншою буквою або і взагалі обійтися без неї. Але використання таких змінних є звичним і зручним для міркувань, а тому досить поширеним у математиці.

У ширшому розумінні під *квантором* розуміють операцію, яка застосовується до так званих підкванторних виразів, що містять *змінні, які пов'язуються* даним квантором. У результаті отримують вираз, який від цих змінних уже не залежить, але зміст якого залежить від сукупності значень підкванторного виразу на області, яку пробігають змінні квантора. Зокрема, окрім "офіційних" кванторів $\forall$ і $\exists$, у математиці широко використовують квантори $\sum\limits_i \ldots$, $\prod\limits_i \ldots$, $\int_a^b \ldots dx$ тощо.

Змінна є вільною (пов'язаною) для формули F, якщо хоча б одне з входжень цієї змінної у формулу F є вільним (пов'язаним). Зауважимо, що та сама змінна може бути у формулі F як вільною, так і пов'язаною.

Формули, які не містять вільних входжень змінних, називають *замкненими формулами* або *твердженнями*.

Якщо $x_1, \ldots, x_n$ — список усіх вільних змінних формули F, то формулу $\forall x_1 \ldots \forall x_n\, F$ називають *замиканням загальності* формули F.

Відомо, як часто використовуються в математиці підстановки замість змінних певних виразів (частковим випадком є заміни змінних). Використовуються вони і в логіці. Але треба зробити кілька застережень.

У формулах логіки предикатів щось підставляти можна лише замість вільних входжень змінних. Зрозуміло, що це "щось" має бути термом. Але дозволяються не всі підстановки термів. Терм t називають *вільним для змінної* x у формулі F, якщо жодне вільне входження змінної x не лежить в області дії квантора за якою-небудь змінною, що зустрічається в t. Підставляти в дану формулу замість даної змінної можна лише ті терми, які вільні для цієї змінної.

Приклад. *Терм* $t_1 = z + 2$ *є вільним для змінної* y *у формулі* $\exists\, x\, (y > x)$, *а терм* $t_2 = z + x$ *таким не є.*

Зауваження. *Подібні нюанси з підстановками замість змінних певних виразів виникають і в інших розділах математики. Наприклад, підстановка в рівність* $g(y) = \int_a^b f(x, y)dx$ *замість* y *змінної* z *є цілком законною і приводить до рівності* $g(z) = \int_a^b f(x, z)dx$, *рівносильної початковій. У той же час підстановка замість* y *змінної* x *є некоректною, бо приводить до рівності* $g(x) = \int_a^b f(x, x)dx$, *яка з початкової жодним чином не випливає.*

Той факт, що змінні $x_1, \ldots, x_n$ є вільними у формулі F, ми позначатимемо як $F(x_1, \ldots, x_n)$, а результат заміни змінних $x_1, \ldots, x_n$ термами $t_1, \ldots, t_n$ позначатимемо як $F(t_1, \ldots, t_n)$.

3.3. Семантика формул

Сама по собі формула логіки предикатів жодного змісту не має — це просто побудована за певними правилами послідовність символів із певного алфавіту. Щоб надати формулі зміст, її треба *інтерпретувати* — тобто сказати, який саме конкретний зміст мають усі функціональні і предикатні символи та константи, що в ній зустрічаються. Тому під *інтерпретацією* φ мови L першого порядку ми розумітимемо таке.

Фіксуємо непорожню множину M і

— кожному константному символу c ставимо у відповідність деякий елемент $\varphi(c)$ множини M;

— кожному предикатному символу $P \in \Omega_1$ арності n ставимо у відповідність n-арний предикат $\varphi(P) : M^n \to \{0, 1\}$ на M;

— кожному функціональному символу $f \in \Omega_2$ арності n ставимо у відповідність n-арну функцію $\varphi(f) : M^n \to M$ на M.

У випадках, коли інтерпретація відома або байдуже, про яку саме інтерпретацію йдеться, символи мови L і їхні образи в інтерпретації зазвичай позначають однаково і говорять про *елемент* c, *функцію* f і *предикат* P.

Якщо φ — інтерпретація мови L на множині M, то пару $\mathfrak{M} = (M, \varphi)$ називатимемо *алгебричною системою* для мови L. Множину M називають *основною множиною* або *носієм* алгебричної системи.

Для даної інтерпретації φ мови L на множині M кожне відображення $\mu : V \to M$ множини V предметних змінних мови L в M називають *деталізацією* інтерпретації φ. Пару $\langle \varphi, \mu \rangle$ називають *оцінкою* мови L.

У результаті оцінки кожен терм мови L стає іменем певного елемента з множини M(тобто набуває певного значення з M), а кожна формула набуває певного значення істинності. Значення $\varphi_\mu(t)$ від терма t в оцінці $\langle \varphi, \mu \rangle$ визначають індуктивно:

(1) якщо $t = c$, де c — символ константи, то $\varphi_\mu(t) = \varphi(c)$;

(2) якщо $t = x$, де x — символ предметної змінної, то $\varphi_\mu(t) = \mu(x)$;

(3) якщо $t = f(t_1, \ldots, t_k)$, де f — функціональний символ арності k, то $\varphi_\mu(t) = \varphi(f)(\varphi_\mu(t_1), \ldots, \varphi_\mu(t_k))$.

Значення $\varphi_\mu(F)$ формули F також визначають індуктивно:

(1) якщо $F = P(t_1, \ldots, t_k)$ — атомарна формула, то $\varphi_\mu(F) = \varphi(P)(\varphi_\mu(t_1), \ldots, \varphi_\mu(t_k))$;

(2) якщо $F = \neg G$, то $\varphi_\mu(F) = \neg\varphi_\mu(G)$;

(3) якщо $F = G_1 \circ G_2$, де $\circ$ — одна з бінарних логічних зв'язок, то $\varphi_\mu(F) = \varphi_\mu(G_1) \circ \varphi_\mu(G_2)$;

(4) якщо $F = \forall x\, G$, то $\varphi_\mu(F) = 1$ тоді й лише тоді, коли для кожної деталізації μ', яка відрізняється від μ щонайбільше значенням змінної x (тобто $\mu|_{V\setminus\{x\}} = \mu'|_{V\setminus\{x\}}$), справедлива рівність $\varphi_{\mu'}(G) = 1$;

(5) якщо $F = \exists x\, G$, то $\varphi_\mu(F) = 1$ тоді й лише тоді, коли для деякої деталізації μ', яка відрізняється від μ щонайбільше значенням змінної x, справедлива рівність $\varphi_{\mu'}(G) = 1$.

Таким чином, оцінку мови будують у два етапи. На першому відбувається інтерпретація на деякій множині M нелогічних (предикатних і функціональних) символів мови (тобто завдяки тому, що предикатні та функціональні символи мови набувають змісту конкретних предикатів і функцій на M, на M задається певна алгебрична структура). На другому етапі відбувається фіксація значень змінних. Після цього кожна формула набуває певного значення істинності з множини $\{0, 1\}$.

Зауважимо, що оцінка замкненої формули залежить лише від інтерпретації (тобто для всіх деталізацій даної інтерпретації значення істинності замкненої формули буде однаковим).

Якщо формула F набуває у даній інтерпретації (оцінці) логічного значення 1, то говоримо, що у цій інтерпретації (оцінці) формула F *стає істинною*. Якщо розглядають лише інтерпретацію, без подальшої її деталізації, то говорячи про істинність чи хибність формули F у даній інтерпретації завжди маємо на увазі замикання загальності формули F.

Аналогічно слід розуміти вираз "формула F *стає хибною* у даній інтерпретації (оцінці)".

Тепер уже зрозуміло, навіщо ми виділили в логіці предикатів два типи виразів — терми і формули — і яка між ними відмінність. Терми служать для позначення об'єктів (конкретних або залежних від певних параметрів — змінних), а формули виражають зв'язки між цими об'єктами.

Вправа 3.2. *Чи є вільною або пов'язаною змінною символ x у формулі $(x^2)' = 2x$?*

Більшість тверджень, які зустрічаються як у математиці, так і поза її межами, можна записати у вигляді формул логіки предикатів, певним чином підібравши відповідні предикати. Формалізуючи твердження певної математичної теорії (напр., арифметики або геометрії) ми вибираємо за атомарні (тобто базові) лише деякі з предикатів, що в ній зустрічаються. Навіть у межах

даної теорії цей вибір можна робити багатьма способами. Вибір конкретних атомарних предикатів часто диктується потребами конкретної задачі[30].

Розглянемо чотири приклади запису тверджень засобами, що належать до логіки предикатів.

Приклади. 1. *На множині людей задано такі предикати:* $J(x) =$ *"*x *— юрист",* $R(x) =$ *"*x *— шахрай",* $P(x) =$ *"*x *— політик". Тоді за допомогою цих предикатів твердження "кожен політик — якщо не шахрай, то юрист" можна записати так:*

$$\forall\, x(P(x) \to (\neg R(x) \to J(x)).$$

Або так:

$$\forall\, x((P(x) \wedge \neg R(x)) \to J(x)).$$

2. *На множині натуральних чисел* $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ *задано предикати* $S(x, y, z)$ *(який дорівнює 1 тоді й лише тоді, коли* $x + y = z$*) і* $D(x, y, z)$ *(який дорівнює 1 тоді й лише тоді, коли* $x \cdot y = z$*). Запишемо за допомогою цих предикатів твердження "множина простих чисел є нескінченною".*

Почнемо з того, що переформулюємо наше твердження в дусі давніх греків: "для кожного простого числа існує більше за нього просте число". Уже видно, що нам знадобляться допоміжні предикати $P(x)$ = *"*x *— просте число" і "*$x > y$*". Оскільки поняття простого числа формулюється в термінах подільності, то нам буде потрібен і предикат "*$x \mid y$*", який легко виразити через* D*:* $x \mid y = \exists\, z D(x, z, y)$*. Розглянемо ще два унарні предикати "*$x = 0$*" =* $S(x, x, x)$ *та "*$x = 1$*" =* $\forall\, y D(x, y, y)$*. Тепер ми можемо виразити предикати "*$x > y$*" та* $P(x)$ *через базисні:*

$$x > y = \exists\, z(z \neq 0 \wedge S(z, y, x)),$$
$$P(x) = x > 1 \wedge \forall\, y(y \mid x \to (y = 1 \vee y = x)).$$

Після цієї підготовчої роботи вже можна записати і твердження "множина простих чисел є нескінченною":

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

Перекладаючи з природної мови на формальну, потрібно враховувати не лише *структуру* даного твердження, а і його *зміст* і навіть *контекст*.

[30] У розвинених математичних теоріях зазвичай не обмежуються лише атомарними предикатами, а вводять нові, які виражають через атомарні за допомогою певних рівностей (тобто є скороченими позначеннями певних складних предикатів).

3. *"Усі спадні і зростаючі функції є монотонними".*
Формалізація, яка напрошується:

$$\forall\, f\, (\textit{Спад}\,(f) \wedge \textit{Зрост}\,(f) \to \textit{Мон}\,(f)),$$

є неправильною, бо умову "Спад $(f)\wedge$*Зрост* (f)*" задовольняють лише функції-константи. Правильною формалізацією буде*

$$\forall f(\textit{Спад}\,(f) \vee \textit{Зрост}\,(f) \to \textit{Мон}\,(f)).$$

Отже, говорилося "і", а перекладати треба "або".

4. *Розглянемо два твердження:*
(а) "Усі студенти мехмату поважають один одного" і
(б) "Усі команди ліги змагаються одна з одною",
які мають тотожну граматичну форму.

Правильна формалізація твердження (а) має вигляд

$$\forall x, y\, (\textit{студ.мехмат}\,(x) \wedge \textit{студ.мехмат}\,(y) \to \textit{поваж}\,(x, y) \wedge \textit{поваж}\,(y, x)). \tag{3.4}$$

Можна й коротше:

$$\forall x, y\, (\textit{студ.мехмат}\,(x) \wedge \textit{студ.мехмат}\,(y) \to \textit{поваж}\,(x, y)).$$

А для тотожного за формою твердження (б) правильна формалізація має вигляд

$$\forall x\, (\textit{команда}\,(x) \to \exists y\, (\textit{команда}\,(y) \wedge \textit{змаг}\,(x, y) \wedge \textit{змаг}\,(y, x))). \tag{3.5}$$

Крім усього, на відміну від попереднього прикладу, вилучити $\textit{змаг}\,(y, x)$ *тут не можна.*

Формулу F називають *виконливою*, якщо існує оцінка, за якої ця формула стає істинною.

Замкнену формулу F називають *загальнозначущою*, якщо вона буде істинною за всіх інтерпретацій. Загальнозначущі формули відіграють у логіці предикатів роль, подібну до ролі тавтологій у логіці висловлювань.

Інколи буває корисним варіант поняття загальнозначущості, пов'язаний із потужністю області інтерпретації: формулу F називають $\mathfrak{m}$-*загальнозначущою*, якщо вона буде істинною у всіх інтерпретаціях на множинах потужності

$\mathfrak{m}$. Формулу F називають *скінченно загальнозначущою*, якщо вона буде n-загальнозначущою для всіх натуральних чисел n (тобто у всіх інтерпретаціях на скінченних множинах).

Зрозуміло, що кожна загальнозначуща формула буде і скінченно загальнозначущою. Але зворотне твердження не є правильним.

Приклади. 1. *На двохелементній множині існує лише чотири різні предикати арності* 1. *Тому для довільних предикатів* $P_1, \ldots, P_5$ *арності* 1 *наступна формула*

$$\forall x\,(P_1(x) \leftrightarrow P_2(x)) \vee \forall x\,(P_1(x) \leftrightarrow P_3(x)) \vee \cdots$$
$$\cdots \vee \forall x\,(P_1(x) \leftrightarrow P_5(x)) \vee \forall x\,(P_2(x) \leftrightarrow$$
$$\leftrightarrow P_3(x)) \vee \cdots \vee \forall x\,(P_4(x) \leftrightarrow P_5(x))$$

буде 2*-загальнозначущою.*

2. *Формула*

$$\big(\forall x\forall y\forall z(x \le y \wedge y \le z \to x \le z) \wedge \forall x\forall y(x \le y \wedge y \le x \to x = y) \wedge$$
$$\wedge\forall x(x \le x) \wedge \forall x\forall y(x \le y \vee y \le x)\big) \to \exists y\forall x(x \le y)$$

стверджує, що в лінійно впорядкованій множині є найбільший елемент. Це твердження виконується в кожній скінченній лінійно впорядкованій множині, але не виконується, наприклад, у множині $\mathbb{N}$ *із звичайним відношенням порядку. Тому ця формула є скінченно загальнозначущою, але не є загальнозначущою.*

Уже говорилось, що загальнозначущі формули є аналогами тавтологій логіки висловлювань. Однак, якщо для перевірки формул логіки висловлювань на тавтологічність є нехай і громіздкі, але ефективні алгоритми (напр., побудова таблиць істинності), то для перевірки на загальнозначущість таких алгоритмів нема. Причому нема не тому, що їх досі ніхто не знайшов, а тому, що їх взагалі не існує (якщо алфавіт мови включає хоча б один предикатний символ арності ≥ 2). Це складає зміст відомої *теореми Черча про нерозв'язність числення предикатів*. Історично це був перший приклад доведення нерозв'язності змістовної алгоритмічної проблеми.

Оскільки єдиного алгоритму не існує, то доводити загальнозначущість формул можна лише за допомогою змістовних міркувань. З іншого боку, для спростування загальнозначущості досить одного контрприкладу.

Перевірка формули на загальнозначущість тісно пов'язана з перевіркою на виконливість: формула буде загальнозначущою тоді й лише тоді, коли її заперечення не буде виконливою формулою.

Нехай F — замкнена формула мови L. Алгебричну систему $\mathfrak{M}=(M,\varphi)$ називають *моделлю* формули F (позначають: $\mathfrak{M}_L \models_\varphi F$), якщо F стає істинною в інтерпретації φ. Якщо мова L відома або в даному контексті не є важливою, говоримо просто про модель $\mathfrak{M}$ і пишемо $\mathfrak{M} \models F$.

Якщо формула F — незамкнена, то моделлю F називають модель замикання загальності формули F.

Множину Δ формул називають *семантично несуперечливою* (або просто *несуперечливою*), якщо існує модель, яка є спільною для всіх формул із Δ. У протилежному разі множину Δ називають (*семантично*) *суперечливою*.

Зауваження. *Ми навішували квантори лише за змінними, значеннями яких були елементи певної множини. Логіку з такими кванторами називають логікою першого порядку і їй в математиці належить головна роль. Взагалі кажучи, можна розглядати і квантори за змінними, які пробігають сукупність усіх підмножин даної множини, сукупність усіх предикатів (або функцій) даної арності на даній множині, сукупність усіх множин предикатів (або функцій) тощо. Так з'являються логіки вищих порядків. Але вони, по-перше, влаштовані значно складніше, а по-друге, відіграють у математиці досить скромну роль. Тому ми обмежимося лише логікою першого порядку.*

Є думка, відома під назвою *тези Гільберта*, що коли всі позалогічні математичні припущення формулювати явно, то кожне математичне твердження можна виразити в логіці першого порядку. На користь цієї тези свідчить весь досвід розвитку математики, однак користуватися нею на практиці часто не дуже зручно. Наприклад, у формулюваннях теорем про властивості неперервних функцій природно з'являються квантори по функціях, а в теоремі Келі про те, що кожна група ізоморфна деякій групі підстановок, маємо як квантор загальності по сукупності абстрактних груп, так і квантор існування по сукупності груп підстановок.

Однак у логіці при дослідженні властивостей усієї сукупності математичних тверджень, їхньої структури, логічних зв'язків між ними тощо, обмежитися лише логікою першого порядку дуже зручно. Це дозволяє уникнути багатьох труднощів суто технічного характеру.

3.4. Рівносильні формули. Логічний наслідок

Кажуть, що формула F є *логічним* (або *семантичним*) *наслідком* із множини формул Δ (і пишуть $\Delta \models F$), якщо для кожної оцінки, при якій усі формули з Δ є істинними, формула F також є істинною. Іншими словами, якщо для якоїсь оцінки формула F є хибною, то при цій оцінці має бути хибною принаймні одна формула з Δ.

У цих термінах загальнозначущість формули F означає, що вона є логічним наслідком із порожньої множини формул (у цьому випадку замість $\varnothing \models F$ пишуть просто $\models F$). А суперечливість множини формул Δ означає, що логічним наслідком із Δ є тотожно хибна формула 0: $\Delta \models 0$.

Легко зрозуміти, що наслідок 1.2 залишається справедливим і для формул логіки предикатів.

Формули A і B називають *логічно рівносильними* (або *семантично рівносильними*, або просто *рівносильними*), якщо кожна з них є логічним наслідком з іншої. Тобто формули A і B будуть рівносильними, якщо при кожній оцінці ці формули мають однакові значення істинності. Звідси одразу випливає, що відношення рівносильності є відношенням еквівалентності. Факт рівносильності формул A і B позначають так: $A \equiv B$.

З означення рівносильності формул одразу випливає таке твердження.

Твердження 3.1. *Замкнені формули F_1 і F_2 будуть рівносильними тоді й лише тоді, коли формула $F_1 \leftrightarrow F_2$ буде загальнозначущою.*

Теорема 3.1 (про підстановку)**.** *Якщо $A(x_1, \ldots, x_n)$ і $B(x_1, \ldots, x_n)$ — рівносильні формули логіки висловлювань, то для довільних формул $F_1, \ldots, F_n$ логіки предикатів формули $A(F_1, \ldots, F_n)$ і $B(F_1, \ldots, F_n)$ також будуть рівносильними.*

Доведення. При кожній деталізації будь-якої інтерпретації формул логіки предикатів формули $F_1, \ldots, F_n$ набувають певних логічних значень $\varepsilon_1, \ldots, \varepsilon_n$. Але тоді з рівносильності $A(x_1, \ldots, x_n)$ і $B(x_1, \ldots, x_n)$ як формул логіки висловлювань випливає, що при кожній деталізації формули $A(F_1, \ldots, F_n)$ і $B(F_1, \ldots, F_n)$ набувають однакових логічних значень. □

З теореми 3.1 випливає, що майже все, що раніше говорилося про логічний наслідок і рівносильні формули у логіці висловлювань (зокрема, теорема 1.5, основні закони логіки висловлювань, твердження 1.1, 1.2, теорема 1.7 про основні типи логічних наслідків) лишається справедливим і для логіки предикатів. Аналогом відповідних рівносильностей для логічних наслідків є і закони де Моргана для кванторів.

Теорема 3.2 (закони де Моргана)**.**

$$\neg(\forall\, x\, \mathfrak{A}(x)) \equiv \exists\, x\, (\neg\mathfrak{A}(x)), \quad \neg(\exists\, x\, \mathfrak{A}(x)) \equiv \forall\, x\, (\neg\mathfrak{A}(x))$$

Доведення. Доведемо перший із законів. Нехай $y_1, \ldots, y_n$ — список усіх вільних змінних (для лівої і правої формул він той самий), а φ — довільна інтерпретація цих формул на деякій множині M. Припустимо, що ліва частина на наборі $(m_1, \ldots, m_n)$ набуває значення 1. Тоді формула $\forall\, x\, \mathfrak{A}(x)$ на цьому наборі набуває значення 0. Це означає, що предикат $\mathfrak{A}(x, m_1, \ldots, m_n)$ не є тотожно істинним. Отже, предикат $\neg\mathfrak{A}(x, m_1, \ldots, m_n)$ є виконливим, а тому права частина на наборі $(m_1, \ldots, m_n)$ також набуває значення 1.

Якщо ж ліва частина на наборі $(m_1, \ldots, m_n)$ набуває значення 0, то формула $\forall\, x\, \mathfrak{A}(x)$ на цьому наборі набуває значення 1. Тому предикат $\mathfrak{A}(x, m_1, \ldots, m_n)$ буде тотожно істинним, а його заперечення $\neg\mathfrak{A}(x, m_1, \ldots, m_n)$ — тотожно хибним. Але тоді права частина на наборі $(m_1, \ldots, m_n)$ також набуває значення 0.

Отже, ліва і права частини завжди набувають однакових значень істинності, а тому є рівносильними.

Другий закон доводять аналогічно. □

Але в логіці предикатів є й інші важливі рівносильності, які не мають аналогів у логіці висловлювань.

Теорема 3.3 (основні рівносильності логіки предикатів)**.** *Символом* K *позначається будь-який із кванторів* $\forall$ *або* $\exists$.

1) *Якщо змінна* x *не зустрічається у формулі* $\mathfrak{A}$*, то* $\mathsf{K}x\, \mathfrak{A} \equiv \mathfrak{A}$.

2) *Якщо змінна* z *не зустрічається у формулі* $\mathfrak{A}$*, то*
$\mathsf{K}x\, \mathfrak{A}(x) \equiv \mathsf{K}z\, \mathfrak{A}(z)$.

3) $\forall\, x \forall\, y\, \mathfrak{A}(x, y) \equiv \forall\, y \forall\, x\, \mathfrak{A}(x, y)$, $\exists\, x \exists\, y\, \mathfrak{A}(x, y) \equiv \exists\, y \exists\, x\, \mathfrak{A}(x, y)$.

4) $\forall\, x\, (\mathfrak{A}(x) \wedge \mathfrak{B}(x)) \equiv \forall\, x\, \mathfrak{A}(x) \wedge \forall\, x\, \mathfrak{B}(x)$,
$\exists\, x\, (\mathfrak{A}(x) \vee \mathfrak{B}(x)) \equiv \exists\, x\, \mathfrak{A}(x) \vee \exists\, x\, \mathfrak{B}(x)$.

5) $\mathsf{K}_1 x\, \mathfrak{A}(x) \vee \mathsf{K}_2 x\, \mathfrak{B}(x) \equiv \mathsf{K}_1 x\, \mathsf{K}_2 y\, (\mathfrak{A}(x) \vee \mathfrak{B}(y))$,
$\mathsf{K}_1 x\, \mathfrak{A}(x) \wedge \mathsf{K}_2 x\, \mathfrak{B}(x) \equiv \mathsf{K}_1 x\, \mathsf{K}_2 y\, (\mathfrak{A}(x) \wedge \mathfrak{B}(y))$.

6) *Якщо формула* $\mathfrak{B}$ *не містить вільних входжень змінної* x*, то*
$\mathsf{K}x\, \mathfrak{A}(x) \vee \mathfrak{B} \equiv \mathsf{K}x\, (\mathfrak{A}(x) \vee \mathfrak{B})$, $\mathsf{K}x\, \mathfrak{A}(x) \wedge \mathfrak{B} \equiv \mathsf{K}x\, (\mathfrak{A}(x) \wedge \mathfrak{B})$.

7) *Якщо формула* $\mathfrak{B}$ *не містить вільних входжень змінної* x*, то*
$\forall\, x\, \mathfrak{A}(x) \to \mathfrak{B} \equiv \exists\, x\, (\mathfrak{A}(x) \to \mathfrak{B})$, $\exists\, x\, \mathfrak{A}(x) \to \mathfrak{B} \equiv \forall\, x\, (\mathfrak{A}(x) \to \mathfrak{B})$.

8) *Якщо формула* $\mathfrak{A}$ *не містить вільних входжень змінної* x*, то*
$\mathfrak{A} \to \mathsf{K}x\, \mathfrak{B}(x) \equiv \mathsf{K}x\, (\mathfrak{A} \to \mathfrak{B}(x))$.

Доведення. Рівносильності 1) і 2) випливають безпосередньо з означення операцій навішування кванторів. Рівносильності 3) очевидні.

4) Доведемо першу з рівносильностей. Нехай $y_1, \ldots, y_n$ — список усіх вільних змінних (для лівої і правої формул він той самий), а φ — довільна інтерпретація цих формул на деякій множині M. Припустимо, що ліва частина на наборі $(m_1, \ldots, m_n)$ набуває значення 1. Це означає, що предикат $\mathfrak{A}(x, m_1, \ldots, m_n) \wedge \mathfrak{B}(x, m_1, \ldots, m_n)$ є тотожно істинним. З означення кон'юнкції предикатів випливає, що кожен із предикатів $\mathfrak{A}(x, m_1, \ldots, m_n)$ та $\mathfrak{B}(x, m_1, \ldots, m_n)$ також є тотожно істинним. Але тоді кожна з формул $\forall\, x\, \mathfrak{A}(x)$ і $\forall\, x\, \mathfrak{B}(x)$ набуває на наборі $(m_1, \ldots, m_n)$ значення 1. Тому кон'юнкція цих формул (тобто права частина) також набуває значення 1.

Аналогічно розглядають випадок, коли ліва частина на наборі $(m_1, \ldots, m_n)$ набуває значення 0.

Решту рівносильностей пропонуємо довести самостійно. □

Зауваження. *На відміну від однойменних кванторів, різнойменні квантори переставляти не можна. Наприклад, при стандартній інтерпретації символу "=" на неодноелементній множині формула* $\forall\, x\, \exists\, y\, \neg(x = y)$ *набуває значення* 1*, а формула* $\exists\, y\, \forall\, x\, \neg(x = y)$ *— значення* 0*.*

Означення 3.1. *Кажуть, що формула логіки предикатів записана у* ***випередженій нормальній формі*** *(або* ***пренексній нормальній формі****), якщо вона має вигляд*

$$\mathsf{K}_1 x_{i_1} \cdots \mathsf{K}_r x_{i_r} F(x_1, \ldots, x_n)\ ,$$

де $\mathsf{K}_1, \ldots, \mathsf{K}_r$ *— квантори, а формула* $F(x_1, \ldots, x_n)$ *містить лише операції* $\vee$*,* $\wedge$ *та* $\neg$*, причому знак заперечення* $\neg$ *може стояти лише перед предикатами. Зокрема, формула* $F(x_1, \ldots, x_n)$ *не містить кванторів.*

Теорема 3.4. *Кожна формула логіки предикатів рівносильна формулі, записаній у випередженій нормальній формі.*

Доведення. Спочатку, використовуючи рівносильності логіки висловлювань, виключаємо всі зв'язки, окрім $\neg, \vee, \wedge$ і кванторів. Далі, застосовуючи закони де Моргана (як для логіки висловлювань, так і з теореми 3.3, вилучаємо знак заперечення до предикатів. Наступним кроком, використовуючи рівносильність із теореми 3.3, п.2), робимо так, щоб різні квантори пов'язували різні змінні і щоб жодна змінна не була одночасно вільною і пов'язаною. Нарешті, використовуючи рівносильності (теорема 3.3, п.6)

$$\forall x A(x) \wedge B \equiv \forall x (A(x) \wedge B), \quad \exists x A(x) \wedge B \equiv \exists x (A(x) \wedge B),$$
$$\forall x A(x) \vee B \equiv \forall x (A(x) \vee B), \quad \exists x A(x) \vee B \equiv \exists x (A(x) \vee B),$$

виносимо всі квантори наперед і отримуємо випереджену нормальну форму. □

Зауваження. 1. *Природно виникає питання про єдиність випередженої нормальної форми для даної формули. Квантори можна виносити в різному порядку і використовуючи різні рівносильності, тому вигляд кванторної приставки (і навіть кількість кванторів у ній) залежатиме від способу зведення. Наприклад, формула* $\forall x A(x) \wedge \forall x B(x)$ *рівносильна як формулі* $\forall x(A(x) \wedge B(x))$*, так і формулі* $\forall x \forall y(A(x) \wedge B(y))$*. Тому єдина канонічна форма, подібна до досконалої* **НФ** *у логіці висловлювань, серед випереджених нормальних форм жодним природним чином не виділяється (навіть якщо обмежитися зв'язками* $\neg, \vee, \wedge$*). Навіть більше, з'ясування рівносильності двох формул, записаних у випередженій нормальній формі, є однією з важливих і важких задач логіки предикатів.*

2. *Якщо в початковій формулі крім кванторів зустрічаються лише зв'язки* $\neg, \vee, \wedge, \rightarrow$*, то з теореми 3.3 випливає, що до випередженої нормальної форми можна зводити таким чином, щоб відносний порядок зв'язок* $\neg, \vee, \wedge, \rightarrow$ *не порушувався.*

4. Формальні (аксіоматичні) теорії

Хоча термін “доведення” є чи не найголовнішим у математиці, ледве не її синонімом, він не має точного означення. В усій своїй повноті це поняття належить математиці не більше ніж психології, адже доведення — це міркування, яке переконує нас аж настільки, що ми готові за його допомогою переконувати інших. Взагалі кажучи, у математиків зазвичай не буває серйозних розходжень у питанні, вважати чи ні конкретний ланцюжок міркувань доведенням певного математичного факту. Однак такого інтуїтивного поняття доведення замало, коли ставиться під сумнів можливість самого існування доведення. Такі сумніви з’явилися, наприклад, у XIX ст. щодо V постулату після довгих і марних зусиль його довести.

Одним із шляхів (у логіці — основним) уточнення і формалізації поняття доведення є дослідження різних аксіоматичних теорій або, трохи ширше, числень. Основними компонентами аксіоматичної теорії є певна мова, набір аксіом і набір правил виводу. Точний опис цих компонентів дає змогу дати строге означення поняття доведення і, зокрема, одержати точні формулювання й обгрунтування неможливості доведення тих або інших тверджень.

4.1. Формальна теорія

У найширшому розумінні теорія — це певна мова L разом із деякою множиною T речень (формул) цієї мови. Мову L вибирають так, щоб структура її речень якоюсь мірою відображала їхній зміст. Тому, зазвичай, це певна штучна мова. З кожною мовою пов’язана множина її інтерпретацій, за яких речення (формули) мови стають істинними або хибними твердженнями.

Є два найпоширеніші шляхи виникнення теорій. За першого *семантичного* підходу до побудови теорії T вона складається з усіх формул мови L, істинних при будь-якій інтерпретації з даної фіксованої множини інтерпретацій. У цьому разі, зазвичай, створюються теорії, початково пов’язані із якоюсь однією конкретною структурою (напр., математичний аналіз або теорія функцій комплексної змінної).

Другий підхід є *синтаксичним*: теорія складається з усіх формул, які можуть бути виведені відповідно до заданих *правил синтаксичного слідування* з певної заданої множини *аксіом*. Такі теорії найчастіше пов’язані з цілим класом структур (напр., теорія груп або теорія векторних просторів).

У математиці ці два підходи дуже часто поєднують і використовують одночасно. Це дуже зручно. Однак, коли об’єктом дослідження стають самі теорії, то ці підходи слід чітко розрізняти. Тому далі зупинимося докладніше на другому з них.

Алфавітом називають фіксовану непорожню (скінченну або нескінченну) множину, елементи якої ми вміємо розрізняти. Елементи алфавіту називають *символами* або *буквами*. Скінченні послідовності букв із алфавіту A називають словами над A.

З технічних міркувань зручно ввести додатковий допоміжний символ, який називають *пробілом* або *порожньою буквою*.

У найзагальнішому вигляді *формальна теорія* (або *аксіоматична теорія*, або *числення*) складається з таких компонент:

— скінченний або зліченний алфавіт;

— серед усіх слів у даному алфавіті виділяється підмножина так званих *правильно побудованих формул*[31] (скорочено ппф);

— серед усіх ппф виділяється підмножина так званих *аксіом* ;

— скінченна кількість так званих *правил виводу*, які є скінченноарними предикатами на множині ппф.

Якщо правилом виводу є $(n+1)$-арний предикат P і $P(A_1, \ldots, A_n, A) = 1$, то кажуть, що формула A одержана з формул $A_1, \ldots, A_n$ за допомогою правила P, і часто записують це у вигляді

$$\frac{A_1, \ldots, A_n}{A}\,(P) \quad \text{або просто} \quad \frac{A_1, \ldots, A_n}{A}.$$

Нехай дана певна множина Δ формул (вона може бути й порожньою). Формули з Δ називатимемо *гіпотезами*. *Виводом* формули A з множини гіпотез Δ називають скінченну послідовність A_1, A_2, …, A_n формул, яка задовольняє такі дві умови:

1) кожна формула A_i є або аксіомою, або гіпотезою, або одержується з якихось попередніх формул за одним із правил виводу;

2) остання формула A_n збігається з A[32].

Якщо такий вивід існує, то пишуть $\Delta \vdash A$ і кажуть, що A *виводиться* з Δ або що A є *видідною* з Δ. Якщо треба підкреслити, що йдеться саме про теорію L, то пишуть $\Delta \vdash_L A$. Число n називають довжиною виводу. Зі скінченності виводу випливає, що в кожному виводі може використовуватися лише скінченна кількість гіпотез.

Формулу A називають *теоремою*, якщо вона виводиться з порожньої множини гіпотез. Те, що A є теоремою, позначають $\vdash A$.

[31] Більшість слів у даному алфавіті, як, до речі, і в будь-якій живій мові, позбавлені змісту. Тому ппф — це такі слова, яким у певній інтерпретації символів алфавіту можна надати зміст.

[32] Зауважимо, що довільний початок A_1, A_2, …, A_k виводу A_1, A_2, …, A_n сам є виводом формули A_k.

Зауваження. 1. *Від множини ппф і від множини аксіом зазвичай вимагають, щоб вони були розв'язними. Тобто повинні існувати ефективні процедури перевірки, чи є дане слово ппф і чи є дана ппф аксіомою*[33].

2. *Від правил виводу вимагають, щоб вони були ефективно обчислювальними. Тобто для кожного правила виводу має існувати ефективна процедура перевірки, чи задовольняє даний набір формул даному правилу.*

3. *Правила виводу часто називають "логікою" даної теорії. Ця "логіка" може не мати жодного стосунку до звичайної логіки (див. приклад 2). Але далі розглядатимемо лише теорії, в яких роль "логіки" виконують звичайні правила логіки.*

Зрозуміло, що коли множина аксіом і правила виводу є розв'язними, то можна ефективно перевірити, чи є дана послідовність формул виводом. Але властивість бути теоремою у загальному випадку вже не є розв'язною: якщо ми не знайшли виводу конкретної формули, то це ще не означає, що такого виводу взагалі не існує. Таким чином, "формальність" формальної теорії T полягає в існуванні деякої процедури для перевірки правильності міркувань, яку можна застосовувати механічно. Зауважимо, що йдеться лише про процедуру *перевірки* доведень, а не про процедуру їхнього знаходження!

Приклади. 1. *Теорія $\mathcal{T}$ має алфавіт $S = \{a, b\}$, одну аксіому a і два правила виводу:* $\dfrac{A}{aAa}$ *і* $\dfrac{A}{Ab}$. *За допомогою індукції неважко довести (зробіть це!), що теоремами цієї теорії будуть усі такі і лише такі слова в алфавіті $\{a, b\}$, які містять непарну кількість букв a, причому для кожного входження букви b кількість букв a ліворуч від неї має бути на непарне число більшою від кількості букв a праворуч.*

2. *Шахи можна розглядати як формальну теорію: є алфавіт для запису шахових позицій, одна аксіома — початкова шахова позиція, і правила виводу — детально виписані в "Шаховому кодексі" правила ходів, за якими можна від однієї позиції переходити до іншої. Теоремами будуть ті позиції, які хоча б у принципі можуть виникнути під час гри.*

Зауваження. *Будувати виводи теорем даної формальної теорії L — це задача власне теорії L. Задача ж логіки — це вивчення самої L. Твердження про властивості L називають метатеоремами. Далі, у процесі вивчення числення висловлювань (як прикладу формальної теорії), нас цікавитиме*

[33] Строгі означення понять "ефективна процедура" і "розв'язна множина" наведено у наступному розділі.

зв'язок цього числення з логікою висловлювань. Метатеоремою буде, зокрема, твердження про те, що множина теорем числення висловлювань збігається з множиною тавтологій (так звана теорема про повноту числення висловлювань).

4.2. Числення висловлювань

Ми детально розглянемо числення висловлювань (коротко: ЧВ) з двох міркувань. По-перше, це цікавий і порівняно простий приклад аксіоматичної теорії. Зокрема, числення висловлювань є однією з небагатьох змістовних теорій, для яких властивість бути теоремою є розв'язною. По-друге, воно входить як складова частина в інші, складніші і змістовніші теорії.

Нині є багато різних версій числення висловлювань. Вони розрізняються наборами аксіом, правилами виводу і навіть алфавітами (для полегшення математичного дослідження такого числення часто зручно обмежитись невеликим повним набором логічних зв'язок, розглядаючи інші зв'язки як певні скорочення). Одну з таких версій ми нижче і розглянемо.

a) Алфавіт ЧВ містить нескінченну множину символів змінних, два логічні символи 0 і $\to$ і два службові символи — ліву (і праву) дужки.

b) Правильно побудовані формули (ппф) визначають так:
(1) кожна змінна і 0 є ппф;
(2) якщо $\mathcal{A}$ і $\mathcal{B}$ — ппф, то $(\mathcal{A} \to \mathcal{B})$ — також ппф;
(3) ппф є тільки те, що можна одержати (можливо, багатократним) застосуванням правил (1) і (2).

Інші логічні зв'язки вводять як скорочення:

$$\neg\mathcal{A} := \mathcal{A} \to 0, \;\; 1 = \neg 0 := 0 \to 0, \tag{4.1}$$

$$\mathcal{A} \vee \mathcal{B} := \neg\mathcal{A} \to \mathcal{B}, \;\; \mathcal{A} \wedge \mathcal{B} := \neg(\mathcal{A} \to \neg\mathcal{B}), \tag{4.2}$$

$$\mathcal{A} \leftrightarrow \mathcal{B} := (\mathcal{A} \to \mathcal{B}) \wedge (\mathcal{B} \to \mathcal{A}), \;\; \mathcal{A} \oplus \mathcal{B} := \neg(\mathcal{A} \leftrightarrow \mathcal{B}). \tag{4.3}$$

c) ЧВ містить три *схеми* аксіом:
(1) $\mathcal{A} \to (\mathcal{B} \to \mathcal{A})$;
(2) $(\mathcal{A} \to (\mathcal{B} \to \mathcal{C})) \to ((\mathcal{A} \to \mathcal{B}) \to (\mathcal{A} \to \mathcal{C}))$;
(3) $((\mathcal{A} \to 0) \to 0) \to \mathcal{A}$.
Кожна схема аксіом дає нескінченну сім'ю аксіом після підстановки у відповідну схему замість $\mathcal{A}$, $\mathcal{B}$ і $\mathcal{C}$ конкретних ппф.

Зауважимо, що на підставі скорочення $\neg\mathcal{A} := \mathcal{A} \to 0$ схему аксіом (3) можна переписати у вигляді $\neg\neg\mathcal{A} \to \mathcal{A}$.

d) Єдиним правилом виводу ЧВ є правило *modus ponens* (MP):

$$\frac{\mathcal{A}, \mathcal{A} \to \mathcal{B}}{\mathcal{B}} \ .$$

Легко бачити, що множина ппф і множина аксіом ЧВ є нескінченними, але розв'язними. А правильність застосування правила modus ponens завжди можна перевірити.

Теорема 4.1 (про виправданість аксіоматизації)**.** *Кожна теорема числення висловлювань є тавтологією.*

Доведення. Безпосередньо перевіряють, що аксіоми є тавтологіями. Крім того, з тотожної істинності формул $\mathcal{A} \to \mathcal{B}$ і $\mathcal{A}$ випливає тотожна істинність формули $\mathcal{B}$. Тому індукція за довжиною виводу доводить теорему. □

З теореми про виправданість аксіоматизації одразу випливає *несуперечливість* числення висловлювань, тобто неможливість того, щоб для якоїсь формули A як сама формула A, так і її заперечення $\neg A$ були теоремами. Адже із формул A і $\neg A$ щонайбільше одна може бути тавтологією.

Доведення наступної леми дає одночасно перші приклади виводів у численні висловлювань.

Лема 4.1. $a) \vdash A \to A;\ b) \vdash 0 \to A.$

Доведення. $a)$ 1. $(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))$; (A.2)
2. $A \to ((A \to A) \to A)$; (A.1)
3. $(A \to (A \to A)) \to (A \to A)$; (MP до 1 і 2)
4. $A \to (A \to A)$; (A.1)
5. $A \to A$; (MP до 3 і 4)

$b)$ 1. $((A \to 0) \to 0) \to A$; (A.3)
2. $(((A \to 0) \to 0) \to A) \to (0 \to (((A \to 0) \to 0) \to A))$; (A.1)
3. $0 \to (((A \to 0) \to 0) \to A)$; (MP до 1 і 2)
4. $(0 \to (((A \to 0) \to 0) \to A)) \to ((0 \to ((A \to 0) \to 0)) \to (0 \to A))$; (A.2)
5. $(0 \to ((A \to 0) \to 0)) \to (0 \to A)$. (MP до 3 і 4)
6. $0 \to ((A \to 0) \to 0)$; (A.1)
7. $0 \to A$. (MP до 5 і 6) □

Імплікацію $\to$ свого часу ми визначили таким чином, що твердження $\models A \to B$ і $A \models B$ виявилося рівносильним. Далі хочемо одержати дедуктивний аналог цього факту — довести рівносильність тверджень $\vdash A \to B$ і $A \vdash B$.

Теорема 4.2 (теорема дедукції)**.** *$A \vdash B$ тоді й лише тоді, коли $\vdash A \to B$.*

Доведемо цю теорему навіть у загальнішому вигляді.

Теорема 4.3 (теорема дедукції)**.** *Для довільної множини формул Δ $\Delta, A \vdash B$ тоді й лише тоді, коли $\Delta \vdash A \to B$.*

Доведення. Достатність умови очевидна: якщо $A_1, \ldots, A_n = A \to B$ — вивід формули $A \to B$ із Δ, то послідовність $A_1, \ldots, A_n = A \to B, A, B$ буде виводом формули B із множини гіпотез Δ, A.

Необхідність. Покажемо індукцією за довжиною виводу

$$A_1, A_2, \ldots, A_n = B \tag{4.4}$$

формули B із множини гіпотез Δ, A як можна цей вивід переробити у вивід

$$\ldots, A \to A_1, \ldots, A \to A_2, \ldots, A \to A_n = A \to B \tag{4.5}$$

формули $A \to B$ із множини гіпотез Δ.

Формула A_1 може бути або аксіомою, або гіпотезою, або формулою A. У перших двох випадках маємо вивід

$$A_1, \;\; A_1 \to (A \to A_1), \;\; A \to A_1.$$

Для випадку $A_1 = A$ вивід формули $A \to A_1$ є в доведенні леми 4.1.

Припустимо тепер, що ми вже побудували частину

$$\ldots, A \to A_1, \ldots, A \to A_2, \ldots, A \to A_k \tag{4.6}$$

виводу (4.5). Для формули A_{k+1} у виводі (4.4) можливі 4 можливості: вона є або аксіомою, або гіпотезою, або формулою A, або одержується з попередніх формул A_i та $A_j = A_i \to A_{k+1}$ $(i, j \le k)$ за правилом MP. У трьох перших випадках міркуємо як і раніше. У четвертому випадку для побудови чергового фрагмента виводу (4.5) використаємо те, що вивід (4.6) уже містить формули $A \to (A_i \to A_{k+1})$ (*) та $A \to A_i$ (**):

1. $(A \to (A_i \to A_{k+1})) \to ((A \to A_i) \to (A \to A_{k+1}))$; (A.2)
2. $(A \to A_i) \to (A \to A_{k+1})$; (MP до (*) і 1)
3. $A \to A_{k+1}$. (MP до (**) і 2) □

Вправа 4.1. *Доведіть, що* $\mathcal{A} \to (\mathcal{B} \to \mathcal{C}) \vdash \mathcal{B} \to (\mathcal{A} \to \mathcal{C})$.

Зауваження. 1. *Теорема дедукції є першим серйозним кроком до доведення адекватності вибраної аксіоматизації алгебри висловлювань.*

2. *Неявно теорему дедукції для доведення математичних фактів використовують дуже часто (у вигляді так званого методу введення припущень). У найпростішій формі це виконується, коли твердження формулюється у формі* $\vdash A \to B$ *("Якщо ..., то ..."), а доведення має форму* $A \vdash B$ *("Припустимо, що .... Тоді ...").*

3. *Вивід формули* $A \to B$*, який будують за допомогою алгоритму з доведення теореми дедукції, зазвичай виявляється занадто громіздким. Зокрема, він часто містить лишні фрагменти, які можна вилучити. Але це є ціною, яку доводиться платити за загальність та ефективність методу.*

4. *Доведення теореми дедукції використовує лише перші дві схеми аксіом.*

Як приклад ефективного використання теореми дедукції розглянемо доведення наступної леми.

Лема 4.2 (правило силогізму)**.**

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

Доведення. За теоремою дедукції

$$A \to B,\ B \to C \vdash A \to C \Leftrightarrow A \to B,\ B \to C, A \vdash C.$$

Для правої частини будуємо вивід:

1. $A \to B$;
2. A;
3. B
4. $B \to C$;
5. C. □

Твердження 4.1 (про транзитивність поняття вивідності)**.** *Якщо кожна формула* $A \in \Omega$ *виводиться з множини формул* Δ *і* $\Omega \vdash B$*, то* $\Delta \vdash B$*.*

Доведення. Нехай $A_1, \ldots, A_n$ — усі ті формули із Ω, які зустрічаються у виводі B із Ω. Тоді послідовність формул

$$\underbrace{\ldots, A_1}_{\text{вивід } A_1},\ \underbrace{\ldots, A_2}_{\text{вивід } A_2},\ \ldots,\ \underbrace{\ldots, A_n}_{\text{вивід } A_n},\ \underbrace{\ldots, B}_{\text{вивід } B}$$

є виводом B із Δ. □

Побудова виводу безпосередньо з аксіом часто є досить важкою і громіздкою задачею. Тому використовують різні допоміжні засоби. Окрім згаданої вище теореми дедукції до таких засобів належать так звані *похідні правила виводу*: кожному співвідношенню $\mathcal{A}_1, \ldots, \mathcal{A}_n \vdash \mathcal{B}$ відповідає похідне правило виводу

$$\frac{\mathcal{A}_1, \ldots, \mathcal{A}_n}{\mathcal{B}}.$$

Застосовуючи для виводу даного твердження похідні правила виводу, ми насправді одержуємо не вивід цього твердження, а лише строге доведення існування такого виводу. Однак посилання на похідні правила виводу, як і посилання на теорему дедукції, завжди можна суто механічно замінити відповідними виводами.

Кілька важливих похідних правил виводу, які згодом нам знадобляться, дає наступна лема.

Лема 4.3. *a)* $A \vdash \neg\neg A$*;*
b) $\neg\neg A \vdash A$;
c) $A, \neg B \vdash \neg(A \to B)$;
d) $A \to B, \neg A \to B \vdash B$;
e) $\neg A \to \neg B \vdash B \to A$;
f) $B \to A \vdash \neg A \to \neg B$;
g) $A \to \neg B \vdash B \to \neg A$;
h) $A, \neg A \vdash B$.

Доведення. a) Враховуючи скорочення $\neg\mathcal{A} = \mathcal{A} \to 0$, наше співвідношення можна переписати у вигляді $A \vdash (A \to 0) \to 0$. Згідно з теоремою дедукції, друге співвідношення рівносильне такому: $A, A \to 0 \vdash 0$. А для нього вивід зовсім простий: 1. $A \to 0$; 2. A; 3. 0.

b) Випливає з теореми дедукції і схеми аксіом (3).

c) Враховуючи скорочення $\neg\mathcal{A} = \mathcal{A} \to 0$ і теорему дедукції, маємо рівносильне співвідношення $A, B \to 0, A \to B \vdash 0$, для якого маємо простий і короткий вивід: 1. A; 2. $A \to B$; 3. B; 4. $B \to 0$; 5. 0.

d) Враховуючи b) і правило силогізму, досить показати, що $A \to B, \neg A \to B \vdash \neg\neg B$. Подібно як в a), це співвідношення можна замінити рівносильним $A \to B, (A \to 0) \to B, B \to 0 \vdash 0$. Тут вивід трохи складніший:

1. $A \to B$ (гіпотеза);
2. $B \to 0$ (гіпотеза);
3. $A \to 0$ (правило силогізму до 1 і 2);
4. $(A \to 0) \to B$ (гіпотеза);
5. B (MP до 3 і 4);
6. 0 (MP до 2 і 5).

e) Враховуючи скорочення $\neg\mathcal{A} = \mathcal{A} \to 0$ і теорему дедукції, маємо рівносильне співвідношення $(A \to 0) \to (B \to 0), B \vdash A$. Для нього одержуємо такий вивід:

1. $((A \to 0) \to (B \to 0)) \to (((A \to 0) \to B) \to ((A \to 0) \to 0))$; (A.2)
2. $(A \to 0) \to (B \to 0)$; (гіпотеза)
3. $((A \to 0) \to B) \to ((A \to 0) \to 0)$; (MP до 1 і 2)
4. $B \to ((A \to 0) \to B)$; (A.1)
5. B; (гіпотеза)
6. $(A \to 0) \to B$; (MP до 4 і 5)
7. $(A \to 0) \to 0$; (MP до 3 і 6)
8. $((A \to 0) \to 0) \to A$; (A.3)
9. A. (MP до 7 і 8)

f) Із теореми дедукції випливає, що досить довести рівносильне співвідношення $B \to A, \neg A \vdash \neg B$. Враховуючи a) і b), із $\neg\neg B \to B, B \to A, A \to \neg\neg A$ за правилом силогізму отримуємо $\neg\neg B \to \neg\neg A$. Звідси на підставі e) отримуємо $\neg A \to \neg B$. Нарешті, застосовуючи до останньої формули і гіпотези $\neg A$ правило MP, отримуємо $\neg B$.

g) Пропонуємо довести самостійно.

h) Враховуючи a) і b), досить довести, що $A, \neg A \vdash \neg\neg B$, що за теоремою дедукції рівносильне співвідношенню $A \vdash \neg A \to \neg\neg B$. Із e) і f) випливає, що останнє співвідношення рівносильне співвідношенню $A \vdash \neg B \to A$, яке у свою чергу рівносильне очевидному співвідношенню $A, \neg B \vdash A$. □

Уважний читач зауважив, що насправді в жодному з пунктів доведення леми 4.3 ми не будували виводу формули F із співвідношення $\ldots \vdash F$. Ми лише за допомогою теореми дедукції доводили існування такого виводу. Але доведення теореми дедукції є конструктивним і тепер при бажанні можна легко виписати потрібні виводи.

Нагадаємо, що для довільних $\alpha \in \{0, 1\}$ та змінної x ми позначали через x^α змінну x, якщо $\alpha = 1$, і формулу $\neg x$, якщо $\alpha = 0$. Це позначення можна узагальнити на довільні формули: якщо $x_1, \ldots, x_n$ — список усіх змінних, які зустрічаються у формулі F, то для довільного булевого вектора $(\alpha_1, \ldots, \alpha_n)$ покладемо

$$F^{(\alpha_1,\ldots,\alpha_n)} := \begin{cases} F, & \text{якщо } F(\alpha_1, \ldots, \alpha_n) = 1; \\ \neg F, & \text{якщо } F(\alpha_1, \ldots, \alpha_n) = 0. \end{cases}$$

Зручно також для довільного булевого вектора $(\alpha_1, \ldots, \alpha_n)$ покласти

$$0^{(\alpha_1,\ldots,\alpha_n)} := 1.$$

Лема 4.4 (Кальмара)**.** *Якщо усі змінні, від яких залежить формула F, зустрічаються у списку $x_1, \ldots, x_n$, то для довільного булевого вектора $\alpha = (\alpha_1, \ldots, \alpha_n)$*

$$x_1^{\alpha_1}, \ldots, x_n^{\alpha_n} \vdash F^{(\alpha_1,\ldots,\alpha_n)}.$$

Доведення. Застосуємо індукцію за довжиною формули F. Тоді замість $x_1^{\alpha_1}, \ldots, x_n^{\alpha_n}$ писатимемо x^α, а замість $F^{(\alpha_1,\ldots,\alpha_n)}$ — F^α.

Для формул довжини 0 (символів змінних і константи 0) наше твердження випливає з леми 4.1.

Розглянемо спочатку випадок, коли формула F має вигляд $F = \neg A$, причому для формули A твердження леми вже доведено. Нехай $F(\alpha) = 1$. Тоді $A(\alpha) = 0$ і $x^\alpha \vdash \neg A$. Але $\neg A = F = F^\alpha$. Якщо ж $F(\alpha) = 0$, то $A(\alpha) = 1$ і $x^\alpha \vdash A$. Але $A \vdash \neg\neg A$ і $\neg\neg A = \neg F = F^\alpha$. Тому $x^\alpha \vdash F^\alpha$.

Нехай тепер формула F має вигляд $F = A \to B$, причому для формул A і B твердження леми вже доведено. Розглянемо три можливі випадки:

1) $A(\alpha) = 0$. Тоді $(A \to B)(\alpha) = 1$ і $(A \to B)^\alpha = A \to B$. За припущенням індукції $x^\alpha \vdash \neg A$, а за лемою 4.3, п.h) $\neg A \vdash A \to B$. Тому $x^\alpha \vdash A \to B$.

2) $A(\alpha) = 1$, $B(\alpha) = 0$. Тоді $(A \to B)(\alpha) = 0$ і $(A \to B)^\alpha = \neg(A \to B)$. За припущенням індукції $x^\alpha \vdash A$ і $x^\alpha \vdash \neg B$. Крім того за лемою 4.3, п.c) $A, \neg B \vdash \neg(A \to B)$. Тому $x^\alpha \vdash \neg(A \to B)$.

3) $A(\alpha) = 1$, $B(\alpha) = 1$. У цьому випадку $(A \to B)(\alpha) = 1$ і $(A \to B)^\alpha = A \to B$. За припущенням індукції $x^\alpha \vdash B$, а за схемою аксіом (1) $B \vdash A \to B$. Тому $x^\alpha \vdash A \to B$. □

4.2.1. Теореми числення висловлювань

Використовуючи лему Кальмара, теорему дедукції і лему 4.3, п.d), неважко довести основну теорему про числення висловлювань.

Теорема 4.4 (теорема про повноту для числення висловлювань)**.**
$\vdash F$ *тоді й лише тоді, коли* $\models F$.

Доведення. Імплікацію $\vdash F \Rightarrow \models F$ вже доведено (теорема 4.1 про виправданість аксіоматизації). Лишилося довести зворотну імплікацію.

Нехай формула F є тавтологією, а всі змінні, від яких залежить F, зустрічаються у списку $x_1, \ldots, x_n$. Тоді для кожного булевого вектора $\alpha = (\alpha_1, \ldots, \alpha_n)$ $F(\alpha) = 1$ і $F^\alpha = F$. За лемою Кальмара

$$x_1^{\alpha_1}, \ldots, x_{n-1}^{\alpha_{n-1}}, x_n^{\alpha_n} \vdash F. \tag{4.7}$$

Зокрема,

$$x_1^{\alpha_1}, \ldots, x_{n-1}^{\alpha_{n-1}}, x_n \vdash F,$$
$$x_1^{\alpha_1}, \ldots, x_{n-1}^{\alpha_{n-1}}, \neg x_n \vdash F.$$

За теоремою дедукції

$$x_1^{\alpha_1}, \ldots, x_{n-1}^{\alpha_{n-1}} \vdash x_n \to F,$$
$$x_1^{\alpha_1}, \ldots, x_{n-1}^{\alpha_{n-1}} \vdash \neg x_n \mapsto F.$$

За лемою 4.3, п.d) $x_n \to F, \neg x_n \to F \vdash F$. Тому

$$x_1^{\alpha_1}, \ldots, x_{n-1}^{\alpha_{n-1}} \vdash F. \tag{4.8}$$

Але набір $(\alpha_1, \ldots, \alpha_{n-1})$ — довільний. Тому, повторюючи попередні міркування, ми можемо з лівої частини (4.7) виключити $x_{n-1}^{\alpha_{n-1}}$ і т. д. Зрештою, після n подібних кроків, отримаємо $\vdash F$. □

Зауважимо, що доведення теореми про повноту ЧВ є конструктивним у тому сенсі, що воно дає алгоритм, за допомогою якого для довільної тавтології за її таблицею істинності можна побудувати її вивід.

Наслідок 4.1. *Формули A і B рівносильні тоді й лише тоді, коли $A \vdash B$ і $B \vdash A$.*

Доведення. Рівносильність A і B означає, що формули $A \to B$ і $B \to A$ є тавтологіями, отже, теоремами ЧВ. Але за теоремою дедукції співвідношення $\vdash A \to B$ і $A \vdash B$ еквівалентні. □

Оскільки для кожної формули A тавтологією може бути щонайбільше одна з формул A і $\neg A$, то з теореми повноти одразу випливає *несуперечливість* числення висловлювань: *для жодної формули A в численні висловлювань не можна вивести як A, так і її заперечення $\neg A$* (тобто не буває такого, щоб обидві формули A і $\neg A$ були теоремами числення висловлювань).

Вправа 4.2. *Доведіть, що*
a) $A_1, \ldots, A_n \vdash B \Leftrightarrow A_1, \ldots, A_n \models B$;
b) $A \equiv B$ *тоді й лише тоді, коли* $A \vdash B$ *і* $B \vdash A$.

Теорема 4.5 (про непоповнювальність ЧВ)**.** *Якщо ЧВ поповнити схемою аксіом, яка задається формулою, що не є тавтологією, то нове числення буде суперечливим.*

Доведення. Нехай формула $A = A(a_1, \ldots, a_n)$ не є тавтологією. Тоді існує набір $(\varepsilon_1, \ldots, \varepsilon_n)$ значень змінних, на якому ця формула набуває значення 0. Підставляючи тепер у формулу A замість змінної a_i формулу $0 \vee \neg 0$, якщо $\varepsilon_i = 1$, і формулу 0, якщо $\varepsilon_i = 0$, одержимо формулу B, яка буде теоремою розширеної теорії.

З другого боку, формула B є тотожно хибною. Але тоді $\neg B$ є тавтологією, а тому також є теоремою розширеної теорії. Отже, в розширеній теорії обидві формули — B і $\neg B$ — є теоремами, а тому вона суперечлива. □

Множину формул Δ називають *локально несуперечливою*, якщо кожна її скінченна підмножина є несуперечливою.

Теорема 4.6 (теорема компактності для логіки висловлювань)**.** *Множина формул Δ несуперечлива тоді й лише тоді, коли вона локально несуперечлива.*

У термінах моделей теорему компактності можна переформулювати так.

Теорема 4.7 (теорема компактності для логіки висловлювань)**.** *Множина формул Δ має модель тоді й лише тоді, коли кожна її скінченна підмножина має модель.*

Для доведення теореми 4.7 нам знадобляться деякі поняття й факти з теорії частково впорядкованих множин (вони знадобляться і пізніше при доведенні теореми 4.15).

Нехай на множині M є відношення часткового порядку $\leq$. Підмножину $A \subseteq M$ називають *ланцюгом*, якщо вона лінійно впорядкована (тобто для довільних елементів a і b з A буде $a \leq b$ або $b \leq a$). Підмножину $A \subseteq M$ називають *обмеженою згори*, якщо існує такий елемент $c \in M$, що $a \leq c$ для всіх $a \in A$. Елемент $c \in M$ називають *максимальним*, якщо в M не існує елемента, більшого за c.

Лема 4.5 (Цорна)**.** *Якщо в частково впорядкованій множині M кожен ланцюг обмежений згори, то в M для кожного елемента a існує такий максимальний елемент c, що $a \leq c$.*

У різних розділах математики нині відомо близько двох десятків тверджень, рівносильних лемі Цорна (серед них і знаменита аксіома вибору). Якщо одне із цих тверджень взяти за аксіому, інші стають теоремами. В алгебрі прийнято за аксіому брати саме лему Цорна. Ми теж так зробимо.

Доведення теореми 4.7 *Необхідність* умови очевидна.

Достатність. Позначимо через M множину всіх локально несуперечливих множин формул. M є природно впорядкованою за включенням. Оскільки

об'єднання кожного зростаючого ланцюга локально несуперечливих множин знову є локально несуперечливою множиною, то частково впорядкована множина $(M, \subset)$ задовольняє умову леми Цорна. А тому кожна локально несуперечлива множина формул міститься в деякій максимальній за включенням локально несуперечливій множині.

Лема 4.6. *Якщо Δ — максимальна за включенням локально несуперечлива множина, то для кожної формули F або $F \in \Delta$, або $\neg F \in \Delta$.*

Доведення. Припустимо, що Δ — максимальна локально несуперечлива множина, але жодна з формул F і $\neg F$ не належить Δ. Тоді існують такі скінченні підмножини $\Delta_1 \subseteq \Delta$ і $\Delta_2 \subseteq \Delta$, що кожна з множин $\Delta_1 \cup \{F\}$ і $\Delta_2 \cup \{\neg F\}$ є суперечливою. У такому разі $\Delta_1 \models \neg F$ і $\Delta_2 \models F$. Але тоді скінченна множина $(\Delta_1 \cup \Delta_2) \subseteq \Delta$ буде суперечливою, що суперечить умові леми. □

Вважаємо тепер, що Δ — максимальна локально несуперечлива множина. Для кожного простого висловлювання s покладемо $\varphi(s) = 1$, якщо $s \in \Delta$, і $\varphi(s) = 0$, якщо $s \notin \Delta$. З леми 4.6 випливає, що φ є коректно визначеною інтерпретацією. Індукцією за довжиною формул покажемо, що ця інтерпретація є моделлю для $F \in \Delta$. Для формул довжини 0, тобто простих висловлювань, це випливає з означення інтерпретації φ.

Розглянемо спочатку випадок, коли формула $F \in \Delta$ має вигляд $F = \neg A$, причому для формули A твердження вже доведено. Тоді за лемою 4.6 $A \notin \Delta$ і за припущенням індукції $\varphi(A) = 0$. Тому $\varphi(F) = 1$. Навпаки, якщо $\varphi(F) = 1$, то $\varphi(A) = 0$ і за припущенням індукції $A \notin \Delta$. Але тоді за лемою 4.6 $F \in \Delta$.

Далі нехай формула F має вигляд $F = A \to B$, причому для формул A і B твердження вже доведено. Якщо $\varphi(F) = 0$, то $\varphi(A) = 1$, $\varphi(B) = 0$. За припущенням індукції $A \in \Delta$ і $B \notin \Delta$. Але тоді $\neg B \in \Delta$. Оскільки набір формул $A \to B$, A, $\neg B$ є суперечливим, то $F \notin \Delta$. Якщо ж $\varphi(F) = 1$, то або $\varphi(A) = 0$, або $\varphi(A) = 1, \varphi(B) = 1$. У першому випадку $A \notin \Delta$, тому $\neg A \in \Delta$. Із суперечливості набору $\neg(A \to B)$, $\neg A$ випливає, що $F \in \Delta$. У другому випадку $A \in \Delta$, $B \in \Delta$ і включення $F \in \Delta$ випливає із суперечливості набору $\neg(A \to B)$, A, B.

Отже, φ є моделлю для Δ.

Нехай тепер Δ' — довільна локально несуперечлива множина. Вона міститься в якійсь максимальній локально несуперечливій множині $\Delta \supseteq \Delta'$. Зрозуміло, що кожна модель для Δ буде моделлю і для Δ', а існування моделі для Δ вже доведено. □

Зауваження. 1. *Якщо множина простих висловлювань скінченна або зліченна, то максимальну за включенням локально несуперечливу множину,*

яка містить дану локально несуперечливу множину Δ і для кожної формули F містить або F, або $\neg F$, можна побудувати, не використовуючи лему Цорна. Справді, у цьому випадку множина всіх формул є зліченною і ми можемо їх занумерувати: F_1, F_2, …, F_1, …, причому можемо це зробити конструктивно. Визначимо тепер ланцюг

$$\Delta_0 \subseteq \Delta_1 \subseteq \Delta_2 \subseteq \cdots \subseteq \Delta_n \subseteq \cdots$$

таким чином: $\Delta_0 = \Delta$;

$$\Delta_{n+1} = \begin{cases} \Delta_n \cup \{F_{n+1}\}, & \text{якщо ця множина локально несуперечлива;} \\ \Delta_n \cup \{\neg F_{n+1}\}, & \text{у протилежному разі.} \end{cases}$$

Покладемо $\Delta' = \bigcup_i \Delta_i$. Тоді $\Delta \subseteq \Delta'$ і для кожної формули F або $F \in \Delta'$, або $\neg F \in \Delta'$. Щоб для побудови моделі для Δ використати міркування з попереднього доведення, лишилося показати, що множина Δ' — локально несуперечлива. Для цього досить показати, що локально несуперечливою є кожна з множин Δ_n. Це легко зробити індукцією за номером n. Справді, $\Delta_0 = \Delta$ локально несуперечлива за умовою. Якщо $\Delta_{n+1} = \Delta_n \cup \{F_{n+1}\}$, то Δ_{n+1} локально несуперечлива за побудовою. Нехай тепер $\Delta_{n+1} = \Delta_n \cup \{\neg F_{n+1}\}$. Це можливе лише у випадку, коли існує така скінченна підмножина $\Theta_1 \subseteq \Delta_n$, що множина $\Theta_1 \cup \{F_{n+1}\}$ є суперечливою. Припустимо, що Δ_{n+1} — локально суперечлива. Тоді існує така скінченна підмножина $\Theta_2 \subseteq \Delta_n$, що множина $\Theta_2 \cup \{F_{n+1}\}$ є суперечливою. Скінченна множина $\Theta_1 \cup \Theta_2 \subseteq \Delta_n$ повинна мати модель. Але в цій моделі одна з формул F або $\neg F$ повинна бути істинною, що суперечить суперечливості кожної з множин $\Theta_1 \cup \{F_{n+1}\}$ і $\Theta_2 \cup \{F_{n+1}\}$. Отже, і в цьому випадку множина Δ_{n+1} є локально несуперечливою.

2. *Для теореми* 4.6 *можна дати суто топологічне доведення, з якого стане зрозуміло, чому вона так називається. Нехай S — множина всіх простих висловлювань, а $B = \{0, 1\}$ — двоелементний топологічний простір із дискретною топологією. Розглянемо простір $I = \prod_{s \in S} B$ з топологією добутку. За відомою з топології теоремою Тихонова він буде компактним як добуток компактних просторів. Елементи множини I можна розглядати як інтерпретації логіки висловлювань. Кожній формулі F поставимо у відповідність множину $\widehat{F}$ тих інтерпретацій з I, за яких F буде істинною. Формула F містить лише скінченну кількість простих висловлювань, тому множина $\widehat{F}$ буде відкритою. Оскільки $I \setminus \widehat{F} = \widehat{\neg F}$, то множина $\widehat{F}$ буде також замкненою. Для локально несуперечливої множини Δ формул розглянемо сім'ю множин $T = \{\widehat{F} \mid F \in \Delta\}$. Оскільки кожна скінченна підмножина з Δ має модель, то кожна скінченна підсім'я із T має непорожній*

перетин. Як відомо, топологічний простір буде компактним тоді й лише тоді, коли з того, що кожна скінченна підсім'я сім'ї M замкнених підпросторів має непорожній перетин, випливає, що і вся сім'я M має непорожній перетин. Тому з компактності I випливає, що вся сім'я множин T також має непорожній перетин. Кожен елемент цього перетину буде моделлю для Δ.

З теореми 4.4 і теореми компактності 4.6 легко одержати теорему повноти у загальнішому вигляді.

Теорема 4.8 (теорема про повноту для числення висловлювань)**.**
$\Delta \vdash F$ *тоді й лише тоді, коли* $\Delta \models F$.

Доведення. Достатність умови очевидна.

Необхідність. Нехай $\Delta \models F$. Тоді множина $\Delta \cup \{\neg F\}$ — суперечлива. За теоремою 4.6 існує така скінченна множина $\{B_1, \ldots, B_n\} \subseteq \Delta$, що множина $\{B_1, \ldots, B_n, \neg F\}$ суперечлива, тобто $B_1, \ldots, B_n, \neg F \models 0$. Але тоді

$$\models B_1 \to (\ldots \to (B_n \to (\neg F \to 0)) \ldots)$$

і за теоремою 4.4

$$\vdash B_1 \to (\ldots \to (B_n \to (\neg F \to 0)) \ldots) .$$

Звідси за теоремою дедукції $B_1, \ldots, B_n \vdash \neg F \to 0$. У свою чергу маємо $\neg F \to 0 \vdash F$, бо $(\neg F \to 0) \to F$ — тавтологія і $\vdash (\neg F \to 0) \to F$. Отже, $B_1, \ldots, B_n \vdash F$, а тому $\Delta \vdash F$. □

4.2.2. Приклади застосування теореми компактності*

Далеко не кожна властивість скінченних підмножин чи підструктур даного математичного об'єкта переноситься на сам об'єкт.

Приклади. 1. *Якщо кожна скінченна підмножина елементів даної групи породжує комутативну групу, то і вся група буде комутативною. Однак, якщо кожна скінченна підмножина даної групи породжує циклічну групу, то вся група не зобов'язана бути циклічною (як, напр., група* $\mathbb{Q}$*).*

2. *Для довільної множини* $M \subseteq \mathbb{R}$ *кожна скінченна підмножина з* M *є обмеженою. Однак сама* M *не зобов'язана бути обмеженою.*

Зауваження. *В алгебрі теореми, які стверджують, що коли кожна породжена скінченною множиною елементів підсистема даної алгебричної системи (напівгрупи, групи, кільця тощо) має дану властивість, то і вся*

система має цю властивість, називають локальними. Свого часу багато математиків займалося пошуком і доведенням таких теорем, причому доведення зазвичай були досить складними. Та після того, як у 30-х рр. минулого століття в математичній логіці було розроблено загальний підхід до отримання таких теорем, їхній потік в алгебрі майже припинився. Однак у тих розділах дискретної математики, де системи задають не аксіоматично, пошук таких теорем триває. Хоча і в таких випадках застосування, наприклад, теореми компактності для логіки висловлювань дозволяє отримати прості доведення теорем, початкові доведення яких були складними[34]. *Два такі приклади ми розглянемо.*

Кажуть, що граф можна розфарбувати в k кольорів, якщо кожній вершині графа можна приписати один із даних k кольорів так, щоб будь-які суміжні вершини мали різний колір.

Теорема 4.9 (де Брейна — Ердеша)**.** *Якщо вершини кожного скінченного підграфа даного графа* Γ *можна розфарбувати в* k *кольорів, то й вершини самого графа* Γ *можна розфарбувати в* k *кольорів.*

Доведення. Нехай V — множина вершин графа Γ, E — множина його ребер, $I = \{1, 2, \ldots, k\}$ — множина кольорів. Для кожної пари $(v, i) \in V \times I$ розглянемо просте висловлювання x_i^v, яке змістовно означає, що "вершина v пофарбована в колір i". Після цього для кожної вершини $v \in V$ напишемо формулу

$$F_v = (x_1^v \wedge \neg x_2^v \wedge \cdots \wedge \neg x_k^v) \vee \cdots \vee (\neg x_1^v \wedge \cdots \wedge \neg x_{k-1}^v \wedge x_k^v)\,,$$

яка означає, що "вершина v пофарбована в якийсь колір, причому тільки один", а для кожної пари $\{u, v\} \in E$ сусідніх вершин — формулу

$$G_{\{u,v\}} = \neg(x_1^u \wedge x_1^v) \wedge \neg(x_2^u \wedge x_2^v) \wedge \cdots \wedge \neg(x_k^u \wedge x_k^v),$$

яка означає, що "сусідні вершини u і v пофарбовані в різні кольори".

За умовою кожна скінченна підмножина множини формул

$$\Delta = \{F_v \mid v \in V\} \cup \{G_{\{u,v\}} \mid \{u, v\} \in E\}$$

має модель. Тоді має модель і вся множина Δ. Але модель для Δ є нічим іншим, як розфарбуванням графа Γ в k кольорів[35]. $\square$

[34] Значно ширші застосування має аналогічна теорема компактності для логіки предикатів, про яку говоритимемо трохи пізніше.

[35] Оригінальне доведення цієї теореми займає 5 сторінок.

Нагадаємо, що шириною частково впорядкованої множини M називають максимальну потужність антиланцюга (= множина попарно непорівняльних елементів) з M, а ланцюгом називають лінійно впорядковану множину (тобто та, яка має ширину 1).

Теорема 4.10 (нескінченна версія теореми Ділуорса)**.** *Кожну частково впорядковану множину M ширини k можна розбити на об'єднання k ланцюгів.*

Доведення. Для кожних елемента $a \in M$ і номера i розглянемо просте висловлювання x_a^i, яке змістовно означає, що "елемент a попадає в i-й ланцюг", а для кожної пари $\{a, b\}$ елементів — просте висловлювання $x_{\{a,b\}}$, яке змістовно означає, що "елементи a і b порівнянні". Після цього для кожного елемента $a \in M$ напишемо формулу

$$F_a = (x_1^a \wedge \neg x_2^a \wedge \cdots \wedge \neg x_k^a) \vee \cdots \vee (\neg x_1^a \wedge \cdots \wedge \neg x_{k-1}^a \wedge x_k^a)\,,$$

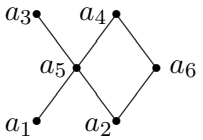
яка означає, що "елемент a попадає в якийсь ланцюг, причому тільки один", а для кожних номера i та пари $\{a, b\}$ порівняльних елементів — формулу

$$G^i_{\{a,b\}} = (x_a^i \wedge x_b^i) \to x_{\{a,b\}}\,,$$

яка означає, що "коли елементи a і b попадають в i-й ланцюг, то вони порівняльні".

Позначимо через Δ сукупність усіх формул F_a і $G^i_{\{a,b\}}$. За теоремою Ділуорса кожна скінченна підмножина з Δ має модель. Тоді має модель і вся множина Δ. Але модель для Δ є нічим іншим, як розбиттям множини M на об'єднання k ланцюгів. □

Зауваження. *Будувати розфарбування графа в k кольорів чи розбиття частково впорядкованої множини ширини k на k ланцюгів за допомогою індукції шляхом долучення на кожному кроці до побудованого розфарбування нової вершини чи до побудованого розбиття нового елемента (щоб потім перейти до об'єднання всіх побудованих розфарбувань чи розбиттів), взагалі кажучи, не вдається навіть для скінченних множин. Для прикладу розглянемо* 6*-елементну частково впорядковану множину M ширини* 2 *із діаграмою Гассе*:



Якщо 5*-елементна частково впорядкована множина* $\{a_1, a_2, a_3, a_4, a_5\}$ *була розбита на* 2 *ланцюги* $a_1 < a_3$ *і* $a_2 < a_5 < a_4$*, то вставити новий елемент* a_6 *у ці ланцюги не можна. Але розбиття* M *на два ланцюги існує:* $a_1 < a_5 < a_3$ *і* $a_2 < a_6 < a_4$.

4.3. Числення предикатів

Предметом числення предикатів є теорія виводу, що базується на структурі тверджень, в яких використовуються логічні зв'язки, предикати і квантори. У цьому сенсі воно є розширенням числення висловлювань. Ми обмежимось лише таким численням предикатів, у якому квантори дозволено застосовувати лише до предметних змінних. Щоб відрізнити таке числення предикатів від інших (де квантори можуть застосовуватися до натуральних чисел, до підмножин, до предикатів тощо), його називають *вузьким численням предикатів* або *численням предикатів першого порядку*.

Подібно тому як для побудови числення висловлювань ми обмежилися лише двома логічними зв'язками, а інші зв'язки вводили пізніше як скорочення для певних формул, для побудови числення предикатів (скорочено ЧП) ми також обмежимося лише квантором загальності $\forall$. Тому алфавіт ЧП містить логічні символи $0, \to, \forall$, допоміжні символи "(", ")", "," нескінченну множину символів змінних, деяку (можливо, порожню) множину предметних символів (індивідних констант), непорожню множину предикатних символів і (можливо, порожню) множину функціональних символів. Сукупність предметних, предикатних і функціональних символів називають *сигнатурою* ЧП. Якщо множина функціональних символів порожня, говорять про *чисте* (або *вузьке*) числення предикатів.

Інші логічні зв'язки вводять відповідно до правил (4.1)–(4.3), а квантор існування $\exists$ — як скорочення

$$\exists x \mathcal{A} := \neg \forall x \neg \mathcal{A}. \tag{4.9}$$

Оскільки, з одного боку, для ЧВ доведено теорему повноти, а з іншого боку, є ефективна процедура перевірки формули на тавтологічність, буде зручно *всі тавтології* ЧВ включити до *схем аксіом* ЧП (питання незалежності системи аксіом нас зараз не цікавить). Тому аксіоматика для ЧП буде така:

тавтології ЧВ;

(A4) $\forall x \mathcal{A}(x) \to \mathcal{A}(t)$, де t — терм[36];

(A5) $\forall x \big(\mathcal{A} \to \mathcal{B}(x)\big) \to \big(\mathcal{A} \to \forall x \mathcal{B}(x)\big)$.

[36] Схема аксіом (A4) фактично дозволяє переходити від *загального* до *часткового*. Тому аксіоми, отримані за цією схемою, називають *аксіомами конкретизації*.

Однак у схемах (A4) і (A5) є природні обмеження. У схемі (A4) термом t замінюють усі вільні входження змінної x у формулу $\mathcal{A}(x)$, причому після такої заміни кожна змінна у кожному входженні терма t повинна залишатися вільною (тобто забороняється так звана *колізія*). Кажуть ще, що терм t має бути *вільним* для змінної x у формулі $\mathcal{A}(x)$.

Типовий приклад колізії: якщо взяти $\mathcal{A} = \exists y\ (y \neq x)$ і $t = y$, то схема (A4) дає хибне на довільній неодноелементній множині твердження:

$$\forall x\ \exists y\ (y \neq x)\ \rightarrow\ \exists y\ (y \neq y).$$

У схемі (A5) формула $\mathcal{A}$ не повинна містити вільних входжень змінної x.

Крім аксіом, є ще два *правила виводу*:

(MP) $\dfrac{\mathcal{A}, \mathcal{A} \rightarrow \mathcal{B}}{\mathcal{B}}$ (правило *modus ponens*);

(Gen) $\dfrac{\mathcal{A}(x)}{\forall y \mathcal{A}(y)}$ (правило *узагальнення*).

В останньому випадку кажуть, що правило Gen застосовується до змінної x. Крім того, щоб не виникала колізія, y не має бути вільною змінною в $\mathcal{A}(x)$.

Виводом формули A з множини формул Δ називають скінченну послідовність $F_1, F_2, \ldots, F_n$ формул, яка закінчується формулою A і кожна формула якої є або аксіомою, або гіпотезою (тобто належить множині Δ), або одержується з якихось *попередніх* формул за одним із правил виводу. На застосування правила Gen є обмеження: його можна застосовувати лише до тих змінних x, які не входять вільно у жодну з тих формул із Δ, що безпосередньо використовуються в тій частині виводу, яка передує використанню правила. Але коли множина Δ є порожньою або містить лише *замкнені* формули, то жодних обмежень на використання правила Gen нема.

Формули з Δ називають *гіпотезами* або *засновками*, а формулу A — *висновком*. Факт існування виводу формули A з множини формул Δ позначають так: $\Delta \vdash A$.

Приклад. $\neg\exists x \neg A \vdash \forall x A$. *Враховуючи, що квантор* $\exists$ *є скороченням за правилом* (4.9)*, потрібно довести, що* $\neg\neg\forall x \neg\neg A \vdash \forall x A$.

Потрібний вивід може виглядати так:

1. $\neg\neg\forall x \neg\neg A$ (*гіпотеза*);
2. $\neg\neg\forall x \neg\neg A \rightarrow \forall x \neg\neg A$ (*похідне правило виводу* ЧВ);
3. $\forall x \neg\neg A$ (MP *до* 1 *і* 2);
4. $\forall x \neg\neg A \rightarrow \neg\neg A$ (A.4);

5. $\neg\neg A$ (MP *до* 3 *і* 4);
6. $\neg\neg A \to A$ (*похідне правило виводу* ЧВ);
7. A (MP*до* 5 *і* 6);
8. $\forall x A$ (Gen *до* 7).

На останньому кроці правило Gen *можна застосовувати, бо змінна* x *вільно до гіпотез не входить.*

Доведенням формули A називають її вивід із порожньої множини гіпотез. Якщо доведення формули A існує (тобто якщо $\vdash A$), то A називають *теоремою*.

Теорема 4.11 (про виправданість аксіоматизації)**.** *Із* $\vdash F$ *випливає* $\models F$.

Доведення. Спочатку перевіряється, чи всі аксіоми є загальнозначущими (база індукції). Далі — індукція за довжиною виводу формули F. □

Наслідок 4.2. *Кожна теорема числення предикатів є загальнозначущою формулою.*

Теорема 4.12 (про транзитивність вивідності)**.** *Якщо* $\Delta \vdash A$ *і* $\Gamma \cup \{A\} \vdash B$*, то* $\Delta, \Gamma \vdash B$.

Зауваження. *На відміну від твердження* 4.1 *про транзитивність поняття вивідності у ЧВ теорема* 4.12 *не є очевидною. Причина в тому, що коли* $A_1, \ldots, A_n = A$ *— вивід* A *з* Δ*, а* $B_1, \ldots, B_m = B$ *— вивід* B *з* $\Gamma \cup \{A\}$*, то послідовність* $A_1, \ldots, A_n A, B_1, \ldots, B_m$*, взагалі кажучи, не буде виводом* B *з* $\Delta \cup \Gamma$*. Адже в останньому випадку можуть не виконуватися обмеження на застосування правила* Gen.

Доведення. Оскільки вивід використовує лише скінченну кількість гіпотез, то множини Δ і Γ можна вважати скінченними. Тоді скінченною буде і множина X змінних, які входять вільно хоча б в одну з формул із $\Delta \cup \Gamma \cup \{A\}$. Нехай тепер

$$A_1, A_2, \ldots, A_n = A \tag{4.10}$$

— вивід A з Δ, а

$$B_1, B_2, \ldots, B_m = B \tag{4.11}$$

— вивід B з $\Gamma \cup \{A\}$. Припустимо, що вивід (4.10) містить формулу $A_k = \forall y F(y)$, яку отримують застосуванням правила Gen до формули $F(x)$, причому x не входить вільно у жодну з тих формул з Δ, які передують у виводі (4.10) формулі A_k. Візьмемо довільну змінну z, яка не входить в X, і в послідовності $A_1, A_2, \ldots, A_{k-1}$ усі вільні входження x замінимо на z. У результаті

отримаємо вивід $A'_1, \ldots, A'_{k-1}, A_k, \ldots, A_n$, в якому наявність формули A_k обґрунтовуємо застосуванням правила Gen до змінної, яка не міститься в X.

Виконуючи таке перетворення виводу (4.10) до кожної з формул A_i, які отримуємо застосуванням правила Gen, і вибираючи кожного разу нову змінну, зрештою отримаємо вивід A з Δ, в якому правило Gen не застосовується до змінних із X. Аналогічно і вивід B з $\Gamma \cup \{A\}$ можна перетворити на вивід, в якому правило Gen не застосовується до змінних із X. Якщо тепер до нового виводу A з Δ дописати новий вивід B з $\Gamma \cup \{A\}$, то отримаємо коректний вивід B з $\Delta \cup \Gamma$. □

Теорема 4.13 (дедукції). *$\Delta \vdash A \to B$ тоді й лише тоді, коли $\Delta, A \vdash B$.*

Доведення. Достатність. Якщо послідовність $F_1, \ldots, A \to B$ є виводом $A \to B$ з Δ, то послідовність $F_1, \ldots, A \to B, A, B$ (останню формулу одержують з двох попередніх за правилом MP) є виводом B з $\Delta \cup \{A\}$.

Необхідність. Покажемо індукцією за довжиною виводу

$$B_1, B_2, \ldots, B_n = B \tag{4.12}$$

формули B із множини гіпотез Δ, A, як можна переробити його у вивід

$$\ldots, A \to B_1, \ldots, A \to B_2, \ldots, A \to B_n = A \to B \tag{4.13}$$

формули $A \to B$ із множини гіпотез Δ. Враховуючи доведення теореми дедукції для числення висловлювань (теорема 4.3), досить розглянути лише випадок застосування правила Gen.

Нехай формулу $B_k = \forall y F(y)$ одержано із $B_i = F(x)$ $(i < k)$ застосуванням правила Gen і існування виводу $\Delta \vdash A \to B_i$ вже доведено. Тоді вивід із Δ формули $A \to \forall y F(y)$ має вигляд

(1) $A \to F(x)$;
(2) $\forall z(A \to F(z))$; (Gen)
(3) $\forall z(A \to F(z)) \to (A \to \forall z F(z))$; (A5)
(4) $A \to \forall z F(z)$; (MP до 3 і 2)

(Брати в цій частині виводу одразу y, а не z, не можна, бо y може вільно входити в A. Тому z — нова змінна.)

(5) $\forall z F(z) \to F(y)$; (A4)
(6) $\forall y(\forall z F(z) \to F(y))$; (Gen)
(7) $\forall y(\forall z F(z) \to F(y)) \to (\forall z F(z) \to \forall y F(y))$; (A5)
(8) $\forall z F(z) \to \forall y F(y)$. (MP до 7 і 6)

Зауважимо, що рядки (5)–(8) гіпотез не використовують, тому формула $\forall z F(z) \to \forall y F(y)$ є теоремою ЧП.

За правилом силогізму (теорема 1.7, п.3) із (4) і (8) виводиться формула $A \to \forall y F(y)$. □

Приклад. *Покажемо, що формула*

$$\forall x(A(x) \to B(x)) \to (\forall xA(x) \to \forall xB(x))$$

є теоремою числення предикатів. Застосовуючи двічі теорему дедукції, бачимо, що для цього досить показати справедливість співвідношення

$$\forall x(A(x) \to B(x)),\ \forall xA(x) \vdash \forall xB(x).$$

А для такого співвідношення неважко явно вказати потрібний вивід:

1. $\forall x(A(x) \to B(x)) \to (A(t) \to B(t))$; (A.4)
2. $\forall x(A(x) \to B(x))$; (гіпотеза)
3. $A(t) \to B(t)$; (MP *до* 1 *і* 2)
4. $\forall xA(x) \to A(t)$; (A.4)
5. $\forall xA(x)$; (*гіпотеза*)
6. $A(t)$; (MP *до* 4 *і* 5)
7. $B(t)$; (MP *до* 3 *і* 6)
8. $\forall xB(x)$. (Gen *до* 7)

Як уже зрозуміло з наведених вище прикладів, і в ЧВ, і в ЧП будувати виводи для доведення теорем, користуючись лише аксіомами і правилами виводу, не дуже зручно. Виводи навіть простих тверджень виходять дуже громіздкими і, що ще важливіше, дуже несхожими на звичні в математиці способи міркувань. Тому такі виводи використовуються головним чином у теоретичних дослідженнях, де важливо, щоб виводи мали просту структуру. До того ж у доведенні конкретних теорем зазвичай обмежуються лише *доведенням існування виводу*, а не побудовою самого виводу. Ми вже бачили, наскільки корисною в таких випадках може бути теорема дедукції або теорема про транзитивність вивідності. За подальшого вивчення цих числень неминуче приходять до численних *похідних* правил виводу. Список найважливіших похідних правил для ЧВ наведено у лемі 4.3. Для ЧП такий список можна значно розширити. Ці правила дозволяють значно спростити доведення існування виводу і близькі до звичайної практики математичні міркування, що дуже полегшує роботу математика. У кінцевому результаті приходять до поняття "неформального доведення" у звичній розмовній манері: посилання на правила виводу і тавтології вилучають і увагу звертають лише на припущення (гіпотези) та вже раніше доведені теореми. Тому пошук доведень на основі похідних правил виводу інколи називають *технікою природного виводу*. Але при цьому зберігається головна позитивна риса формальних доведень: оскільки таке доведення легко перетворити на формальне, то в разі виникнення сумніву, чи є дане неформальне доведення справді доведенням, його правильність можна з'ясувати суто механічним шляхом.

Теорема дедукції дозволяє узагальнити теорему 4.11 за допомогою теореми 4.14.

Теорема 4.14 (про виправданість аксіоматизації)**.** *Якщо* $\Delta \vdash F$*, то* $\Delta \models F$.

Доведення. Нехай $\Delta \vdash F$. Якщо врахувати, що вивід має скінченну довжину, то існує така скінченна множина $\Delta_0 = \{F_1, \ldots, F_n\} \subseteq \Delta$, що $\Delta_0 \vdash F$. За теоремою дедукції останнє співвідношення рівносильне співвідношенню

$$\vdash F_1 \to (F_2 \to (\cdots \to (F_n \to F) \cdots)), \tag{4.14}$$

яке за теоремою 4.11 рівносильне співвідношенню

$$\models F_1 \to (F_2 \to (\cdots \to (F_n \to F) \cdots)). \tag{4.15}$$

За наслідком 1.2, який справедливий і для логіки предикатів, із (4.15) отримуємо $\Delta_0 \models F$ і, через те, що $\Delta_0 \subseteq \Delta$, то $\Delta \models F$. □

Якщо $\Delta \vdash 0$, то множину формул Δ називають *дедуктивно суперечливою* або *дедуктивно несумісною*. У протилежному разі Δ називають *дедуктивно несуперечливою* або *дедуктивно сумісною*.

Зрозуміло, що кожна підмножина дедуктивно несуперечливої множини також є дедуктивно несуперечливою. Крім того (адже у виводі бере участь лише скінченна множина формул), кожна нескінченна дедуктивно суперечлива множина містить скінченну суперечливу підмножину.

З теореми 4.14 про виправданість аксіоматизації одразу випливає наслідок.

Наслідок 4.3. *Якщо множина формул* Δ *має модель, то ця множина є дедуктивно несуперечливою.*

Зауваження. *Хоча суттєві риси числення предикатів описані ще Фреге 1879 р., перше точне формулювання цього числення як самостійної формальної системи належить Гільберту і Аккерману у 1928 р.*

4.4. Основні теореми ЧП та зв'язки між ними

Із теореми 4.14 про виправданість аксіоматизації випливає, що в ЧП теоремами будуть лише загальнозначущі формули і воно несуперечливе. Для несуперечливої теорії має сенс ставити питання про її *повноту*, тобто чи містить вона достатню для певної мети кількість теорем. Залежно від цієї мети з'являються різні уточнення дуже загального поняття повної теорії.

У випадку ЧП мета полягає в тому, щоб кожна загальнозначуща формула була теоремою. І ми покажемо, що в цьому сенсі ЧП справді є повною теорією. Важливою для доведення цього факту є теорема 4.15.

Теорема 4.15 (про несуперечливість)**.** *Множина формул Δ має модель тоді й лише тоді, коли Δ є дедуктивно несуперечливою.*

Нам знадобиться лема 4.7.

Лема 4.7. *Об'єднання довільного зростаючого ланцюга*

$$\cdots \subseteq \Omega_{i-1} \subseteq \Omega_i \subseteq \Omega_{i+1} \subseteq \cdots$$

дедуктивно несуперечливих множин формул також є дедуктивно несуперечливим.

Доведення. Припустимо, що множина є $\Omega = \bigcup_i \Omega_i$ є дедуктивно суперечливою. Тоді $\Omega \vdash 0$. У цьому виводі трапляється лише скінченна кількість гіпотез із Ω, кожна з яких трапляється в якійсь множині Ω_i. Серед цієї скінченної множини індексів можна вибрати найбільший. Нехай це j. Тоді всі гіпотези з виводу $\Omega \vdash 0$ трапляються вже в Ω_j і $\Omega_j \vdash 0$. Але це суперечить припущенню, що всі Ω_i дедуктивно несуперечливі. □

Доведення теореми 4.15. *Необхідність*. Нехай Δ має модель $\mathfrak{M}$, але є дедуктивно суперечливою (тобто для деякої формули F як $\Delta \vdash F$, так і $\Delta \vdash \neg F$). Із теореми 4.14 про виправданість аксіоматизації випливає, що в моделі $\mathfrak{M}$ обидві формули F і $\neg F$ будуть істинними. Однак це неможливо.

Достатність. Що можна використати для побудови моделі? Лише те, що є в самому ЧП: алфавіт, терми, формули. Тут є певна аналогія з теоремою Келі для груп, де групу зображують підстановками елементів самої групи.

Ідея побудови моделі для множини формул Δ виглядає так.

У нас є деяке ЧП *з алфавітом L, множина $F(L)$ усіх формул цього числення і деяка дедуктивно несуперечлива підмножина $\Delta \subset F(L)$. Ми хочемо побудувати розширення алфавіту $L' \supseteq L$ і таку множину формул $\Delta' \subseteq F(L')$, які б задовольняли умови*:

(a) $\Delta' \supseteq \Delta$;

(b) *множина Δ' — дедуктивно несуперечлива*;

(c) *для довільної формули $F \in F(L')$ множина Δ' містить рівно одну з формул F і $\neg F$*;

(d) *якщо Δ' містить формулу вигляду $\exists x F(x)$, то в L' знайдеться такий терм t, що $F(t) \in \Delta'$.*

Зрозуміло, що кожна модель множини Δ' буде і моделлю множини Δ. А коли Δ' задовольняє умови (b)–(d), то модель для Δ' можна побудувати так.

— Основною множиною є множина $M = T(L')$ термів над алфавітом L'.

— Функціональний символ f арності n інтерпретують як функцію від n змінних, яка кожному наборові термів $(t_1, \ldots, t_n)$ ставить у відповідність терм $f(t_1, \ldots, t_n)$.

— Предикатний символ P арності n інтерпретують як n-арний предикат, який на наборі термів $(t_1, \ldots, t_n)$ дорівнює 1 тоді й лише тоді, коли $P(t_1, \ldots, t_n) \in \Delta'$.

— У деталізації цієї інтерпретації змінній x надамо значення x (Увага! Останній символ x розглядаємо вже як терм — елемент множини $T(L')$).

Зауважимо, що за такої деталізації значенням кожного терма t буде він сам. Крім того, кожна формула, яка виводиться з Δ', уже із самого початку міститься в Δ'. Справді, якщо $\Delta' \vdash F$, але $F \notin \Delta'$, то $\neg F \in \Delta'$ і $\Delta' \vdash \neg F$, що суперечить дедуктивній несуперечливості множини Δ'.

Лема 4.8. *Якщо множина формул Δ' задовольняє умови* (b) *і* (c)*, то з $\Delta' \vdash F$ випливає $F \in \Delta'$.*

Доведення. Нехай $\Delta' \vdash F$. Якщо $F \notin \Delta'$, то згідно із (c) $\neg F \in \Delta'$. Але тоді $\Delta' \vdash \neg F$, що суперечить умові (b). □

Лема 4.9. *Якщо розширення $L' \supseteq L$ і $\Delta' \supseteq \Delta$ задовольняють умови* (b)–(d)*, то описана вище деталізація справді є моделлю множини Δ'. Точніше, формула F є істинною у цій деталізації тоді й лише тоді, коли $F \in \Delta'$.*

Доведення. Значення істинності формули F у цій деталізації позначимо через $\varphi(F)$. Застосуємо індукцію за кількістю l зв'язок у формулі. Базою індукції є атомарні формули ($l = 0$), для яких твердження леми випливає з правила інтерпретації предикатних символів.

Припустимо тепер, що для всіх формул, які містять не більше ніж l зв'язок, твердження леми вже доведено. Нехай формула F містить $l + 1$ зв'язку. Залежно від головної зв'язки F може мати один із двох виглядів:

$$\text{I. } F = A \to B; \qquad \text{II. } F = \forall x A(x)\,.$$

За припущенням індукції для формул A і B твердження леми виконується.

I. Розіб'ємо цей випадок на три підвипадки.

I.1. $\varphi(A) = 0$. Тоді $\varphi(A \to B) = 1$. За припущенням індукції, $A \notin \Delta'$, а тому $\neg A \in \Delta'$. Оскільки $\neg A, A \vdash B$, то $\neg A \vdash A \to B$. Отже, $\Delta' \vdash A \to B$, а тому $A \to B \in \Delta'$.

I.2. $\varphi(B) = 1$. Тоді $\varphi(A \to B) = 1$. За припущенням індукції, $B \in \Delta'$. Оскільки з $\Delta' \vdash B$ і $\Delta' \vdash B \to (A \to B)$ випливає, що $\Delta' \vdash A \to B$, то $A \to B \in \Delta'$.

I.3. $\varphi(A) = 1$ і $\varphi(B) = 0$. Тоді $\varphi(A \to B) = 0$. За припущенням індукції, $A \in \Delta'$, $B \notin \Delta'$, $\neg B \in \Delta'$. Із співвідношення $A, \neg B \vdash \neg(A \to B)$ випливає, що $\Delta' \vdash \neg(A \to B)$ і $\neg(A \to B) \in \Delta'$. Але тоді $A \to B \notin \Delta'$.

II. Досить довести, що $\varphi(F) = 0$ тоді й лише тоді, коли $F \notin \Delta'$.

Нехай спочатку $\varphi(\forall x A(x))=0$. Тоді існує такий терм t, що $A(t)=0$. Зауважимо, що коли формули F_1 і F_2 відрізняються одна від одної лише перепозначенням пов'язаних змінних, то $F_1 \equiv F_2$. Більше того, для таких формул індукцією за кількістю зв'язок легко доводиться, що $F_1 \vdash F_2$ і $F_2 \vdash F_1$. Отже, такі дві формули містяться (або не містяться) в Δ' одночасно. Тому без обмеження загальності можна вважати, що жодна змінна, яка зустрічається в t, не є пов'язаною в F. З аксіоми (A4) і теореми дедукції випливає: $\forall x A(x) \vdash A(t)$. Тому, коли б формула $\forall x A(x)$ містилась у Δ', то в Δ' містилась би і формула $A(t)$. Однак за припущенням індукції, $A(t) \notin \Delta'$. Отже, $\forall x A(x) \notin \Delta'$.

Нехай тепер $\forall x A(x) \notin \Delta'$. Зауважимо, що $\neg\neg\forall x\neg\neg A(x) \vdash \forall x A(x)$ (див. приклад на с. 84). Тому $\neg\neg\forall x\neg\neg A(x) \notin \Delta'$. Але $\neg\neg\forall x\neg\neg A(x) = = \neg\exists x\neg A(x)$. Тому $\exists x\neg A(x) \in \Delta'$. Отже, існує такий терм t, що $\neg A(t) \in \Delta'$. Але тоді $A(t) \notin \Delta'$. Звідси $\varphi(A(t)) = 0$ і $\varphi(\forall x A(x)) = 0$. □

Побудову потрібних розширень $L' \supseteq L$ і $\Delta' \supseteq \Delta$ розіб'ємо на три етапи.

A. Спочатку покажемо, що можна побудувати розширення $\Delta_1 \supseteq \Delta$, яке б задовольняло умови (b) і (c). Для цього розглянемо множину

$$M = \{\Omega \subseteq F(L) \mid \Delta \subseteq \Omega \text{ і } \Omega \text{ є дедуктивно несуперечливою}\}.$$

Згідно з лемою 4.7 частково впорядкована за включенням множина M задовольняє умову леми Цорна. Нехай Δ_1 — довільний максимальний елемент із M. Оскільки множина Δ_1 — дедуктивно несуперечлива, то з кожної пари формул F і $\neg F$ вона містить не більше однієї. Доведемо, що принаймні одну із цих формул множина Δ_1 містить. Справді, у протилежному разі з максимальності Δ_1 випливало б, що кожна з множин $\Delta_1 \cup \{F\}$ і $\Delta_1 \cup \{\neg F\}$ є дедуктивно суперечливою. Але із співвідношень $\Delta_1, F \vdash 0$, $\Delta_1, \neg F \vdash 0$ і теореми дедукції отримуємо, що $\Delta_1 \vdash F \to 0$ і $\Delta_1 \vdash \neg F \to 0$. Звідси та із співвідношення $F \to 0, \neg F \to 0 \vdash 0$ у свою чергу випливало б, що $\Delta_1 \vdash 0$, тобто дедуктивна суперечливість множини Δ_1.

Таким чином, множина Δ_1 задовольняє умови (b) і (c).

B. Далі покажемо, як можна побудувати розширення $L^1 \supseteq L$ і $\Delta^1 \supseteq \Delta$, які задовольняли б умови (b) і (d). Для цього для кожної формули A вигляду $A = \exists x F(x)$ із Δ приєднаємо до L нову змінну y_A, а до Δ — нову формулу $F(y_A)$. Якби отримана множина Δ^1 вийшла дедуктивно суперечливою, то існував би вивід $\Delta^1 \vdash 0$, який через свою скінченність використовував би лише

скінченну кількість нових формул $F(y_A)$. Тому для доведення дедуктивної несуперечливості множини Δ^1 досить розглянути випадок приєднання до L однієї змінної y_A, а до Δ — однієї формули $F(y_A)$.

Отже, припустимо, що Δ дедуктивно несуперечлива, $\exists x F(x) \in \Delta$, змінна y не зустрічається в Δ, а множина $\Delta \cup \{F(y)\}$ — дедуктивно суперечлива. Тоді $\Delta, F(y) \vdash 0$. За теоремою дедукції $\Delta \vdash F(y) \to 0$. Крім того, $\vdash (F(y) \to 0) \to \neg F(y)$, бо $(A \to 0) \to \neg A$ — тавтологія. Отже, $\Delta \vdash \neg F(y)$. Оскільки y не зустрічається в Δ, то можна застосовувати правило узагальнення: $\Delta \vdash \forall x \neg F(x)$. Але за припущенням множині Δ належить формула $\exists x F(x) = \neg\forall x \neg F(x)$, тому $\Delta \vdash \neg\forall x \neg F(x)$. Таким чином, припущення про дедуктивну суперечливість $\Delta \cup \{F(y)\}$ приводить до суперечності з припущенням про дедуктивну несуперечливість Δ[37].

C. Розглянемо тепер два ланцюги розширень:

$$\Delta \subseteq \Delta_1 \subseteq \Delta^1 \subseteq \Delta_2 \subseteq \Delta^2 \subseteq \cdots \subseteq \Delta_k \subseteq \Delta^k \subseteq \Delta_{k+1} \subseteq \cdots$$

і

$$L \subseteq L^1 \subseteq L^2 \subseteq \cdots \subseteq L^k \subseteq \cdots,$$

де кожне з розширень $\Delta^k \subseteq \Delta_{k+1}$ будується як у пункті A, а кожне з розширень $\Delta_k \subseteq \Delta^k$ і $L^{k-1} \subseteq L^k$ — як у пункті B. Тоді множина $\Delta' = \bigcup_n \Delta_n$ як множина формул над алфавітом $L' = \bigcup_n L^n$ задовольняє всі три умови (b)–(d). Справді, несуперечливість Δ' випливає з леми 4.7. Якщо F — довільна формула над алфавітом L', то оскільки F містить скінченну кількість символів, F буде формулою над деяким алфавітом L^k. Але тоді множина Δ_{k+1} (а тим самим і множина Δ') містить рівно одну з формул F і $\neg F$ (Δ' не може містити обидві формули через свою несуперечливість). Нарешті, якщо Δ' містить формулу вигляду $\exists x F(x)$, то ця формула міститься в якійсь множині Δ_k. Але тоді для деякого терму t множина Δ^k містить формулу $F(t) \in \Delta'$. □

Уже давно є усталена думка, що в математиці "існувати" — це бути вільним від протиріч[38]. Теорему про несуперечливість можна розглядати як нехай і обмежене рамками логіки предикатів, але строге оформлення цієї думки.

З теоремою про несуперечливість безпосередньо пов'язано ще кілька важливих теорем.

[37] Взагалі кажучи, у попередніх міркуваннях суперечливість Δ доводилася над алфавітом $L \cup \{y\}$, але оскільки змінних в L нескінченно багато, а у виводах формул $\forall x \neg F(x)$ і $\neg\forall x \neg F(x)$ бере участь лише скінченна кількість гіпотез, а тому і скінченна кількість змінних, то ці виводи можна переробити на виводи над алфавітом L.

[38] Можливо, першим, хто її озвучив понад 100 років тому, був знаменитий французький математик Пуанкаре.

Теорема 4.16 (адекватності)**.** $\Delta \vdash F$ *тоді й лише тоді, коли* $\Delta \models F$.

Теорему адекватності часто називають *теоремою Ґеделя про повноту*. Назва "теорема адекватності" акцентує увагу на тому, що *граматика* (синтаксис) нашого числення визначена добре: ми одержуємо все, що треба (тобто всі загальнозначущі формули будуть теоремами), і нічого зайвого.

Теорема 4.17 (принцип локалізації)**.** *Якщо* $\Delta \models F$*, то* $\Delta_0 \models F$ *для деякої скінченної підмножини* $\Delta_0 \subseteq \Delta$.

Теорема 4.18 (компактності)**.** *Множина формул* Δ *має модель тоді й лише тоді, коли кожна скінченна підмножина* $\Delta_0 \subseteq \Delta$ *має модель.*

Зв'язки між теоремами 4.15–4.18 описують такою діаграмою:

$$4.15 \iff 4.16 \implies 4.17 \iff 4.18.$$

Доведення. a) 4.15 $\implies$ 4.16. Враховуючи теорему про виправданість аксіоматизації, потрібно довести лише, що з $\Delta \models F$ випливає $\Delta \vdash F$. Нехай $\Delta \models F$. Тоді множина $\Delta \cup \{\neg F\}$ не має моделі і за теоремою 4.15 буде дедуктивно суперечливою. Тому з $\Delta \cup \{\neg F\}$ виводиться кожна формула, зокрема, $\Delta, \neg F \vdash F$. Звідси, за теоремою дедукції, $\Delta \vdash \neg F \to F$. Після цього вивід із множини Δ формули F будується вже легко:

1. $\neg F \to \neg F$ (тавтологія);
2. $\neg F \to F$ (виводиться з Δ);
3. $(\neg F \to \neg F) \to ((\neg F \to F) \to F)$ (тавтологія);
4. $(\neg F \to F) \to F$ (MP до 3 і 1);
5. F (MP до 4 і 2).

b) 4.16 $\implies$ 4.15. З теореми 4.16 (і навіть з її легшої половини — теореми 4.14 про виправданість аксіоматизації) випливає, що множина формул Δ, яка має модель, є дедуктивно несуперечливою. Тому досить довести лише зворотне твердження. Але якщо дедуктивно несуперечлива множина формул Δ не має моделі, то $\Delta \models 0$. За теоремою 4.16 звідси маємо $\Delta \vdash 0$, що суперечить припущенню.

c) 4.16 $\implies$ 4.17. Якщо $\Delta \models F$, то $\Delta \vdash F$. Оскільки вивід має скінченну довжину, то існує така скінченна підмножина $\Delta_0 \subseteq \Delta$, що $\Delta_0 \vdash F$. Але тоді $\Delta_0 \models F$.

d) 4.17 $\implies$ 4.18. Нехай Δ не має моделі. Тоді $\Delta \models 0$ і за теоремою 4.17 існує така скінченна підмножина $\Delta_0 \subseteq \Delta$, що $\Delta_0 \models 0$, тобто Δ_0 не має моделі.

е) 4.18 $\Longrightarrow$ 4.17. Нехай $\Delta \models F$. Тоді множина $\Delta \cup \{\neg F\}$ не має моделі. Отже, не має моделі й деяка скінченна підмножина $\Delta_0 \subseteq \Delta \cup \{\neg F\}$. Можна вважати, що $\Delta_0 = \Delta_1 \cup \{\neg F\}$ де $\Delta_1 \subseteq \Delta$. Але тоді $\Delta_1 \models F$. $\square$

4.5. Деякі наслідки основних теорем*

У доведенні теореми 4.15 про несуперечливість ЧП носієм моделі несуперечливої множини формул Δ є множина термів цього ЧП. Термами, зокрема, є змінні, тому множина термів нескінченна. З нескінченності початкового алфавіту ЧП випливає, що за тих розширень мови, які будувалися в доведенні теореми 4.15, потужність алфавіту не змінювалася. Тому із цього доведення одразу випливає така теорема.

Теорема 4.19 (Льовенгейма — Сколема)**.** *Якщо алфавіт L числення предикатів має нескінченну потужність $\mathfrak{m}$, то кожна дедуктивно несуперечлива множина формул цього числення має модель потужності $\mathfrak{m}$.*

Зауваження. *Виникає природне питання: а що буде, коли алфавіт містить предикат рівності, а Δ містить формулу, яка вимагає, наприклад, щоб модель мала не більше ніж k елементів? Чи не суперечить це теоремі Льовенгейма — Сколема? Суперечність насправді уявна: у цьому випадку в моделі з доведення теореми 4.15 предикат рівності буде інтерпретуватися не відношенням рівності, а лише відношенням еквівалентності. Щоб отримати так звану нормальну модель, коли предикат рівності інтерпретується саме як відношення рівності, треба перейти до множини класів еквівалентності. А їх уже буде скінченна кількість.*

Із теореми Льовенгейма — Сколема випливає існування для деяких теорій дуже несподіваних моделей. Чи не найбільшою несподіванкою став так званий *парадокс Сколема* — існування *нестандартної* моделі для теорії множин. На межі XIX–XX ст. у канторівській теорії множин (її зараз називають *наївною*) виявили багато суперечностей, які скромно назвали *парадоксами*. Поява парадоксів була пов'язана із занадто вільним трактуванням поняття множини. Оскільки важливість теорії множин для математики на той момент була вже безсумнівною, від цих суперечностей треба було позбавитись. Тому запропоновано обмежитися розглядом лише тих множин, які дозволяються певним списком аксіом[39]. Виокремлено кілька варіантів такої аксіоматизації[40]. Однак в усіх випадках множина аксіом є *зліченною* і використовує зліченний алфавіт. А тому за теоремою 4.19 *для теорії множин має існувати* ***злічена*** *модель*. На перший погляд це суперечить існуванню незліченних множин, зокрема і теоремам Кантора про незліченність $\mathbb{R}$ і незліченність множини $\mathcal{B}(\mathbb{N})$ всіх підмножин множини $\mathbb{N}$. Суперечність зникає, якщо зрозуміти, що насправді теореми Кантора говорять

[39] У результаті відповідно перероблена теорія множин навіть зміцнила своє становище в математиці — на теоретико-множинну основу були переведені майже всі її розділи. А парадокси перестали турбувати переважну більшість математиків і зайняли свою скромну нішу — стали предметом досліджень вузького кола спеціалістів з основ теорії множин.

[40] Нині стандартною системою аксіом для теорії множин вважають так звану *систему аксіом Цермело — Френкеля*.

лише про *неможливість* у межах формальної теорії множин установити взаємно однозначну відповідність між $\mathbb{N}$ та кожною із цих множин.

Серед численних застосувань теореми компактності для ЧП розглянемо лише одне з найпростіших.

Теорема 4.20. *Якщо множина формул Δ має скінченні моделі як завгодно великої потужності, то вона має і нескінченні моделі.*

Доведення. Нехай множина формул Δ має скінченні моделі як завгодно великої потужності. Поповнимо алфавіт зліченною кількістю індивідних констант $c_1, c_2, \ldots$ і для кожної пари $i \neq j$ індексів розглянемо формулу $c_i \neq c_j$. Нехай Θ — множина всіх таких формул. Тоді множина $\Delta' = \Delta \cup \Theta$ буде локально несуперечливою. Справді, якщо підмножина $\Delta_0 \subset \Delta'$ — скінченна, то вона містить лише скінченну множину формул з Θ. А тому у формулах з Δ_0 зустрічається лише скінченна кількість нових констант $c_{i_1}, \ldots, c_{i_k}$. Але тоді в тій моделі $\mathfrak{M}_0$ множини Δ, яка має не менше ніж k елементів, ці константи можна проінтерпретувати попарно різними елементами. У такій інтерпретації констант усі формули з Δ_0 стануть істинними в $\mathfrak{M}_0$, і $\mathfrak{M}_0$ буде моделлю множини Δ_0.

За теоремою компактності множина Δ' повинна мати модель $\mathfrak{M}$. У цій моделі різні константи $c_1, c_2, \ldots$ повинні інтерпретуватися різними елементами. Тому $\mathfrak{M}$ мають містити нескінченно багато елементів. Зрозуміло, що $\mathfrak{M}$ є також моделлю множини Δ. □

З теореми 4.20, зокрема, випливає, що хоча багато класів алгебричних структур, як-от групи, кільця, поля, булеві алгебри тощо, задаються наборами аксіом, побудувати аксіоматичну теорію *скінченних* груп, кілець і т. д. не можна в принципі.

4.6. Силогістика Арістотеля*

Багато людей, зокрема і серед математиків, уважають, що першою в історії науки аксіоматичною системою є геометрія Евкліда. Насправді геометрія Евкліда була другою. Першою в історії людської думки аксіоматичною системою була силогістика Арістотеля, викладена ним у його «Першій аналітиці». Детальний виклад силогістики не входить у наші плани, тому зупинимося на ній лише коротко.

Традиційна логіка має справу з *поняттями*, які поділяють на *одиничні* та *загальні*. Одиничне поняття — це просто ім'я певного предмета. Зміст загального поняття визначають указуванням сукупності тих властивостей, що характеризують усі предмети, які підпадають під це поняття. Клас предметів, які мають цю характеристичну сукупність властивостей, утворює об'єм поняття.

Одномісні предикати можна розглядати як *властивості* предметів. Сукупність властивостей $P_1, \ldots, P_n$ можна замінити однією: "мати кожну з властивостей $P_1, \ldots, P_n$". Тому з погляду змісту "загальне поняття" традиційної логіки можна ототожнити з одномісним предикатом.

Для кожного одномісного предиката P можна утворити *клас* $M = \{x \mid P(x)\}$ усіх предметів, що мають властивість P. Цей клас і характеризує *об'єм* поняття.

Арістотель розглядав 4 типи *суджень* (у нашій термінології — висловлювань):
— загальностверджувальні: "усі S є P";
— загальнозаперечувальні: "жодне S не є P";
— частковостверджувальні: "деякі S є P";
— частковозаперечувальні: "деякі S не є P".

Висловлювання таких типів традиційно позначають $A(S,P)$, $E(S,P)$, $I(S,P)$ і $O(S,P)$ відповідно.

Виокремлювали судження, по-перше, "за якістю": $A(S,P)$, $I(S,P)$ — *стверджувальні*, $E(S,P)$,$E(S,P)$, $O(S,P)$ — *заперечувальні*; по-друге, "за кількістю": $A(S,P)$, $E(S,P)$ — *загальні*, $I(S,P)$, $O(S,P)$ — *часткові*.

Основним питанням арістотелевої силогістики було існування правил, які дозволяли б виводити судження (*висновок*) одного з типів $A(S,P)$, $E(S,P)$, $I(S,P)$ і $O(S,P)$ із двох інших суджень (*засновків*) типів A, E, I або O, з яких перше пов'язує поняття P із третім поняттям M, а друге — поняття S із тим самим третім поняттям M.

Чи не найвідомішим із таких правил є так званий *модус силогізму* $bArbArA$, який у традиційних позначеннях записують так:

$$\frac{\begin{array}{c} A\,(M,P) \\ A\,(S,M) \end{array}}{A\,(S,P)} \quad .$$

Розглянемо два одномісні предикати $F(x)$ і $G(x)$, і нехай змінна x пробігає елементи фіксованої непорожньої множини D. Покладемо $S = \{x \in D \mid F(x)\}$, $P = \{x \in D \mid G(x)\}$. Тоді всі чотири типи суджень Арістотеля можна переписати в теоретико-множинних або логічних термінах:

Традиційне позначення	Теоретико-множинна інтерпретація	Логічна інтерпретація
$A(S,P)$	$S \subseteq P$	$\forall x(F(x) \to G(x))$
$E(S,P)$	$S \cap P = \varnothing$	$\forall x(F(x) \to \neg G(x))$
$I(S,P)$	$S \cap P \neq \varnothing$	$\exists x(F(x) \wedge G(x))$
$O(S,P)$	$\neg(S \subseteq P)$	$\exists x(F(x) \wedge \neg G(x))$

Модус $bArbArA$ у нових позначеннях тепер можна переписати, наприклад, так:

$$\big((M \subseteq P) \wedge (S \subseteq M) \to (S \subseteq P)\big) \, .$$

У термінах суджень Арістотеля можливі чотири схеми правил виводу (у традиційній термінології — 4 *фігури силогізму*):

$$\begin{array}{cccc} I & II & III & IV \\ *\,(M,P) & *\,(P,M) & *\,(M,P) & *\,(P,M) \\ \underline{*\,(S,M)} & \underline{*\,(S,M)} & \underline{*\,(M,S)} & \underline{*\,(M,S)} \\ *\,(S,P) & *\,(S,P) & *\,(S,P) & *\,(S,P) \end{array}$$

У кожній із цих фігур букви A, E, I або O можна розставити $4^3 = 64$ способами. Всього одержимо 256 можливих правил виводу (модусів силогізму). Однак застосування багатьох із цих правил приводитиме до помилок. Ті модуси силогізму, за допомогою яких з істинних засновків завжди одержують істинні висновки, називають *правильними*[41]. Усі вони одержали спеціальні назви:

I фігура	II фігура	III фігура	IV фігура
$b\mathbf{A}rb\mathbf{A}r\mathbf{A}$	$c\mathbf{E}s\mathbf{A}r\mathbf{E}$	$d\mathbf{A}t\mathbf{I}s\mathbf{I}$	$c\mathbf{A}l\mathbf{E}m\mathbf{E}s$
$c\mathbf{E}l\mathbf{A}r\mathbf{E}nt$	$c\mathbf{A}m\mathbf{E}str\mathbf{E}s$	$t\mathbf{E}r\mathbf{I}s\mathbf{O}$	$fr\mathbf{E}s\mathbf{I}s\mathbf{O}n$
$d\mathbf{A}r\mathbf{II}$	$f\mathbf{E}st\mathbf{I}n\mathbf{O}$	$d\mathbf{II}s$	$d\mathbf{I}m\mathbf{A}t\mathbf{I}s$
$f\mathbf{E}r\mathbf{IO}$	$b\mathbf{A}r\mathbf{O}c\mathbf{O}$	$b\mathbf{O}c\mathbf{A}d\mathbf{O}$	$+\, b\mathbf{A}m\mathbf{A}l\mathbf{I}p$
$\star\, b\mathbf{A}b\mathbf{A}r\mathbf{I}$	$\star\, c\mathbf{E}s\mathbf{A}r\mathbf{O}$	$+\, d\mathbf{A}r\mathbf{A}pt\mathbf{I}$	$+\, f\mathbf{E}s\mathbf{A}p\mathbf{O}$
$\star\, c\mathbf{E}l\mathbf{A}r\mathbf{O}nt$	$\star\, c\mathbf{A}m\mathbf{EO}stro$	$+\, f\mathbf{E}l\mathbf{A}pt\mathbf{O}n$	$\star\, c\mathbf{A}m\mathbf{E}n\mathbf{O}$

Виділені жирним шрифтом голосні букви у цих назвах вказують на вибір букв A, E, I або O. Наприклад, модус $f\mathbf{E}l\mathbf{A}pt\mathbf{O}n$ має вигляд

$$\begin{array}{c} E\,(M,P) \\ \underline{A\,(M,S)} \\ O\,(S,P) \end{array}.$$

В арістотелевій логіці правильних модусів усього 19. Це ті модуси з таблиці, біля яких немає знака $\star$. Крім того, є ще 5 так званих *слабких* модусів (у таблиці вони відмічені знаком $\star$). Слабкі модуси одержують із тих правильних модусів, які мають висновок загального типу, заміною його на висновок часткового типу. З урахуванням і слабких модусів для кожної фігури маємо по 6 правильних модусів. Вони і становлять систему аксіом арістотелевої логіки: міркування буде правильним, якщо воно використовує лише правильні модуси.

Зауваження. *У традиційній логіці, починаючи з Арістотеля і аж до XX ст., не визнавали понять із порожнім об'ємом. Тому в цій логіці було 19 правильних модусів. Однак для математиків така позиція є дуже незручною, бо не завжди вдається встановити, чи є об'єм поняття непорожнім (скажімо, досі невідомо, чи є порожнім об'єм поняття "непарне досконале число"). Але коли допустити поняття з порожнім об'ємом, то частина арістотелевих модусів стає хибною (у таблиці вони відмічені знаком $+$).*

[41] Для знаходження всіх правильних модусів зручно користуватися діаграмами Венна.

5. Алгоритми (ефективна обчислювальність)

5.1. Історія розвитку поняття алгоритму. Приклади

У повсякденному житті під алгоритмом розуміють набір правил дій, який за певних умов приводить до однозначного розвитку подій і певного результату. Наприклад, програма для комп'ютера, кулінарний рецепт, статут охоронної служби, інструкції дипломату для переговорів.

Тому теорію алгоритмів можна розуміти як дуже широко — дослідження алгоритмів і алгоритмічної діяльності у різних сферах життя, до біологічної і психологічної включно, так і значно вужче — як розроблення конкретних алгоритмів (зокрема, як добре зробити те, що взагалі можна зробити). Нас ці питання цікавитимуть лише в дуже обмеженій сфері діяльності — у математиці.

Уже на початковому етапі розвитку математики (Давній Єгипет, Вавилон) для розв'язування певних типів задач стали використовувати різні процедури суто механічного характеру. Наприклад, для обчислення площ фігур, об'ємів тіл чи коренів квадратного рівняння для знаходження невідомої величини виконували певну послідовність обчислень за жорстко встановленими правилами. У Давній Греції коло таких процедур значно розширилося: розв'язування різних задач на побудову за допомогою циркуля і лінійки, знаменитий алгоритм Евкліда тощо.

У IX ст. аль-Хорезмі розробив правила чотирьох арифметичних дій над числами у десятковій системі числення. У Європі сукупність цих правил отримала назву "алгоризм", яка пізніше перетворилася на "алгоритм". Згодом цей термін набув у математиці набагато ширшого значення і став означати не лише правила виконання арифметичних дій, а й довільний більш-менш чітко описаний набір правил для розв'язування різних класів задач.

Величезний інтерес математиків до алгоритмів чи, у трохи ширшому сенсі, до конструктивних результатів викликаний тим, що наявність алгоритму для розв'язування певного класу задач дозволяє, хоча б у принципі, розв'язувати ці задачі без особливих інтелектуальних зусиль, і в цьому сенсі тривіалізує певну область математики.

Кількастолітній прогрес у розвитку таких методів породив переконання в тому, що остаточним розв'язанням кожної математичної, і навіть філософської, проблеми має бути відшукання відповідного алгоритму. Яскравими представниками таких поглядів були Декарт (аналітична геометрія створювалася ним із метою зробити геометрію доступною алгебричним методам і замінити міркування обчисленнями, тобто алгоритмізувати геометрію),

Ляйбніц, Гільберт (зокрема, у частині його знаменитих проблем ідеться про пошук алгоритмів для розв'язання певних класів задач).

У XX ст. слово "алгоритм" вийшло далеко за межі математики і стало означати чітко сформульований набір правил-інструкцій, строге і послідовне виконання яких дозволяє одержати бажаний результат.

До XX ст. математика переважно мала справу із числами й обчисленнями. За незначними винятками поняття алгоритму ототожнювалося з поняттям методу обчислень. Типовим прикладом може слугувати питання про розв'язність у цілих числах діофантового рівняння $ax^2 + bxy + cu^2 = d$: було вказано обчислювальний процес, який для довільних цілих чисел a, b, c і d дозволяв за скінченну кількість кроків дати відповідь на це питання. Зокрема, жодного разу не виникало різних думок і сумнівів щодо того, вважати чи ні конкретний процес алгоритмом. Тому поняття алгоритму мало швидше методологічне значення, хоча поступово і набувало все більшої чіткості.

Таке положення було задовільним, доки різні алгоритмічні проблеми вдавалося розв'язувати шляхом побудови відповідних алгоритмів. Положення суттєво змінилося на початку другої чверті XX ст., коли з'явилися перші сумніви стосовно існування алгоритмів для розв'язання деяких задач. Проблеми доведення існування алгоритму і доведення його відсутності є принципово різними: якщо першу можна розв'язати, побудувавши відповідний алгоритм (для чого достатньо й інтуїтивного поняття), то для доведення неіснування треба точно знати, неіснування чого доводиться. Інтуїтивного поняття алгоритму як *рецепта досягнення результату за допомогою однозначної послідовності дій* для цього недостатньо. Незрозуміло, якими засобами може виражатися цей рецепт, які дії можна використовувати, хто оцінює однозначність і т. д. [42].

Тому постала задача формалізації поняття алгоритму, його точного визначення. Було помічено, що в математиці алгоритми — це набори правил чи інструкцій, які дозволяють, відштовхуючись від одного тексту (вхідних даних задачі), однозначно будувати інший текст (відповідь задачі). Зокрема, шкільні правила додавання і множення чисел у стовпчик насправді є не правилами додавання і множення чисел, а правилами перероблення *записів* цих чисел у десятковій системі числення у *записи* їхніх суми і добутку. Якщо працювати, наприклад, не в десятковій системі, а в системі числення з основою 16 із цифрами $0, 1, \ldots, 9, a, b, c, d, e, f$, то шкільні правила додавання і множення у стовпчик треба змінювати. Зокрема, однакові записи будуть позначати в

[42] Задовільнішим (особливо за нинішнього поширення комп'ютерів) є означення алгоритму як *програми на певній універсальній мові програмування* (напр., на Бейсіку).

цих системах різні числа, а для суми і добутку таких чисел ми одержимо і різні записи.

Вперше завдання строгого означення поняття алгоритму було вирішене в середині 30-х рр. минулого століття. Майже одразу з'явилося кілька різних підходів до такого означення. Але всі вони мали спільні риси:
a) чітко окреслюється набір текстів, з яким працюють алгоритми;
b) фіксується набір дозволених перетворень текстів (елементарних кроків алгоритмів);
c) вказуються правила, як із цих елементарних кроків будувати алгоритми.

Після формалізації поняття алгоритму стало можливим доведення алгоритмічної нерозв'язності конкретних математичних проблем. Перші такі приклади стосувалися логіки (з деякими з них ми пізніше познайомимося), але згодом отримано ряд важливих результатів з інших областей математики. Першим прикладом алгоритмічно нерозв'язної масової задачі поза межами логіки був такий.

Приклад. *Розглядають групи, задані твірними і визначальними співвідношеннями:*

$$G = \langle a_1, a_2, \ldots \mid R_1, R_2, \ldots \rangle,$$

де $A = \{a_1, a_2, \ldots\}$ *— множина твірних групи, а визначальні співвідношення* $R_1, R_2, \ldots$ *мають вигляд* $w = e$ *або* $w_1 = w_1$*, де* w, w_1, w_2 *— слова в алфавіті* $A \cup A^{-1}$ *(як кількість твірних, так і кількість визначальних співвідношень може бути нескінченною).*

Наприклад, циклічна група порядку n *може бути задана твірними і визначальними співвідношеннями як* $C_n = \langle a \mid a^n = e \rangle$*, а діедральна група* D_n *— як*

$$D_n = \langle a, b \mid a^2 = e,\ b^n = e,\ ab = b^{-1}a \rangle.$$

Задані в такий спосіб групи з'являються в багатьох розділах математики (напр., саме так з'являються фундаментальні групи в топології).

Для заданих у такий спосіб груп природно виникає проблема тотожності: як дізнатися, коли два слова в алфавіті $A \cup A^{-1}$ *задають той самий елемент групи, а коли — різні? Наприклад, для групи* C_n *ця проблема розв'язується легко:* $a^m = a^k$ *тоді й лише тоді, коли* $m \equiv k \pmod{n}$*. Неважко побудувати такий алгоритм для комутативних груп.*

Проблема полягала в тому, щоб знайти алгоритм, який був би придатний для всіх груп, заданих скінченною кількістю твірних і визначальних співвідношень. 1955 р. П. С. Новіков довів, що такого алгоритму не існує.

Найбільший розголос отримало доведення алгоритмічної нерозв'язності десятої проблеми Гільберта[43]. Останню крапку в цьому доведенні поставили 1970 р. Григорій Чудновський (на той час студент першого курсу механіко-математичного факультету Київського університету) і Юрій Матіясевич.

Ми розглянемо два підходи до формалізації поняття алгоритму — рекурсивні функції і машини Тюрінга.

5.2. Обчислювальні функції

Аналіз поняття алгоритму ми почнемо з поняття обчислювальної функції. Для алгоритмічних проблем типово, коли в умову задачі входить набір параметрів $(x_1, \ldots, x_n)$, які є цілими числами, а результатом також є певне ціле число y. Тобто йдеться про існування алгоритму для обчислення деякої функції $f : \mathbb{N}^n \to \mathbb{N}$, де $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ — множина натуральних чисел. Тому спочатку обмежимося лише такими функціями.

Сукупність усіх процесів, що підпадають під інтуїтивне поняття алгоритму, надзвичайно широка й неозора, тоді як клас обчислювальних функцій від натуральних аргументів значно вужчий. Пізніше ми побачимо, що обмеження такими функціями не є істотним: обчислення довільних функцій і навіть роботу довільних алгоритмів можна звести до обчислення функцій від натуральних аргументів.

Насамперед нас цікавитиме випадок $n = 1$. Кожну обчислювальну функцію $f : \mathbb{N} \to \mathbb{N}$ задають певним скінченним текстом у даному скінченному алфавіті. Множина всіх таких текстів — зліченна, у той час як множина всіх функцій $f : \mathbb{N} \to \mathbb{N}$ — незліченна. Звідси одразу випливає існування необчислювальних функцій.

Це міркування виглядає поки що не дуже переконливим: а чому ми маємо працювати лише з одним уточненням поняття алгоритму? Можливо, існує настільки багато різних "обчислювальних засобів", які можна точно описати, що для кожної функції $f : \mathbb{N} \to \mathbb{N}$ можна буде підібрати свій алгоритм, який обчислює цю функцію?

Як ми побачимо трохи пізніше, фундаментальним відкриттям теорії обчислювальності є твердження, що на останнє питання треба дати негативну відповідь.

[43] У знаменитому списку проблем Гільберта її формулювання найкоротше: *"Нехай дано діофантове рівняння з довільними невідомими і цілими числовими коефіцієнтами. Вказати спосіб, за допомогою якого можна після скінченної кількості кроків установити, чи має це рівняння розв'язок у цілих числах"*.

Ряд нетривіальних властивостей обчислювальних функцій можна встановити, навіть не маючи строгого означення таких функцій.

Теорема 5.1. *Для кожного розумного означення поняття обчислювальної функції $\mathbb{N} \to \mathbb{N}$ існує обчислювальна функція, яка не буде скрізь визначеною і яку принципово не можна довизначити до скрізь визначеної обчислювальної функції.*

Доведення. Як сказано вище, для кожного розумного означення поняття обчислювальної функції вони (точніше, алгоритми їхнього обчислення) мають задаватися скінченними текстами у певному скінченному алфавіті. Тому їх зліченна кількість і їх можна занумерувати: $f_0(n)$, $f_1(n)$, $f_2(n)$, ... Покладемо $g(n) = f_n(n) + 1$. Очевидно, що функція $g(n)$ є обчислювальною у тому ж сенсі, в якому є обчислювальними функції $f_0(n)$, $f_1(n)$, $f_2(n)$, ... Якби вона була скрізь визначеною або її можна було доповнити до скрізь визначеної, то після такого доповнення вона збігалася б із якоюсь f_k. Але g і f_k набувають різних значень у точці k. Отже, функція $g(n)$ не є скрізь визначеною і не може бути довизначена до скрізь визначеної обчислювальної функції. □

Теорема 5.2. *Для кожного розумного означення поняття обчислювальної функції $f : \mathbb{N} \to \mathbb{N}$ існує обчислювальна функція, яка не визначена в жодній точці.*

Доведення. Як і в попередньому доведенні, ми можемо занумерувати обчислювальні функції: $f_0(x)$, $f_1(x)$, $f_2(x)$, ... Припустимо, що кожна функція f_n визначена хоча б в одній точці. Оскільки алгоритми обчислення значень функцій працюють дискретно, кроками, то можна запустити такий діагональний процес: спочатку перший крок обчислення $f_0(0)$, потім наступний крок обчислення $f_0(0)$ і перші кроки обчислення $f_0(1)$ та $f_1(0)$, потім наступні кроки обчислення $f_0(0)$, $f_0(1)$, $f_1(0)$ та перші кроки обчислення $f_0(2)$, $f_1(1)$ і $f_2(0)$ і т. д. Якщо функція f_n визначена в точці k, то рано чи пізно ми значення $f_n(k)$ обчислимо.

Визначимо тепер функцію $h(n)$ таким чином: $h(n)$ — це перше значення аргументу x, при якому завершилось обчислення значення $f_n(x)$. Зокрема, f_n визначена в точці $h(n)$. Тоді функція

$$g(n) = f_n(h(n)) + 1$$

є скрізь визначеною обчислювальною функцією, яка, проте, не зустрічається у списку $f_0(x)$, $f_1(x)$, $f_2(x)$, ... Отже, припущення про визначеність кожної обчислювальної функції хоча б в одній точці приводить до суперечності. □

Зауваження. *Для конкретних підходів до поняття обчислювальності вказати обчислювальну ніде не визначену функцію зазвичай дуже легко. Зокрема і для рекурсивних функцій і машин Тюрінга ми це зробимо пізніше. Але з теореми 5.2 випливає, що такі функції виникатимуть при кожному підході до поняття обчислювальності.*

5.3. Рекурсивні функції

Переходимо до строгих означень. Числова функція $f : \mathbb{N}^n \to \mathbb{N}$, значення якої за різних значень параметрів $(x_1, \ldots, x_n)$ можна обчислювати за допомогою того самого алгоритму, називають ефективно *обчислювальною* (або *обчислювальною*) функцією.

У такому "означенні" поняття обчислювальної функції ще лишається інтуїтивним. Але аналіз самої ідеї детермінованого процесу обчислень приводить до уявлення про те, що такий процес можна розкласти на елементарні кроки із скінченного та фіксованого раз і назавжди списку. До того ж кожну обчислювальну функцію можна отримати за скінченну кількість кроків, кожен з яких полягає в застосуванні одного із цих елементарних кроків до раніше побудованих або найпростіших функцій.

Однією з реалізацій цієї ідеї стало поняття рекурсивної функції. Вперше клас рекурсивних функцій описав 1931 р. Гедель. Із зовсім інших міркувань прийшов до цього класу Черч 1936 р. Тоді ж Кліні ввів ширший клас частково обчислювальних функцій, названих частково рекурсивними. До вивчення цих класів функцій ми й переходимо.

Далі, досліджуючи обчислювальні функції, зручно вважати, що алгоритм для обчислення функції $f(\boldsymbol{x})$ починає працювати при будь-якому допустимому вході $\boldsymbol{x}$, а невизначеність функції $f(\boldsymbol{x})$ у точці $\boldsymbol{x}$ рівносильна тому, що під час входу $\boldsymbol{x}$ алгоритм працює нескінченно довго і не видає жодного результату.

5.3.1. Примітивно рекурсивні функції

Функції $f : \mathbb{N}^k \to \mathbb{N}$, про які йтиметься, можуть бути *частковими*, тобто визначеними не на всій множині $\mathbb{N}^k$.

Функції

$$S^1(x) := x + 1\,,$$
$$O^n(x_1, \ldots, x_n) := 0\,,$$
$$I^n_m(x_1, \ldots, x_n) := x_m$$

називають *найпростішими*. Очевидно, що всі вони є ефективно обчислювальними.

Розглянемо тепер три операції, за допомогою яких можна отримувати нові функції. Найпростішою з них є суперпозиція функцій, яку визначають так:

нехай $f_1(x_1,\dots,x_n),\dots,f_m(x_1,\dots,x_n)$ і $g(x_1,\dots,x_m)$ — (часткові) функції. *Суперпозицією* цих функцій (або *підстановкою* функцій f_1, …, f_m у функцію g) називають функцію

$$h(x_1,\dots,x_n)=g\big(f_1(x_1,\dots,x_n),\dots,f_m(x_1,\dots,x_n)\big).$$

Очевидно, що коли функції $f_1,\dots,f_m$ і g ефективно обчислювальні, то такою ж буде і функція $h(x_1,\dots,x_n)$.

Функції, які можна одержати з найпростіших за допомогою суперпозиції (можливо, застосованої кілька разів), називають *елементарними*.

Приклади. 1. *Для кожного додатного цілого числа k функції $f(x)=k$ і $g(x)=x+k$ є елементарними. Це випливає з рівностей*

$$f(x)=\underbrace{S^1(S^1(\cdots S^1}_{k\text{ разів}}(O^1(x))\cdots)),\quad g(x)=\underbrace{S^1(S^1(\cdots S^1}_{k\text{ разів}}(I^1_1(x))\cdots)).$$

2. *Функція $f(x)=x^2$ не є елементарною. Справді, із найпростіших функцій значення відповідного аргументу збільшує лише функція S^1. Але якщо у побудові $f(x)$ функція S^1 використовувалася k разів, то значення аргументу може збільшитися щонайбільше на k. У той же час різниця x^2-x може бути як завгодно великою.*

Клас елементарних функцій ще дуже бідний, щоб його вивчати детально. Тому розглянемо наступну операцію для побудови нових функцій.

Кажуть, що функцію $f(x_1,\dots,x_n,y)$ будують з функцій $g(x_1,\dots,x_n)$ і $h(x_1,\dots,x_n,y,z)$ за допомогою *примітивної рекурсії*, якщо вона визначається такими правилами:

$$\begin{aligned} f(x_1,\dots,x_n,0)&=g(x_1,\dots,x_n),\\ f(x_1,\dots,x_n,y+1)&=h\big(x_1,\dots,x_n,y,f(x_1,\dots,x_n,y)\big).\end{aligned}\tag{5.1}$$

При $n=0$ правила набувають вигляду

$$f(0)=a,\qquad f(y+1)=h(y,f(y)).\tag{5.2}$$

Важливо, що коли функції g і h ефективно обчислювальні, то примітивна рекурсія дає ефективну процедуру і для обчислення функції f. Зокрема, f буде визначеною в точці $(x_1, \ldots, x_n, m{+}1)$ тоді й лише тоді, коли визначеними будуть усі значення

$$a_0 = g(x_1, \ldots, x_n),\ a_1 = h(x_1, \ldots, x_n, 0, a_0),$$
$$a_2 = h(x_1, \ldots, x_n, 1, a_1),\ \ \ldots,\ a_{m+1} = h(x_1, \ldots, x_n, m, a_m).$$

Функції, які можна одержати з найпростіших за допомогою суперпозиції та примітивної рекурсії (можливо, застосованих кілька разів), називають *примітивно рекурсивними*. Зауважимо, що всі примітивно рекурсивні функції є *скрізь визначеними* й *ефективно обчислювальними*.

Зауваження. *Приклади задання функцій рекурсивними процедурами зустрічалися й раніше, зокрема, ще в школі. Але зазвичай на особливостях їх задання увага не акцентувалася. Найвідомішим прикладом рекурсивного задання функції є функція* $n! = (n-1)! \cdot n$. *Арифметичну та геометричну прогресії також визначають рекурсивно.*

Теорема 5.3. *Наведені функції є примітивно рекурсивними*:
(1) $f(x, y) = x + y$;
(2) $f(x, y) = xy$;
(3) $f(x, y) = x^y$ $(x^0 = 1)$;
(4) $sg\, x = \begin{cases} 0, & \text{якщо } x = 0; \\ 1, & \text{якщо } x > 0; \end{cases}$
(5) $\overline{sg}\, x = \begin{cases} 1, & \text{якщо } x = 0; \\ 0, & \text{якщо } x > 0; \end{cases}$
(6) $x \ominus y = \begin{cases} x - y, & \text{якщо } x \geq y; \\ 0, & \text{якщо } x < y; \end{cases}$
(7) $|x - y|$;
(8) $\max(x, y)$;
(9) $\min(x, y)$.

Доведення. (1) За $g(x)$ візьмемо функцію $I^1_1(x) = x$, а за $h(x, y, z)$ — функцію $S^1(I^3_3(x, y, z)) = z + 1$. Отримуємо

$$x + 0 = x = g(x), \quad x + (y + 1) = (x + y) + 1 = h(x, y, x + y).$$

(2) За $g(x)$ візьмемо функцію $O^1(x) = 0$, а за $h(x, y, z)$ — функцію $z + x$. Отримуємо

$$x \cdot 0 = 0 = g(x), \quad x \cdot (y + 1) = xy + x = h(x, y, xy).$$

(3) За $g(x)$ візьмемо функцію $S^1(O^1(x)) = 1$, а за $h(x, y, z)$ — функцію $z \cdot x$. Отримуємо $x^0 = 1 = g(x)$, $x^{y+1} = x^y \cdot x = h(x, y, x^y)$.

(4) У цьому випадку $n = 0$. Тому покладемо $\operatorname{sg} 0 = 0$ і розглянемо $h(x, z) = 1$. Тоді для всіх x матимемо $\operatorname{sg}(x + 1) = h(x, \operatorname{sg} x) = 1$, тобто для всіх $x > 0$ буде $\operatorname{sg} x = 1$.

(5) Доведення аналогічне попередньому, якщо покласти $\overline{\operatorname{sg}}\, 0 = 1$ і взяти $h(x, z) = 0$.

(6) Спочатку розглянемо функцію $x \ominus 1$. Її примітивна рекурсивність випливає з рівностей $0 \ominus 1 = 0$ і $(x + 1) \ominus 1 = x = h(x, x \ominus 1)$, де $h(x, z) = x$. Тепер для доведення примітивної рекурсивності функції $x \ominus y$ можна взяти $g(x) = x$ і $h(x, y, z) = z \ominus 1$. Справді,

$$x \ominus 0 = x = g(x) \quad \text{і} \quad x \ominus (y + 1) = (x \ominus y) \ominus 1 = h(x, y, x \ominus y).$$

(7) $|x - y| = (x \ominus y) + (y \ominus x)$.
(8) $\max(x, y) = x + (y \ominus x)$.
(9) $\min(x, y) = \max(x, y) \ominus |x - y|$. □

Теорема 5.4 ($\sum$-теорема)**.** *Нехай функції*

$$u(x_1, \ldots, x_n), \quad v(x_1, \ldots, x_n) \quad \textit{і} \quad g(x_1, \ldots, x_n)$$

— примітивно рекурсивні. Тоді функція

$$\sum_{i=u(x_1,\ldots,x_n)}^{v(x_1,\ldots,x_n)} g(x_1, \ldots, x_{n-1}, i) \tag{5.3}$$

також буде примітивно рекурсивною.

Доведення. Покажемо спочатку, що примітивно рекурсивною буде функція

$$f(x_1, \ldots, x_{n-1}, y) = \sum_{i=0}^{y} g(x_1, \ldots, x_{n-1}, i).$$

Справді, з рівностей

$$f(x_1, \ldots, x_{n-1}, 0) = g(x_1, \ldots, x_{n-1}, 0),$$
$$f(x_1, \ldots, x_{n-1}, y + 1) = f(x_1, \ldots, x_{n-1}, y) + g(x_1, \ldots, x_{n-1}, y + 1),$$

випливає, що функцію $f(x_1, \ldots, x_{n-1}, y)$ одержують за допомогою примітивної рекурсії з примітивно рекурсивних функцій $g(x_1, \ldots, x_{n-1}, 0)$ і

$$h(x_1, \ldots, x_{n-1}, y, z) = z + g(x_1, \ldots, x_{n-1}, y + 1).$$

А тому вона також є примітивно рекурсивною.

Далі з рівності

$$\sum_{i=y}^{z} g(x_1,\dots,x_{n-1},i) = \sum_{i=0}^{z} g(x_1,\dots,x_{n-1},i) \ominus \sum_{i=0}^{y} g(x_1,\dots,x_{n-1},i) + \\ + g(x_1,\dots,x_{n-1},y)\cdot \overline{\mathrm{sg}}(y \ominus z)$$

випливає, що примітивно рекурсивною буде функція

$$p(x_1,\dots,x_{n-1},y,z) = \sum_{i=y}^{z} g(x_1,\dots,x_{n-1},i).$$

Нарешті, функцію (5.3) одержують із функції $p(x_1,\dots,x_{n-1},y,z)$ за допомогою підстановки $y = u(x_1,\dots,x_n)$, $z = v(x_1,\dots,x_n)$. □

Наслідок 5.1. *Кожна з функцій* $a)$ $\left[\frac{x}{y}\right]$ (*вважаємо, що* $\left[\frac{x}{0}\right] = x$), $b)$ $[\sqrt{x}]$ *є примітивно рекурсивною.*

Доведення. a) $\left[\frac{x}{y}\right] = \sum_{i=1}^{x} \overline{\mathrm{sg}}\,(iy \ominus x)$; b) $[\sqrt{x}] = \sum_{i=1}^{x} \overline{\mathrm{sg}}\,(i^2 \ominus x)$. □

Вправа 5.1. *Доведіть, що функція* $\mathrm{rest}\,(x,y)$ *— остача від ділення* x *на* y (*вважаємо, що* $\mathrm{rest}\,(x,0) = x$) *є примітивно рекурсивною.*

Використовуючи наслідок 5.1, аналогічно теоремі 5.4 доводять теорему 5.5.

Теорема 5.5 ($\prod$-теорема)**.** *Нехай функції*

$$u(x_1,\dots,x_n), \quad v(x_1,\dots,x_n) \quad \textit{і} \quad g(x_1,\dots,x_n)$$

— примітивно рекурсивні. Тоді функція

$$\prod_{i=u(x_1,\dots,x_n)}^{v(x_1,\dots,x_n)} g(x_1,\dots,x_{n-1},i)$$

також буде примітивно рекурсивною.

Нехай $P_i(x_1,\dots,x_n)$ $(i = 1,\dots,k)$ — набір предикатів, жодні два з яких не бувають істинними одночасно, $f_1(x_1,\dots,x_n)$, $\dots$, $f_k(x_1,\dots,x_n)$, $f_{k+1}(x_1,\dots,x_n)$ — (часткові) функції. Кажуть, що функцію $f(x_1,\dots,x_n)$ *задають частинами*, якщо вона задається правилом

$$f(x_1,\dots,x_n) = \begin{cases} f_1(x_1,\dots,x_n), & \text{якщо виконується } P_1(x_1,\dots,x_n); \\ \cdots\cdots\cdots & \cdots\cdots\cdots \\ f_k(x_1,\dots,x_n), & \text{якщо виконується } P_k(x_1,\dots,x_n); \\ f_{k+1}(x_1,\dots,x_n) & \text{в інших випадках.} \end{cases} \tag{5.4}$$

Теорема 5.6 (про функцію, задану частинами)**.** *Якщо у правилі* (5.4) *всі функції* $f_1(x_1, \ldots, x_n)$, $\ldots$, $f_k(x_1, \ldots, x_n)$, $f_{k+1}(x_1, \ldots, x_n)$ *примітивно рекурсивні, а всі предикати* P_i $(i = 1, \ldots, k)$ *задано рівностями* $g_i(x_1, \ldots, x_n) = 0$, *де* g_i *— примітивно рекурсивна функція, то функція* $f(x_1, \ldots, x_n)$ *— також примітивно рекурсивна.*

Доведення. Функцію $f(x_1, \ldots, x_n)$ можна записати у вигляді

$$f = f_1 \cdot \overline{\mathrm{sg}}\, g_1 + \cdots + f_k \cdot \overline{\mathrm{sg}}\, g_k + f_{k+1} \cdot \mathrm{sg}\,(g_1 \cdots g_k). \qquad \square$$

5.3.2. Частково рекурсивні функції

Нехай $f(x_1, \ldots, x_{n+1})$ — якась (часткова) обчислювальна функція арності $n + 1$, причому невизначеність функції f у точці $(a_1, \ldots, a_{n+1})$ рівносильна тому, що алгоритм обчислення функції f на вході $(a_1, \ldots, a_{n+1})$ працює нескінченно довго. Зафіксуємо значення $x_1, \ldots, x_n$ перших n аргументів і розглянемо рівняння

$$f(x_1, \ldots, x_n, y) = 0. \tag{5.5}$$

Розв'язок цього рівняння шукатимемо, послідовно обчислюючи значення

$$f(x_1, \ldots, x_n, 0),\ f(x_1, \ldots, x_n, 1),\ f(x_1, \ldots, x_n, 2),\ \ldots \tag{5.6}$$

Очевидно, що такий пошук триватиме нескінченно довго (іншими словами, знайти розв'язок у такий спосіб не вдасться) лише в одному з трьох випадків:

1) значення $f(x_1, \ldots, x_n, 0)$ не визначене;

2) значення $f(x_1, \ldots, x_n, 0)$, $\ldots$, $f(x_1, \ldots, x_n, k)$ визначені, але жодне з них не дорівнює 0, а значення $f(x_1, \ldots, x_n, k + 1)$ не визначене;

3) усі значення (5.6) визначені, але жодне з них не дорівнює 0.

В усіх інших випадках ми рано чи пізно знайдемо таке значення a, що $f(x_1, \ldots, x_n, a) = 0$.

Через

$$\mu_y\big(f(x_1, \ldots, x_n, y) = 0\big) \tag{5.7}$$

позначимо функцію, значенням якої на наборі $(x_1, \ldots, x_n)$ є найменший розв'язок a рівняння (5.5), якщо цей розв'язок можна знайти описаним вище способом, і яка не визначена на наборі $(x_1, \ldots, x_n)$ у протилежному разі. Будемо говорити, що функція $g(x_1, \ldots, x_n) = \mu_y\big(f(x_1, \ldots, x_n, y) = 0\big)$ одержана з $f(x_1, \ldots, x_n)$ за допомогою *мінімізації* за змінною y. Оператор мінімізації ще називають μ-*оператором*.

Приклади. 1. *Звичайне віднімання можна подати у вигляді* $x - y = \mu_z(|x - (y + z)| = 0)$. *Зауважимо, що функція* $x - y$ *буде визначеною тоді й лише тоді, коли* $x \geq y$.

2. *Функція* $x/2 = \mu_y(x - 2y = 0)$ *буде визначеною тоді й лише тоді, коли* x *є парним числом.*

Важливо підкреслити, що операція мінімізації знаходить найменший розв'язок рівняння (5.5) не завжди. Наприклад, рівняння $y - (x + 1) = 0$ для кожного x має розв'язок $y = x + 1$. У той же час оператор мінімізації $\mu_y(y - (x + 1) = 0)$ дає ніде не визначену функцію, бо для кожного x уже перше значення $0 - (x + 1)$ з послідовності (5.6) є невизначеним.

Функції, які можна одержати з найпростіших за допомогою суперпозиції, примітивної рекурсії та мінімізації, називають *частково рекурсивними*. Скрізь визначені частково рекурсивні функції називаються *загальнорекурсивними*. Очевидно, що такими є всі примітивно рекурсивні функції. Зворотне твердження є хибним, однак приклади загальнорекурсивних функцій, які не є примітивно рекурсивними, будувати досить складно. У зв'язку із цим варто звернути увагу на теореми 5.7 і 5.8.

Крім виразів вигляду (5.7) можна також використовувати вирази

$$\mu_y\big(f(x_1, \ldots, x_n, y) = g(x_1, \ldots, x_n, y)\big),$$
$$\mu_y\big(f(x_1, \ldots, x_n, y) \neq g(x_1, \ldots, x_n, y)\big),$$
$$\mu_y\big(f(x_1, \ldots, x_n, y) > g(x_1, \ldots, x_n, y)\big)$$

і подібні. Вважаємо, що їхні значення збігаються зі значеннями виразів

$$\mu_y\big(|f(x_1, \ldots, x_n, y) - g(x_1, \ldots, x_n, y)| = 0\big),$$
$$\mu_y\big(\overline{\text{sg}}\,|f(x_1, \ldots, x_n, y) - g(x_1, \ldots, x_n, y| = 0\big),$$
$$\mu_y\big(\overline{\text{sg}}\,(f(x_1, \ldots, x_n, y) \ominus g(x_1, \ldots, x_n, y)) = 0\big)$$

відповідно.

Для одномісної функції $f(x)$ функцію $\mu_y(f(y) = x)$ часто називають *оберненою* до $f(x)$ і позначають $f^{-1}(x)$.

Теорема 5.7 (про обмежену мінімізацію)**.** *Нехай* $g(x_1,\ldots,x_n, y)$ *і* $\alpha(x_1,\ldots,x_n)$ *— такі примітивно рекурсивні функції, що для довільних* $x_1, \ldots, x_n$ *рівняння* $g(x_1, \ldots, x_n, y) = 0$ *має хоча б один розв'язок, який не перевищує числа* $\alpha(x_1, \ldots, x_n)$. *Тоді функція*

$$f(x_1, \ldots, x_n) = \mu_y(g(x_1, \ldots, x_n, y) = 0)$$

є примітивно рекурсивною.

Доведення. Нехай

$$h(x_1, \ldots, x_n, z) = \prod_{i=0}^{z} g(x_1, \ldots, x_n, i)\,.$$

Тоді

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{\alpha(x_1,\ldots,x_n)} \operatorname{sg} h(x_1, \ldots, x_n, i)\,.$$

□

Зауваження. *Якщо не вимагати обмеженості згори розв'язків рівняння* $g(x_1, \ldots, x_n, y) = 0$ *певною функцією* $\alpha(x_1, \ldots, x_n)$*, то теорема* 5.7 *перестає бути правильною.*

Приклад. *Раніше вже доведено (наслідок* 5.1, п.b*), що функція* $[\sqrt{x}]$ *є примітивно рекурсивною. Її примітивна рекурсивність випливає також із рівності* $[\sqrt{x}] = \mu_z\big(sg\,((z+1)^2 \ominus x) = 1\big)$ *і теореми* 5.7*, оскільки* $[\sqrt{x}] \le x$*.*

У наступній теоремі доводиться примітивна рекурсивність кількох важливих теоретико-числових функцій.

Теорема 5.8. *Наведені функції є примітивно рекурсивними*:

(1) *Кількість* $\tau(x)$ *дільників числа* x*.*

(2) *Характеристична функція простих чисел*

$$\chi_p(x) = \begin{cases} 1, & \text{якщо } x \text{ — просте число;} \\ 0, & \text{в іншому разі.} \end{cases}$$

(3) *Кількість* $\pi(x)$ *простих чисел, що не перевищують числа* x*.*

(4) $p(n) = \mu_x\big(|\pi(x) - (n+1)| = 0\big)$ *— n-те просте число* ($p(0) = 2$, $p(1) = 3$, ...*.*

(5) $f(k, x)$ *— показник, з яким k-те просте число $p(k)$ входить у канонічний розклад числа x.*

Доведення. (1) $\tau(x) = \sum_{i=0}^{x} \overline{\operatorname{sg}}\,\operatorname{rest}\,(x, i)$.

(2) $\chi_p(x) = \overline{\operatorname{sg}}\,|\tau(x) - 2|$.

(3) $\pi(x) = \sum_{i=0}^{x} \operatorname{sg} \chi_p(i)$.

(4) Щоб застосувати до функції $p(n)$ теорему 5.7, досить показати, що $p(n)$ обмежена згори деякою примітивно рекурсивною функцією $\alpha(n)$. Зокрема, можна взяти $\alpha(n) = 2^{2^n}$. Нерівність $p(n) \le 2^{2^n}$ легко доводити за допомогою індукції. База індукції очевидна, оскільки

$$p(0) = 2 \le 2^{2^0} = 2 \quad \text{і} \quad p(1) = 3 < 2^{2^1} = 4.$$

Індукційний крок легко випливає з класичного доведення теореми Евкліда про нескінченність множини простих чисел:

$$p(n+1) \le p(0)p(1)\cdots p(n)+1 \le 2^{2^0}\cdot 2^{2^1}\cdots 2^{2^n}+1 = \\ = 2^{2^0+2^1+\cdots 2^n}+1 = 2^{2^{n+1}-1}+1 < 2^{2^{n+1}}.$$

Зауважимо, що функцію $p(n)$ можна задати й інакше:

$$p(n) = \mu_y\Big(\sum_{i=0}^{y}\chi_p(i) = n\Big).$$

(5) Оскільки $f(k,x) < x$, то примітивна рекурсивність функції $f(k,x)$ випливає з примітивної рекурсивності функції rest, рівності

$$f(k,x) = \mu_y(\text{rest}(x, p^y(k)) \ne 0) \ominus 1$$

і теореми 5.7.

Зауважимо, що функцію $f(k,x)$ можна задати й іншим чином: $f(k,x) = \sum_{i=1}^{x}\overline{\text{sg}}\,\text{rest}(x,p^i(k))$. □

5.3.3. Деякі зауваження

1. Питання про ефективну обчислювальність тієї чи іншої функції може бути дуже нетривіальним. Наприклад, для функції $f:\mathbb{N}\to\mathbb{N}$

$$f(n) = \begin{cases} n, & \text{якщо для } n = \overline{a_1\ldots a_k} \text{ у десятковому розкладі} \\ & \text{числа } \pi \text{ зустрічається послідовність цифр } a_1\ldots a_k; \\ 0, & \text{у протилежному разі,} \end{cases}$$

питання про її ефективну обчислювальність лишається нез'ясованим (і нема жодних гарантій, що функція виявиться ефективно обчислювальною, хоча жодних сумнівів у коректності її визначення також нема).

2. Більше того, не для кожної функції, про яку відомо, що вона обчислювальна, можна ефективно вказати відповідну обчислювальну процедуру. Типовий приклад:

$$f(n) = \begin{cases} 1, & \text{якщо гіпотеза Рімана справедлива;} \\ 0, & \text{у протилежному разі.} \end{cases}$$

Очевидно, що ця функція обчислювальна. Однак ефективну процедуру для обчислення її значень у наш час вказати не можна.

5.4. Машини Тюрінга

Поняття частково рекурсивної функції є одним із можливих уточнень поняття алгоритму. Однак це уточнення не є прямим, бо поняття обчислювальної функції є вторинним стосовно поняття алгоритму. Виникає природне питання: а чи не можна уточнити безпосередньо саме поняття алгоритму?

Вперше це зробили незалежно американський математик Пост (1936 р. у "Журналі символічної логіки" вийшла його стаття "Фінітні комбінаторні процеси, формулювання I") й англійський математик Тюрінг (який 1937 р. у "Працях Лондонського математичного товариства" опублікував роботу "Про обчислення числа із застосуванням до проблеми розв'язності").

Основна ідея обох робіт однакова: алгоритмічні процеси — це такі процеси, які може виконувати відповідним чином влаштована "машина". Для цього кожен із них описав у точних математичних термінах певний досить вузький клас машин (Тюрінг це зробив явно, Пост — неявно, у нього сам термін "машина" відсутній), щодо яких не було жодних сумнівів: те, що можуть робити ці машини, є алгоритмами. Але з іншого боку виявилося, що на цих машинах можна реалізувати або імітувати *всі* відомі математикам алгоритмічні процеси. До того ж з'ясувалося, що клас функцій, які можна обчислювати на цих машинах, повністю збігається з класом усіх частково рекурсивних функцій. Тому запропоновано вважати реалізовні на таких машинах алгоритми формальними представниками алгоритмів взагалі.

Згодом додали ще кілька десятків інших уточнень поняття алгоритму. Але для кожної нової формалізації цього поняття вдавалося довести її еквівалентність машинам Тюрінга. Ця обставина, історична першість, психологічна прийнятність (машини Тюрінга копіюють у суттєвих рисах роботу людини, яка виконує обчислення за заданою програмою), а головне — зручність для застосувань (вони дозволяють вводити і вивчати різні кількісні характеристики алгоритмів, зокрема і порівнювати їхню складність), привели до того, що машини Тюрінга залишаються найпопулярнішою математичною моделлю алгоритмічних процесів.

Машина Тюрінга (коротко — МТ) складається з

(1) нескінченної в обидва боки і розбитої на клітинки *стрічки*, причому кожна клітинка містить рівно один символ із так званого *зовнішнього* алфавіту;

(2) *головки* зчитування-запису, яка в кожний момент часу оглядає одну клітинку стрічки і може зчитувати символ із цієї клітинки або записувати туди інший символ;

(3) *керуючого пристрою*, який містить *програму* роботи машини Тюрінга і завжди міститься в одному зі скінченної множини *станів*.

Формально МТ задають трійкою (A, Q, δ), де $A = \{a_0, a_1, \ldots, a_n\}$ — скінченний *зовнішній* алфавіт (в якому виділено так званий *порожній* символ a_0, який зазвичай позначають як Λ або 0), $Q = \{q_1, \ldots, q_m, q_{fin}\}$ — скінченний *внутрішній* алфавіт (або алфавіт *станів*, в якому є два виділених стани — q_1 і q_{fin} — які називають відповідно *початковим* і *заключним*), а δ — це функція із $A \times (Q \setminus \{q_{fin}\})$ в $A \times Q \times \{L, R, S\}$, яку називають *функцією переходів* (або *програмою* МТ). Програма МТ може бути не скрізь визначеною.

Зовнішній A і внутрішній Q алфавіти можуть бути довільними, лише не повинні мати спільних символів.

Рівність $\delta(a_j, q_i) = (a_k, q_l, W)$ часто записують у вигляді $a_j q_i \to a_k q_l W$ і називають *командою* МТ. Програму МТ можна задавати таблицею команд або списком команд (відповідні приклади наведено нижче). Множину всіх команд МТ T також називають *програмою* машини T і позначають $\Pi(T)$.

Робота МТ складається з послідовності кроків (або тактів). Перед початком роботи МТ майже всі (тобто за винятком лише скінченної кількості) клітинки стрічки повинні містити порожній символ Λ. Записане на стрічці слово $x = a_{i_1} a_{i_2} \ldots a_{i_p}$ називають *входом* МТ, якщо всі символи ліворуч від a_{i_1} і праворуч від a_{i_p} — порожні. Машина починає роботу в стані q_1, причому її головка оглядає в цей час символ a_{i_1}. Подамо один крок роботи (або *такт*) МТ: якщо МТ в стані q_i, головка оглядає на стрічці символ a_j, а $\delta(a_j, q_i) = (a_k, q_l, W)$, то на місце символу a_j головка записує символ a_k, машина переходить у стан q_l і головка зміщується або на одну клітинку ліворуч (якщо $W = L$), або на одну клітинку праворуч (якщо $W = R$, або лишається на місці (якщо $W = S$). Якщо $q_l = q_{fin}$, то машина зупиняється (так звана *результативна зупинка*), у протилежному разі МТ переходить до наступного кроку. Записане на стрічці після зупинки МТ слово $y = a_{j_1} a_{j_2} \cdots a_{j_r}$, де a_{j_1} — крайній лівий, а a_{j_r} — крайній правий непорожні символи, називають *виходом* МТ. Кількість кроків роботи МТ до результативної зупинки називають *часом роботи* МТ на вході x. Якщо результативна зупинка на вході x ніколи не настає, то кажуть, що вихід МТ на цьому вході не визначений.

Позначимо множину всіх слів над алфавітом A через A^*. Тоді можна сказати, що МТ із зовнішнім алфавітом A переробляє слова над алфавітом A (можливо, не всі) у слова над цим же алфавітом, тобто обчислює деяку словарну функцію $A^* \to A^*$ (можливо, не скрізь визначену).

Далі нам буде зручно слово довжини k, яке містить k букв a, записувати коротко як a^k. Зокрема, натуральне число k в алфавіті $A = \{0, 1\}$ записуватимемо як 1^{k+1} ($+1$ в показнику потрібне, щоб не плутати натуральне число 0 із порожнім символом 0).

Нехай на стрічці записане слово $a_{j_1}a_{j_2}\cdots a_{j_m}$, причому ліворуч від a_{j_1} і праворуч від a_{j_m} є лише порожні символи, а МТ перебуває у стані q_i й оглядає клітинку із символом a_{j_k}, $1\le k\le m$. Слово вигляду $a_{j_1}\cdots a_{j_{k-1}}q_ia_{j_k}\cdots a_{j_m}$ в алфавіті $A\cup Q$ (воно містить рівно один символ із Q) називають *конфігурацією* (або *машинним словом*) МТ. Змістовно конфігурація описує той момент роботи МТ, коли вона в стані q_i оглядає на стрічці символ a_{j_k}.

Конфігурацію вигляду $q_1a_{i_1}\dots a_{i_n}$ називають *початковою*, а вигляду $a_{j_1}\dots a_{j_k}q_{fin}a_{j_{k+1}}\dots a_{j_m}$ — *фінальною* (або *заключною*).

Кажуть, що команда $m = a_jq_r \to a_tq_lW$ *застосовна* до конфігурації $K = a_{j_1}\cdots q_ia_{j_k}\cdots a_{j_m}$, якщо $r = i$ та $j = j_k$. У протилежному разі команда m є *незастосовною* до конфігурації K.

Якщо команда m застосовна до конфігурації K, то *результатом* її застосування є конфігурація

$$K' = \begin{cases} a_{j_1}\cdots q_la_{j_{k-1}}a_ta_{j_{k+1}}\cdots a_{j_m}, & \text{якщо } W = L;\\ a_{j_1}\cdots a_{j_{k-1}}a_tq_la_{j_{k+1}}\cdots a_{j_m}, & \text{якщо } W = R;\\ a_{j_1}\cdots a_{j_{k-1}}q_la_ta_{j_{k+1}}\cdots a_{j_m}, & \text{якщо } W = S. \end{cases}$$

Перехід від K до K' у разі застосовності до K команди m позначатимемо так: $m : K \rightsquigarrow K'$.

Кажуть, що машина T *переводить* конфігурацію K_1 у конфігурацію K_2, якщо існує команда $m \in \Pi(T)$, яка застосовна до конфігурації K_1 і $m : K_1 \rightsquigarrow K_2$. Позначаємо так: $T : K_1 \rightsquigarrow K_2$.

Протоколом (роботи) машини Тюрінга T називають скінченну або нескінченну послідовність $K_0,\ K_1,\ \dots,\ K_n,\ \dots$ конфігурацій, яка задовольняє такі умови:

a) K_0 — початкова конфігурація;

b) якщо K_i не є останньою конфігурацією протоколу, то $T : K_i \rightsquigarrow K_{i+1}$;

c) якщо протокол скінченний, то остання конфігурація має бути фінальною.

Говорять, що машина Тюрінга T визначена на слові $a_{i_1}\dots a_{i_n}$, якщо для T існує скінченний протокол, який починається з конфігурації $K_0 = q_1a_{i_1}\dots a_{i_n}$ і закінчується певною фінальною конфігурацією K_t. Якщо

$$K_t = a_{j_1}\dots a_{j_k}q_{fin}a_{j_{k+1}}\dots a_{j_m},$$

то пишемо $T(a_{i_1}\dots a_{i_n}) = a_{j_1}\dots a_{j_m}$ і кажемо, що слово $a_{j_1}\dots a_{j_m}$ є результатом роботи машини T на слові $a_{i_1}\dots a_{i_n}$. Оскільки для даних МТ T і конфігурації K_0 існує не більше одного протоколу роботи, який починається з K_0, то результат обчислень $T(a_{i_1}\dots a_{i_n})$, якщо він існує, визначений однозначно.

Машина Тюрінга T може бути невизначеною на слові $a_{i_1} \dots a_{i_n}$ з двох причин:

— протокол роботи, який починається з конфігурації $K_0 = q_1 a_{i_1} \dots a_{i_n}$, є нескінченним і не приводить до заключної конфігурації;

— протоколу роботи, який починається з K_0, просто не існує (через те, що програма МТ T може бути не скрізь визначеною, то у процесі побудови такого протоколу може зустрітися конфігурація, яка не є фінальною, але до неї не застосовна жодна команда).

Зауваження. *Машина Тюрінга є ідеалізованою обчислювальною моделлю. Нас не цікавить реальна будова ні керуючого пристрою, ні головки зчитування-запису, ні стрічки. До того ж реальна стрічка не може бути нескінченною. Нас цікавить лише структура тієї послідовності машинних слів, яка виникає під час роботи машини Тюрінга. З математичного погляду машина Тюрінга є просто певним алгоритмом для перероблення машинних слів.*

Нехай A — скінченний алфавіт. Функцію $f : A^* \to A^*$ (можливо, не скрізь визначену) називають *обчислювальною* (*за Тюрінгом*), якщо існує МТ T із зовнішнім алфавітом A (можливо, поповненим порожнім символом Λ), яка породжує цю функцію. Іншими словами,

$T(a_{i_1} \dots a_{i_n}) = a_{j_1} \dots a_{j_k}$ *тоді й лише тоді, коли* $f(a_{i_1} \dots a_{i_n}) = a_{j_1} \dots a_{j_k}$.

Числову функцію $f(x_1, x_2, \dots, x_n) : \mathbb{N}^n \to \mathbb{N}$ називатимемо *обчислювальною за Тюрінгом*, якщо існує МТ із зовнішнім алфавітом $A = \{0, 1\}$, яка кожне слово вигляду

$$01^{x_1+1}01^{x_2+1}0 \dots 01^{x_n+1}0$$

переробляє на слово

$$01^{f(x_1,x_2,\dots,x_n)+1}.$$

Приклади. 1. МТ *із зовнішнім алфавітом* $A = \{0, 1\}$, *яка обчислює функцію* $x + y$, *тобто кожне слово вигляду* $01^{x+1}01^{y+1}0$ *переробляє на слово* $01^{x+y+1}0$:

$0q_1 \to 0q_1R$ — *перехід до початку* x;
$1q_1 \to 1q_2S$ — *початок* x;
$1q_2 \to 1q_2R$ — *проходження через* x;
$0q_2 \to 1q_3R$ — *заповнення проміжку між* x *та* y;
$1q_3 \to 1q_3R$ — *проходження через* y;
$0q_3 \to 0q_4L$ — *кінець* y;
$1q_4 \to 0q_5L$;

$1q_5 \to 0q_6L$ — *витирання зайвих одиниць*;
$1q_6 \to 1q_6L$ — *повернення до початку*;
$0q_6 \to 0q_{fin}S$ — *зупинка*.

2. МТ *із зовнішнім алфавітом* $A = \{0, 1\}$*, яка виконує дублювання, тобто кожне слово вигляду* 01^x0 *(*$x > 0$*) переробляє на слово* 01^x01^x0:
$0q_1 \to 0q_1R$ — *шукаємо початок слова* 1^x;
$1q_1 \to 0q_2R$ — *відмічаємо місце букви, яку дублюватимемо*;
$1q_2 \to 1q_2R$ — *шукаємо кінець слова* 1^x;
$0q_2 \to 0q_3R$ — *переходимо до копії*;
$1q_3 \to 1q_3R$ — *шукаємо кінець копії*;
$0q_3 \to 1q_4L$ — *додаємо до копії ще один символ* 1;
$1q_4 \to 1q_4L$;
$0q_4 \to 0q_5L$;
$1q_5 \to 1q_6L$ — *шукаємо у слові* 1^x *символ*, який дублювали;
$0q_6 \to 1q_1R$ — *відновлюємо продубльований символ*;
$1q_6 \to 1q_6R$ — *переходимо до наступного символу*;
$0q_5 \to 1q_7L$ — *продубльований символ є останнім у слові* 1^x*. Відновлюємо його і йдемо на початок слова, щоб завершити роботу*:
$1q_7 \to 1q_7L$;
$0q_7 \to 0q_{fin}S$.

3. МТ *із зовнішнім алфавітом* $A = \{0, 1\}$*, яка переставляє два аргументи, тобто кожне слово вигляду* 01^x01^y0 $(x, y > 0)$ *переробляє на слово* 01^y01^x0:
$0q_1 \to 0q_1R$ — *шукаємо початок першого слова*;
$1q_1 \to 1q_2R$;
$1q_2 \to 1q_2R$ — *шукаємо кінець першого слова*;
$0q_2 \to 1q_3L$;
$1q_3 \to 0q_4R$ — *відкидаємо від першого слова останню одиницю*;
$1q_4 \to 1q_4R$ — *шукаємо кінець другого слова*;
$0q_4 \to 0q_5L$;
$1q_5 \to 1q_6L$;
$1q_6 \to 0q_7L$ — *друге слово зміщуємо на одну позицію ліворуч, відкидаємо від нього останню одиницю і фіксуємо одиницю, яку забрали від першого слова*;
$1q_7 \to 1q_7L$;
$0q_7 \to 1q_8L$;
$1q_8 \to 0q_9R$ — *відкидаємо від першого слова останню одиницю*;
$1q_9 \to 1q_9R$;

$0q_9 \to 1q_{10}L$;

$1q_{10} \to 0q_7L$ — *друге слово зміщуємо на одну позицію ліворуч, дописуємо до третього слова одну одиницю ліворуч і йдемо на повторення циклу з останніх* 6 *команд*;

$0q_8 \to 0q_{fin}S$.

Зауваження. *Звертаємо увагу, що побудована МТ під час перестановки аргументів не виходить за межі початкового слова* 01^x01^y0.

Вправа 5.2. *Побудуйте* МТ *із зовнішнім алфавітом* $A = \{0, 1\}$*, яка з набору даних виділяє першу компоненту* (*тобто кожне слово, що має вигляд* $01^x01^y0 \ldots 01^z0$ $(x, y, \ldots, z > 0)$ *переробляє на слово* 01^x0.

Вправа 5.3. *Побудуйте* МТ *із зовнішнім алфавітом* $A = \{0, 1\}$*, яка перевіряє рівність двох чисел, тобто обчислює функцію*

$$f(x, y) = \begin{cases} 1, & \text{якщо } x = y; \\ 0, & \text{якщо } x \neq y. \end{cases}$$

Спільними рисами між сучасними комп'ютерами та машинами Тюрінга є:

1. Організація пам'яті: виділяють "атомарні" (тобто далі неподільні) носії інформації (у машинах Тюрінга ними є клітинки стрічки); кожен такий "атомарний" носій може перебувати лише в одному зі скінченного числа станів; уся інформація, що зберігається в даний момент у машині, визначається станами "атомарних" носіїв у цей момент.

2. Робота як послідовність тактів.

3. Вказується певний обмежений набір елементарних дій; за один такт може бути виконана лише одна дія із цього набору.

4. Програма: набір інструкцій, який вказує, які саме елементарні дії і в якому порядку мають бути виконані.

Відмінності між комп'ютерами та машинами Тюрінга:

1. В комп'ютері пам'ять (внутрішня) має обмежений об'єм, а стрічка машини Тюрінга нескінченна.

2. У машинах Тюрінга пам'ять влаштована "лінійно": на наступному такті можна перейти лише до однієї з 2 сусідніх клітинок, а в комп'ютерах у будь-який момент є доступ до будь-якої комірки пам'яті (правда, це стосується обмеженої оперативної пам'яті; зовнішня пам'ять комп'ютера влаштована більш лінійно).

3. Машини Тюрінга можна порівнювати з аналоговими комп'ютерами (в яких алгоритм обчислення функції "запаяно в залізо"). А сучасні цифрові

комп'ютери, в яких на вхід подають як текст програми алгоритму, так і дані для його роботи, фактично є відповідниками так званих *універсальних* МТ (якщо на вхід такої машини подати програму МТ T і слово w, то на виході отримаємо $T(w)$).

Зауважимо, що коли Тюрінг (і Пост) придумали свої машини, комп'ютерів ще не було. Перші комп'ютери почали з'являтися лише в 40-х рр. минулого століття. Водночас творець сучасної архітектури комп'ютерів фон Нойман роботи Тюрінга і Поста знав дуже добре.

5.5. Еквівалентність різних формалізацій

Конкретні алгоритми так чи інакше пов'язані з перетворенням текстів. Це може бути в явній формі, як у випадку машин Тюрінга (які переробляють слова, подані на вхід, на слова, отримані на виході), так і у прихованій (алгоритми для обчислення числових функцій насправді оперують не із числами, а з їхніми *зображеннями*, тобто записами цих чисел у вигляді слів у певних алфавітах).

Тому теорія обчислювальності, яка спирається на поняття рекурсивної функції, має також встановити зв'язок між текстами і числами, щоб зводити операції над текстами до операцій над числами. Один із підходів до встановлення такого зв'язку — *нумерація* текстів, після чого операції над текстами переходять в операції над номерами цих текстів. Принциповим для теорії обчислювальності виявився той факт, що *за будь-якої природної нумерації текстів усі елементарні операції над текстами, які можна покласти в основу процесів алгоритмічного перероблення тестів, перетворюються на рекурсивні функції.*

5.5.1. Позиційна і ґеделівська нумерації

Ми розглянемо два приклади нумерації текстів (перший із них має лише ілюстративне значення, а другий знадобиться далі).

А. *Позиційна нумерація*. На тексти в алфавіті $A = \{a_0, a_1, \ldots, a_{p-1}\}$ можна дивитися як на числа, записані в позиційній системі числення з основою p. Тоді номером слова $a_{i_0}a_{i_1}\ldots a_{i_k}$ буде число $i_0 + i_1 \cdot p + \cdots + i_k \cdot p^k$.

Ця нумерація має два істотні недоліки. По-перше, алфавіт A має бути скінченним, і, по-друге (і що значно важливіше), при звуженні або розширенні алфавіту номери всіх слів дуже змінюються. Цих недоліків позбавлена наступна нумерація.

В. *Ґеделівська нумерація*. Нехай $A = \{a_1, \ldots, a_k, \ldots\}$ — деякий алфавіт (можливо, нескінченний). Випишемо прості числа в їхньому природному порядку: $p_0 = 2, p_1 = 3, p_2 = 5, \ldots$, і номером слова $a_{i_0} a_{i_1} \ldots a_{i_k}$ назвемо число $2^{i_0} \cdot 3^{i_1} \ldots p_k^{i_k}$.

Завдяки нумерації текстів поруч із питанням про задання словарної функції алгоритмом із певного класу можна ставити питання про обчислювальність відповідної числової функції. І виявляється, що коли клас алгоритмів є чітко окресленим, то в кожній розумній нумерації слів, на кшталт указаних вище, відповіді на ці питання збігаються: словарну функцію можна задати алгоритмом із даного класу тоді й лише тоді, коли відповідна числова функція є частково рекурсивною. До обґрунтування цього твердження у випадку машин Тюрінга ми зараз і перейдемо.

Композицією машин Тюрінга T_1 і T_2 із внутрішніми алфавітами

$$Q_1 = \{q_1, \ldots, q_m, q_{fin}\} \quad \text{і} \quad Q_2 = \{q'_1, \ldots, q'_r, q'_{fin}\},$$

відповідно, та тим самим зовнішнім алфавітом $A = \{a_0, a_1, \ldots, a_n\}$ називають машину Тюрінга $T = T_1 \circ T_2$ із внутрішнім алфавітом

$$Q = \{q_1, \ldots, q_m, q'_1, \ldots, q'_r, q'_{fin}\},$$

зовнішнім алфавітом $A = \{a_0, a_1, \ldots, a_n\}$ та програмою, яка є об'єднанням програм машин T_1 і T_2 і наступною заміною кожної команди вигляду

$$a_j q_i \to a_k q_{fin} W \quad \text{на} \quad a_j q_i \to a_k q'_1 W.$$

Легко зрозуміти, що коли МТ T_1 і T_2 породжують словарні функції $f_1 : A^* \to A^*$ і $f_2 : A^* \to A^*$, відповідно, то композиція $T_1 \circ T_2$ цих машин породжує суперпозицію $f_1 \circ f_2$ функцій f_1 і f_2.

Теорема 5.9. *Кожна частково рекурсивна функція є обчислювальною за Тюрінгом.*

Доведення містить чотири природні кроки.

1) *Кожна з найпростіших функцій є обчислювальною за Тюрінгом*. Для функцій $S^1(x)$ і O^n це очевидно, а для функцій I_m^n відповідні МТ легко будуються на основі прикладу 3 на с. 116 і вправи 5.2.

2) *Суперпозиція обчислювальних за Тюрінгом функцій є обчислювальною за Тюрінгом*. Нехай МТ $M_1, \ldots, M_m$ і M обчислюють відповідно функції

$f_1(x_1, \ldots, x_n)$, $\ldots$, $f_m(x_1, \ldots, x_n)$ і $g(x_1, \ldots, x_m)$. Тоді МТ для обчислення функції

$$h(x_1, \ldots, x_n) = g\big(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)\big)$$

можна побудувати, використовуючи композицію МТ $M_1, \ldots, M_m$ і M та МТ для дублювання і перестановки аргументів (приклади 2 і 3 на с. 116). Вкажемо лише, що буде записано на стрічці після завершення кожного з етапів роботи такої МТ:

$$01^{x_1+1}01^{x_2+1}0\ldots01^{x_n+1}0,\ \ldots,$$
$$01^{x_1+1}01^{x_2+1}0\ldots01^{x_n+1}01^{x_1+1}01^{x_2+1}0\ldots01^{x_n+1}0,\ \ldots,$$
$$01^{x_1+1}01^{x_2+1}0\ldots01^{x_n+1}01^{f(x_1,x_2,\ldots,x_n)+1}0,\ \ldots,$$
$$01^{f_1(x_1,x_2,\ldots,x_n)+1}01^{x_1+1}01^{x_2+1}0\ldots01^{x_n+1}0,\ \ldots,$$
$$01^{f_1(x_1,x_2,\ldots,x_n)+1}01^{f_2(x_1,x_2,\ldots,x_n)+1}01^{x_1+1}01^{x_2+1}0\ldots01^{x_n+1}0,\ \ldots,$$
$$\ldots\ldots\ldots\ldots$$
$$01^{f_1(x_1,x_2,\ldots,x_n)+1}01^{f_2(x_1,x_2,\ldots,x_n)+1}0\ldots01^{f_m(x_1,x_2,\ldots,x_n)+1}0,\ \ldots,$$
$$01^{g(f_1(x_1,x_2,\ldots,x_n),f_2(x_1,x_2,\ldots,x_n),\ldots,f_m(x_1,x_2,\ldots,x_n))+1}0.$$

3) *Якщо функції g і h є обчислювальними за Тюрінгом, а функція f будується із g та h за допомогою примітивної рекурсії, то f також є обчислювальною за Тюрінгом*. Нехай МТ M_g і M_h обчислюють, відповідно, функції $g(x_1, \ldots, x_n)$ і $h(x_1, \ldots, x_n, y, z)$. Тоді МТ для обчислення функції $f(x_1, \ldots, x_n, y)$, яка будується із g та h за допомогою примітивної рекурсії, можна побудувати за допомогою композицію МТ M_g і M_h та МТ для дублювання і перестановки аргументів (приклади 2 і 3 на с. 116). Основні етапи роботи такої МТ виглядають так:

$$01^{x_1+1}01^{x_2+1}0\ldots01^{x_n+1}01^{y+1}0,\ \ldots,$$
$$01^{x_1+1}0\ldots01^{x_n+1}01^{y+1}01^{x_1+1}0\ldots01^{x_n+1}01^{1}01^{x_1+1}0\ldots01^{x_n+1}0,\ \ldots,$$
$$01^{x_1+1}0\ldots01^{x_n+1}01^{y+1}01^{x_1+1}0\ldots01^{x_n+1}01^{1}01^{g(x_1,\ldots,x_n)+1}0$$
$$(\text{обчислили } f(x_1, \ldots, x_n, 0)),$$
$$\ldots,\ 01^{x_1+1}0\ldots01^{x_n+1}01^{y}01^{1}01^{h(x_1,\ldots,x_n,0,f(x_1,\ldots,x_n,0))}$$
$$(\text{обчислили } f(x_1, \ldots, x_n, 1)),$$
$$\ldots,\ 01^{x_1+1}0\ldots01^{x_n+1}01^{y-1}01^{2}01^{x_1+1}0\ldots01^{x_n+1}01^{2}01^{f(x_1,\ldots,x_n,1)+1}0,$$
$$\ldots,\ 01^{x_1+1}0\ldots01^{x_n+1}01^{y-1}01^{2}01^{h(x_1,\ldots,x_n,1,f(x_1,\ldots,x_n,1))}$$

$$(\text{обчислили } f(x_1,\dots,x_n,2)),$$
$$\dots\dots\dots\dots\dots$$
$$\dots,\ 01^{x_1+1}0\dots01^{x_n+1}01^{1}01^{y}01^{x_1+1}0\dots01^{x_n+1}01^{y}01^{f(x_1,\dots,x_n,y-1)+1}0,$$
$$\dots,\ 01^{x_1+1}0\dots01^{x_n+1}01^{1}01^{y}01^{h(x_1,\dots,x_n,1,f(x_1,\dots,x_n,y-1))+1}0$$
$$(\text{обчислили } f(x_1,\dots,x_n,y)),$$
$$\dots,\ 01^{f(x_1,\dots,x_n,y)+1}0.$$

Випадок $n = 0$ пропонуємо читачеві розглянути самостійно.

4) *Якщо функція f є обчислювальною за Тюрінгом, а функція g будується з f за допомогою оператора мінімізації, то g також є обчислювальною за Тюрінгом.* Нехай $f(x_1,\dots,x_{n+1})$ — (часткова) обчислювальна функція арності $n+1$ і M_f — МТ, яка її обчислює. МТ M_g для обчислення функції $g(x_1,\dots,x_n) = \mu_y\big(f(x_1,\dots,x_n,y) = 0\big)$ будується за допомогою композиції МТ M_f і МТ для дублювання і перестановки аргументів (приклади 2 і 3 на с. 116) та МТ для порівняння двох чисел (вправа 5.3). Основні етапи роботи M_g виглядають так:

$$01^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}0,\dots,$$
$$01^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}01^{1}01^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}01^{1}0,\dots,$$
$$01^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}01^{1}01^{f(x_1,\dots,x_n,0)+1}0.$$

Якщо $f(x_1,\dots,x_n,0) = 0$, то M_g лишає на стрічці лише $01^{1}0$ і завершує роботу. У протилежному разі M_g продовжує працювати:

$$01^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}01^{1}0,\dots$$
$$01^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}01^{2}01^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}01^{2}0,\dots$$
$$01^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}01^{2}01^{f(x_1,\dots,x_n,1)+1}0.$$

Якщо $f(x_1,\dots,x_n,1) = 0$, то M_g лишає на стрічці лише $01^{2}0$ і завершує роботу. У протилежному разі M_g продовжує працювати:

$$01^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}01^{2}0,\dots,$$
$$01^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}01^{3}01^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}01^{3}0,\dots,$$
$$01^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}01^{3}01^{f(x_1,\dots,x_n,2)+1}0$$

і т. д. □

Теорема 5.10. *При геделівській нумерації кожній словарній функції, обчислювальній за Тюрінгом, відповідає частково рекурсивна функція.*

Доведення. Нехай дано МТ T із зовнішнім алфавітом $A = \{a_0, a_1, \ldots, a_k\}$ та внутрішнім алфавітом $Q = \{q_1, \ldots, q_m, q_{fin}\}$. Об'єднаємо їх у єдиний алфавіт $\{c_0, c_1, \ldots, c_{k+m+1}\}$, де

$$c_i = \begin{cases} a_i, & \text{якщо } 0 \le i \le k; \\ q_j, & \text{якщо } i = k + j,\ 0 < j \le m; \\ q_{fin}, & \text{якщо } i = k + m + 1. \end{cases}$$

Далі доведення містить чотири етапи:

1) *Перехід від вхідного слова* $a_{i_1} \ldots a_{i_n}$ *до початкової конфігурації* $q_1 a_{i_1} \ldots a_{i_n}$.

Слово $a_{i_1} \ldots a_{i_n}$ має номер $2^{i_1} 3^{i_2} \cdots p_{n-1}^{i_n}$, а слово $q_1 a_{i_1} \ldots a_{i_n}$ — номер $2^{k+1} 3^{i_1} \cdots p_n^{i_n}$. Функція $g(x)$, яка за номером x першого слова обчислює номер другого, має вигляд

$$g(x) = 2^{k+1} \prod_{i=0}^{x} p_{i+1}^{f(i,x)},$$

де $f(i, x)$ — показник, з яким i-те просте число p_i входить у канонічний розклад числа x. Оскільки за теоремою 5.8(5) функція $f(i, x)$ є примітивно рекурсивною, то $g(x)$ також примітивно рекурсивна.

2) *Моделювання одного такту роботи МТ.* Через $j(x)$ позначимо функцію, яка обчислює номер місця, на якому в конфігурації з номером x стоїть символ внутрішнього алфавіту:

$$j(x) = \mu_i\big(f(i, x) > k\big) + 1.$$

Нехай тепер у результаті застосування до конфігурації

$$c_{i_1} \ldots c_{i_{j(x)-1}} c_{i_{j(x)}} c_{i_{j(x)+1}} \ldots c_r,\ \ i_{j(x)} = k + j,\ 0 < j \le m,$$

з номером x команди $a_{i_{j(x)}+1} q_j \to a_t q_l W$ отримують конфігурацію

$$c_{i_1} \ldots c_{i_{j(x)-2}} b_1 b_2 b_3 c_{i_{j(x)+2}} \ldots c_r.$$

Функцію $h(x)$, яка за номером x першої конфігурації дає номер другої конфігурації, задають частинами (див. теорему 5.6) залежно від значення пари

чисел $f(j(x),x)$ і $f(j(x)+1,x)$, які визначають команду $a_{i_{j(x)}+1}q_j \to a_t q_l W$. Указана функція має вигляд

$$h(x) = \left[\frac{x}{p_{i_{s-1}-1}^{i_{s-1}} \cdot p_{i_s-1}^{i_s} \cdot p_{i_{s+1}-1}^{i_{s+1}}}\right] \cdot p_{i_{s-1}-1}^{t_1} p_{i_s-1}^{t_2} p_{i_{s+1}-1}^{t_3},$$

де

$$(t_1,t_2,t_3) = \begin{cases} (k+l, i_{s-1}, t), & \text{якщо } W = L; \\ (i_{s-1}, t, k+l), & \text{якщо } W = R; \\ (i_{s-1}, k+l, t), & \text{якщо } W = S. \end{cases}$$

3) *Моделювання роботи МТ в цілому.* Позначимо через $F(x,i)$ номер конфігурації, яка виникла, коли МТ почала роботу на слові з номером x і пропрацювала i тактів. Легко бачити, що

$$F(x,0) = g(x),$$
$$F(x,i+1) = h(F(x,i)).$$

Зауваження. *Усі функції, які з'являлися до цього часу, були примітивно рекурсивними.*

4) *Виділення результату.* Спочатку виділяємо момент зупинки МТ:

$$q_0(x) = \mu_y\big(f(j(F(x,y))-1, F(x,y)) = k+m+1\big).$$

Щоб знайти номер $\mathrm{Fin}(x)$ слова, в яке МТ переробила вхідне слово x, лишилося з кінцевої конфігурації вилучити символ q_{fin}:

$$\mathrm{Fin}(x) = \prod_{i=0}^{j(F(x,q_0(x)))-1} p_i^{f(i,F(x,q_0(x)))} \cdot \prod_{i=j(F(x,q_0(x)))}^{x} p_i^{f(i+1,F(x,q_0(x)))}.$$

□

5.5.2. Теза Черча

Той факт, що при ґеделівській нумерації клас словарних функцій, обчислювальних за Тюрінгом, збігається з класом частково рекурсивних функцій, ліг в основу знаменитої *тези Черча* (у формі Кліні):

Теза Черча: *клас усіх алгоритмічно* $(=$ *ефективно*$)$ *обчислювальних числових функцій збігається з класом частково рекурсивних функцій.*

Починаючи з 1933 р. запропоновано багато дуже різних формальних уточнень поняття алгоритму. Однак відносно всіх уточнень:

— частково рекурсивні функції (Ербран, Ґедель, Кліні, 1934–36),

— μ-рекурсивні функції (Ґедель, Кліні, 1936),

— λ-визначальні функції (Черч, Кліні, 1933–36),

— машини Тюрінга та машини Поста (Тюрінг, Пост, 1936),

— нормальні алгорифми (Марков, 1950),

— графічні схеми (Петер, 1958)

та кількох десятків інших доведено їхню еквівалентність у тому ж сенсі, в якому машини Тюрінга еквівалентні частково рекурсивним функціям.

Ця обставина та те, що

a) усі алгоритми в кожному із цих *точних* значень є алгоритмами в *інтуїтивному* сенсі;

b) усі відомі алгоритми в інтуїтивному сенсі вдається змоделювати алгоритмами в точному значенні,

привели до переконання, що *кожне з цих уточнень адекватно відображає інтуїтивне поняття алгоритму*. Це складає зміст *тези Черча* у широкому її розумінні.

Теза Черча є природничо-науковим постулатом, аналогічним законам Ньютона або началам термодинаміки, і є спробою замінити інтуїтивне поняття алгоритму формальним. Тому математично довести її (у звичайному математичному розумінні поняття доведення) не можна. Але жодного контрприкладу до неї невідомо, а аргументи на її користь наведені вище.

Прийнявши тезу Черча, можна надати необхідну точність формулюванням алгоритмічних проблем і в деяких випадках зробити можливим доведення їхньої нерозв'язності. Справді, завдяки тезі Черча проблему існування алгоритму для розв'язування цієї масової задачі можна звести, наприклад, до питання про рекурсивність певної функції. А далі, використовуючи звичайну математичну техніку, спробувати довести, що ця функція не є рекурсивною.

Саме у такий спосіб Черчу 1936 р. вдалося довести нерозв'язність основної алгоритмічної проблеми логіки предикатів — проблеми з'ясування загальнозначущості формул логіки першого порядку.

З огляду на тезу Черча — що клас обчислювальних функцій збігається з класом частково рекурсивних функцій, то звідси одразу випливає *існування необчислювальних функцій* $f : \mathbb{N} \to \mathbb{N}$. Це очевидно з міркувань потужності: усього функцій $f : \mathbb{N} \to \mathbb{N}$ є континуум багато, а обчислювальних — лише зліченна кількість.

Теорема 5.11. *Існують загальнорекурсивні функції, які не є примітивно рекурсивними.*

Доведення. Кожну примітивно рекурсивну функцію одержують із найпростіших за допомогою певної послідовності операцій суперпозиції та примітивної рекурсії. Цю послідовність можна описати скінченним текстом: потрібно вказати послідовність примітивно рекурсивних функцій, які використовують у побудові даної функції, і для кожної вказати, з яких попередньо побудованих функцій і за допомогою якої операції вона отримується. Із таких текстів можна виокремити ті, які відповідають примітивно рекурсивним функціям однієї змінної, і послідовно їх занумерувати f_0, f_1, f_2, ... Розглянемо тепер функцію $U(n, m)$, яка дорівнює значенню примітивно рекурсивної функції з номером n у точці m. Тоді функція $F(n) = U(n, n)+1$ буде скрізь визначеною ефективно обчислювальною функцією, отже, за тезою Черча, загальнорекурсивною. Але якби $F(n)$ збігалася з якоюсь функцією f_k, то в точці k ми мали б

$$f_k(k) = F(k) = f_k(k) + 1.$$

Отримана суперечність доводить, що загальнорекурсивна функція $F(n)$ не є примітивно рекурсивною. □

Машина Тюрінга задається своїми внутрішнім і зовнішнім алфавітами та програмою. Використовуючи індекси, ми можемо задавати МТ скінченними текстами у спільному для всіх МТ скінченному алфавіті $\{a, q, 0, 1, \ldots, 9, \to, S, L, R, *\}$ (останній символ введено для зручності, щоб можна було, напр., розділяти команди або відмічати початок і кінець слова). Тому є лише зліченна кількість МТ і їх можна ефективно перерахувати:

$$M_1, M_2, \ldots, M_n, \ldots .$$

Очевидно, що за номером n можна ефективно відновити машину M_n (точніше, її алфавіти і програму), а потім змоделювати роботу M_n на будь-якому вхідному слові v. Згідно з тезою Черча це може робити деяка МТ. Приходимо до поняття *універсальної машини Тюрінга* M_U:

$$M_U(n, v) = M_n(v)\,. \tag{5.8}$$

Інколи універсальну машину Тюрінга визначають трохи інакше: на вхід машини M_U подають не пару (n, v), а програму машини M_n і слово v. З одного боку таке означення універсальної машини рівносильне попередньому, з другого — воно майже збігається з розумінням сучасної ЕОМ.

5.6. Канторівська нумерація кортежів

Множина $\mathbb{N}^2$ пар натуральних чисел зліченна, тому її елементи можна занумерувати (тобто вишикувати пари у лінійному порядку). Це можна зробити багатьма способами. Але з погляду теорії алгоритмів природно вимагати, щоб нумерація пар задовольняла такі дві умови:

— для кожної пари (m, n) натуральних чисел можна ефективно обчислити її номер;

— за номером пари (m, n) можна ефективно обчислити її компоненти m і n.

Серед нумерацій, що задовольняють ці умови, однією з найчастіше використовуваних є так звана *канторівська* (її ще називають *нумерацією Пеано*):

$$(0,0), (0,1), (1,0), (0,2), (1,1), (2,0), (0,3), \ldots$$

(пара з меншою сумою компонентів іде раніше, а з двох пар з однаковою сумою компонентів першою іде пара з меншим першим компонентом). Номер $\boldsymbol{n}(x, y)$ пари (x, y) у цій послідовності називають її *канторівським номером*. Отже, $\boldsymbol{n}(0, 0) = 0$, $\boldsymbol{n}(0, 1) = 1$, $\boldsymbol{n}(1, 0) = 2$, ... Через $\boldsymbol{l}(n)$ і $\boldsymbol{r}(n)$ позначимо відповідно лівий і правий компоненти пари з номером n.

Твердження 5.1. a) $\boldsymbol{n}(x, y) = \dfrac{(x+y)(x+y+1)}{2} + x$;

b) $\boldsymbol{l}(n) = n \ominus \dfrac{1}{2}\left[\dfrac{[\sqrt{8n+1}]+1}{2}\right] \cdot \left[\dfrac{[\sqrt{8n+1}] \ominus 1}{2}\right]$;

c) $\boldsymbol{r}(n) = \dfrac{1}{2}\left[\dfrac{[\sqrt{8n+1}]+5}{2}\right] \cdot \left[\dfrac{[\sqrt{8n+1}] \ominus 1}{2}\right] \ominus n$.

Зауваження. *У рівностях* b) *і* c) *використано знак* $\ominus$*, щоб підкреслити, що дія скрізь визначена. Насправді, як легко пересвідчитись, дія* $\ominus$ *у цих рівностях збігається із звичайним відніманням.*

Доведення. a) Кількість пар (x, y), для яких $x + y = k$, дорівнює $k + 1$. Тому номер пари (x, y), враховуючи, що нумерація пар починається з 0, дорівнюватиме

$$\boldsymbol{n}(x, y) = 1 + 2 + 3 + \cdots + (x + y) + x = \frac{(x+y)(x+y+1)}{2} + x.$$

b) З a) випливає, що

$$8n = 4(x + y)^2 + 12x + 4y = (2x + 2y + 1)^2 + 8x - 1.$$

Тому $8n+1 \geq (2x+2y+1)^2$. З іншого боку,

$$8n = (2x+2y+3)^2 - 8x - 9.$$

Тоді $8n+1 < (2x+2y+3)^2$. Отримаємо

$$2x+2y+1 \leq [\sqrt{8n+1}] < 2x+2y+3$$

і

$$x+y+1 \leq \frac{[\sqrt{8n+1}]+1}{2} < x+y+2.$$

Отже,

$$x+y+1 = \left[\frac{[\sqrt{8n+1}]+1}{2}\right] \quad \text{і} \quad x+y = \left[\frac{[\sqrt{8n+1}]-1}{2}\right]. \tag{5.9}$$

З а) тепер одержуємо, що

$$\boldsymbol{l}(n) = x = n - \frac{(x+y)(x+y+1)}{2} = n - \frac{1}{2}\left[\frac{[\sqrt{8n+1}]+1}{2}\right] \cdot \left[\frac{[\sqrt{8n+1}]-1}{2}\right].$$

с) Враховуючи отриману рівність для $\boldsymbol{l}(n)$ і другу з рівностей (5.9), маємо

$$\boldsymbol{r}(n) = y = x+y-\boldsymbol{l}(n) = \left[\frac{[\sqrt{8n+1}]-1}{2}\right] - \boldsymbol{l}(n) =$$
$$= \frac{1}{2}\left[\frac{[\sqrt{8n+1}]+5}{2}\right] \cdot \left[\frac{[\sqrt{8n+1}]-1}{2}\right] - n.$$

□

Зазначимо, що функції $\boldsymbol{l}(n)$, $\boldsymbol{r}(n)$ і $\boldsymbol{n}(x,y)$ із твердження 5.1 є загальнорекурсивними.

Канторівську нумерацію легко узагальнити на набори більшої арності (індекс в $\boldsymbol{n}^k$ вказує на арність відповідної функції):

$\boldsymbol{n}^3(x_1,x_2,x_3) := \boldsymbol{n}(\boldsymbol{n}(x_1,x_2),x_3)$,
$\boldsymbol{n}^4(x_1,x_2,x_3,x_4) := \boldsymbol{n}^3(\boldsymbol{n}(x_1,x_2),x_3,x_4)$

і т. д.

Використовуючи твердження 5.1, за номером $n = \boldsymbol{n}^k(x_1,x_2,\ldots,x_k)$ набору $(x_1,x_2,\ldots,x_k)$ легко відновити його компоненти:

$$\begin{array}{rcl}
x_k & = & \boldsymbol{r}(n), \\
x_{k-1} & = & \boldsymbol{r}(\boldsymbol{l}(n)), \\
x_{k-2} & = & \boldsymbol{r}(\boldsymbol{l}(\boldsymbol{l}(n))), \\
\ldots & \ldots & \ldots \\
x_2 & = & \boldsymbol{r}(\boldsymbol{l}(\ldots\boldsymbol{l}(n)\ldots)), \\
x_1 & = & \boldsymbol{l}(\boldsymbol{l}(\ldots\boldsymbol{l}(n)\ldots)).
\end{array}$$

Канторівська нумерація пар дозволяє переходити від функцій двох аргументів до функцій одного аргументу і навпаки:

$$f(x,y) \to g(\boldsymbol{n}(x,y)), \quad g(n) \to f(\boldsymbol{l}(n), \boldsymbol{r}(n)).$$

Зрозуміло, що такі переходи можна робити і для функцій від більшої кількості аргументів. Тому канторівська нумерація кортежів дозволяє в багатьох питаннях обмежитися функціями лише одного аргументу.

5.7. Ефективне задання множин (розв'язність і перераховність)

Множини можна задавати різними способами:
a) *переліком* (списком своїх елементів);
b) *породжувальною* процедурою або
c) *описом характеристичних властивостей* своїх елементів.

Щодо цих способів можна зауважити таке:

a) цей спосіб придатний лише для скінченних і відносно невеликих множин;

b) породжувальна процедура вказує спосіб побудови чергових елементів множини з уже побудованих елементів або з інших об'єктів. Елементами множини вважають усі ті й лише ті об'єкти, які можна побудувати за допомогою такої процедури;

c) цей спосіб найзвичніший. Множину при цьому виділяють як підмножину тих елементів деякої універсальної множини, що мають задані властивості. Але опис характеристичних властивостей має бути точним і недвозначним. Наприклад, не можна вважати строго заданими множину всіх хороших книг або множину розумних студентів. Другим важливим моментом такого задання множин є вимога можливості перевіряти для елементів універсальної множини наявність у них цих властивостей, тобто задання певної процедури (яку називають *розв'язувальною процедурою*), яка для будь-якого елемента універсальної множини встановлює (*розпізнає*), чи має він задані характеристичні властивості.

Отже, зазвичай маємо справу лише з *розв'язними* (спосіб c)) і перераховними (спосіб b)) множинами (точні означення цих понять дамо трохи нижче). Тому набір (фактичний) множин, з якими мають справу математики, насправді дуже малий.

Зауважимо, що розв'язувальна процедура часто не є породжувальною (напр., коли розглядаємо множину коренів даного рівняння). З іншого боку, породжувальна процедура може не бути розв'язувальною. Навіть більше, для

перераховних множин розв'язувальної процедури може взагалі не існувати. Відповідні приклади наведемо далі.

Приклад. *Нехай A — множина всіх наборів із k цифр, які зустрічаються в десятковому розкладі числа π (підряд у даному порядку). Оскільки є алгоритми, які дозволяють обчислити як завгодно велику кількість знаків числа π (перший алгоритм для обчислення π з якою завгодно точністю запропонував ще Архімед), то для A існує очевидна породжувальна процедура. Для A існує також і розв'язувальна процедура. Справді, із скінченності множини A випливає, що існує список усіх її елементів. Тому можна запропонувати таку розв'язувальна процедуру: беремо даний набір і дивимось, чи зустрічається він у цьому списку. Але, хоча розв'язувальна процедура існує, вказати її для достатньо великих k на нинішньому етапі розвитку математики ми не можемо.*

Далі обмежимося лише підмножинами з $\mathbb{N}^k$ (враховуючи канторівську нумерацію кортежів, можна навіть обмежитися підмножинами з $\mathbb{N}$), хоча більшість понять і тверджень легко переносять на множини загальнішої природи.

Характеристичною функцією підмножини $A \subseteq \mathbb{N}^k$ називають функцію χ_A, яка дорівнює 1 в усіх точках $x \in A$, і дорівнює 0 в усіх точках $x \notin A$.

Частковою характеристичною функцією підмножини $A \subseteq \mathbb{N}^k$ називають функцію $\widehat{\chi}_A$, яка дорівнює 1 в усіх точках $x \in A$, і невизначена в усіх точках $x \notin A$.

Множину $A \subseteq \mathbb{N}^k$ називають *розв'язною* (або *рекурсивною*), якщо її характеристична функція χ_A є загальнорекурсивною, тобто, якщо існує ефективна процедура, яка для кожного елемента $x \in \mathbb{N}^k$ дозволяє з'ясувати, належить він чи ні множині A.

Твердження 5.2 (властивості розв'язних множин)**.** a) *Якщо одна з множин A або $\mathbb{N}^k \setminus A$ розв'язна, то друга також розв'язна.*

b) *Якщо множини A і B — розв'язні, то множини $A \cap B$, $A \cup B$ і $A \setminus B$ також розв'язні.*

c) *Кожна скінченна множина є розв'язною.*

d) *Якщо множини $A \subseteq \mathbb{N}^k$ і $B \subseteq \mathbb{N}^m$ — розв'язні, то множина $A \times B \subseteq \mathbb{N}^{k+m}$ також розв'язна.*

Доведення. a) Це випливає з рівності $\chi_A(x) + \chi_{\mathbb{N} \setminus A}(x) = 1$.

b) Маємо це з рівностей

$$\chi_{A \cap B} = \chi_A \cdot \chi_B, \;\; \chi_{A \cup B} = \chi_A + \chi_B - \chi_A \cdot \chi_B, \;\; \chi_{A \setminus B} = \chi_A - \chi_A \cdot \chi_B.$$

c) За теоремою 5.6 про функцію, задану частинами, функція

$$\chi_{\{a_1,\ldots,a_k\}}(x) = \begin{cases} 1, & \text{якщо } x = a_1; \\ \cdots\cdots\cdots\cdots\cdots\cdots & \\ 1, & \text{якщо } x = a_k; \\ 0 & \text{в інших випадках} \end{cases} \tag{5.10}$$

є примітивно рекурсивною.

d) Це випливає з рівності

$$\chi_{A\times B}(x_1,\ldots,x_k,x_{k+1},\ldots,x_{k+m}) = \chi_A(x_1,\ldots,x_k)\chi_B(x_{k+1},\ldots,x_{k+m}).\ \square$$

Множину $A \subseteq \mathbb{N}^k$ називають *перераховною* (або *рекурсивно перераховною*), якщо її часткова характеристична функція $\widehat{\chi}_A$ є частково рекурсивною.

Твердження 5.3 (властивості перераховних множин)**.** a) *Кожна розв'язна множина є перераховною.*

b) *Якщо множини A і B — перераховні, то множина $A \cap B$ також перераховна.*

c) *Якщо множини A і B — перераховні, то множина $A \cup B$ також перераховна.*

Доведення. a) $\widehat{\chi}_A = (\chi_A - 1) + 1$.

b) $\widehat{\chi}_{A\cap B} = \widehat{\chi}_A \cdot \widehat{\chi}_B$.

c) Нехай T_A і T_B — машини Тюрінга, які обчислюють функції $\widehat{\chi}_A$ і $\widehat{\chi}_B$ відповідно. Можна запропонувати такий алгоритм обчислення $\widehat{\chi}_{A\cup B}(x)$: на першому кроці кожна з машин T_A і T_B працює на вході x один такт, на другому кроці кожна з машин працює ще один такт, на третьому — ще один такт і т. д. Якщо $x \in A \cup B$, то на якомусь кроці матимемо результативну зупинку хоча б однієї з МТ T_A і T_B і $\widehat{\chi}_{A\cup B}(x) = 1$. Якщо ж $x \notin A \cup B$, то на жодному кроці жодна з МТ T_A і T_B не зупиниться і значення $\widehat{\chi}_{A\cup B}(x)$ буде невизначеним. Отже, функція $\widehat{\chi}_{A\cup B}(x)$ є ефективно обчислювальною, а тому, за тезою Черча, вона є частково рекурсивною. $\square$

Теорема 5.12. *Наведені умови рівносильні:*

a) *множина $A \subseteq \mathbb{N}^k$ є перераховною*;

b) *множина $A \subseteq \mathbb{N}^k$ є областю визначення деякої частково рекурсивної функції*;

c) *множина $A \subseteq \mathbb{N}^k$ є множиною значень деякої частково рекурсивної функції.*

Доведення. Враховуючи канторівську нумерацію кортежів, теорему досить довести лише для підмножин із $\mathbb{N}$.

a) $\Rightarrow$ b). Перераховна множина A є областю визначення функції $\widehat{\chi}_A$.

b) $\Rightarrow$ a). Якщо A — область визначення частково рекурсивної функції f, то $\widehat{\chi}_A(x) = \operatorname{sg}(f(x)+1)$. Тому множина A — перераховна.

a) $\Rightarrow$ c). Нехай множина $A \subseteq \mathbb{N}$ є перераховною і T — машина Тюрінга, яка обчислює функцію $\widehat{\chi}_A$. Далі i-й такт роботи машини T на вході k позначимо через $T_i(k)$. Зручно вважати, що коли на вході k машина T зупинилася на j-му такті роботи, то вона перебуває у стані результативної зупинки і на наступних тактах роботи на цьому вході. Розглянемо тепер послідовність

$$T_1(0), T_2(0), T_1(1),\ T_3(0),\ T_2(1),\ T_1(2),\ T_4(0),\ T_3(1),\ T_2(2),\ T_1(3),\ T_5(0), \ldots$$

Очевидно, що ця послідовність є ефективно обчислювальною. Нехай n-й член цієї послідовності має вигляд $T_i(k)$. Розглянемо тепер функцію f, визначену таким чином: $f(n) = k$, якщо $T_i(k)$ — це стан результативної зупинки, і $f(n)$ не визначена у протилежному випадку.

Функція f є ефективно обчислювальною, а тому, за тезою Черча, вона частково рекурсивна. Множина її значень є множиною тих n, на яких машина T зупиняється, тобто збігається з множиною A.

c) $\Rightarrow$ a). Доводять аналогічно попередньому тим самим діагональним методом. Нехай A — множина значень деякої частково рекурсивної функції f. Позначимо через $F(n,k)$ результат k кроків роботи алгоритму для функції f при обчисленні $f(n)$. Перебираючи послідовно пари (n,k) натуральних чисел, отримуємо послідовність цих результатів. Розглянемо функцію $\widehat{\chi}$, яка визначена в точці a (і набуває значення 1) тоді й лише тоді, коли a зустрічається в цій послідовності. Тоді частково рекурсивна функція $\widehat{\chi}$ є частковою характеристичною функцією множини A. $\square$

Теорема 5.12 і теза Черча дозволяють переформулювати означення перераховної множини у такій загальній формі:

множину A називають ***перераховною*** *(або* ***рекурсивно перераховною****), якщо існує алгоритм, який рано чи пізно породжує довільний її елемент.*

Кажуть також, що цей алгоритм *перераховує* множину A. Зауважимо, що в переліку $a_1, a_2, a_3, \ldots$ елементів множини A можуть бути повтори.

Твердження 5.4. a) *Якщо множини A і B — перераховні, то множина $A \times B$ також перераховна.*
b) *Якщо множина $A \subseteq \mathbb{N}^k \times \mathbb{N}^m$ — перераховна, то проєкції A на кожен із множників також перераховні.*

Доведення. а) Якщо є алгоритми, які послідовно породжують елементи a_1, a_2, a_3, ... множини A і елементи $b_1, b_2, b_3, \ldots$ множини B, то за допомогою діагонального методу легко будувати алгоритм, який послідовно породжує елементи

$$(a_1, b_1), (a_2, b_1), (a_1, b_2), (a_3, b_1), (a_2, b_2), (a_1, b_3), (a_4, b_1), (a_3, b_2), (a_2, b_3), \ldots$$

множини $A \times B$.

b) Очевидно. □

У термінах алгоритмів можна переформулювати й означення розв'язної множини:

*Підмножину $A \subseteq \mathbb{N}^k$ називають **розв'язною**, якщо існує алгоритм, який розпізнає елементи множини A, тобто переробляє всі елементи з A на те саме слово v (напр., на "так"), а всі елементи з $\mathbb{N}^k \setminus A$ — на те саме, відмінне від v слово w (напр., на "ні").*

Кажуть також, що цей алгоритм *розпізнає* множину A.

Теорема 5.13. *Функція $f : \mathbb{N} \to \mathbb{N}$ обчислювальна тоді й лише тоді, коли її графік $\Gamma_f = \{(x, y) \mid y = f(x)\}$ є перераховною множиною.*

Доведення. Необхідність. Нехай — f обчислювальна. За теоремою 5.12 існує алгоритм $\mathfrak{A}$, який перераховує її область визначення. Використовуючи композицію цього алгоритму з алгоритмом, який обчислює функцію f, очевидно будується алгоритм, який перераховує графік Γ_f.

Достатність. Нехай алгоритм $\mathfrak{B}$ перераховує графік Γ_f. Тоді для кожного n значення $f(n)$ можна обчислювати так: запускаємо алгоритм $\mathfrak{B}$, і як тільки з'являється пара із Γ_f із першим компонентом n, її другий компонент дає нам $f(n)$. □

Теорема 5.14. *Якщо множина $A \subseteq \mathbb{N}$ перераховна, а функція $f : \mathbb{N} \to \mathbb{N}$ — обчислювальна, то кожна з множин $f(A)$ і $f^{-1}(A)$ також буде перераховною.*

Доведення. Образ $f(A)$ можна отримати так: будуємо перетин графіка $\Gamma_f = \{(x, y) \mid y = f(x)\}$ функції f (він перераховний за теоремою 5.13) з перераховною множиною $A \times \mathbb{N}$ і беремо проєкцію перетину на другу координату. Перераховність цього перетину випливає з твердження 5.4.

Перераховність прообразу $f^{-1}(A)$ доводимо аналогічно. □

Теорема 5.15 (Поста)**.** *Множина S буде розв'язною тоді й лише тоді, коли вона сама і її доповнення $\mathbb{N} \setminus S$ є перераховними.*

Доведення. Нехай множина S є розв'язною. Тоді існує алгоритм $\mathfrak{A}$ для розпізнавання її елементів. Застосовуючи $\mathfrak{A}$ послідовно до чисел 0, 1, 2, ..., одержуємо переліки кожної з множин S і $\mathbb{N} \setminus S$.

Навпаки, нехай кожна з множин S і $\mathbb{N} \setminus S$ є перераховною, а $\mathfrak{A}_1$ і $\mathfrak{A}_2$ — алгоритми, які породжують переліки цих множин. Тоді кожне натуральне число n рано чи пізно зустрінеться в послідовності

$$\mathfrak{A}_1(0),\ \mathfrak{A}_2(0),\ \mathfrak{A}_1(1),\ \mathfrak{A}_2(1),\ \mathfrak{A}_1(2),\ \mathfrak{A}_2(2),\ \mathfrak{A}_1(3),\ \mathfrak{A}_2(3),\ \mathfrak{A}_1(4),\ \ldots,$$

що зразу дає відповідь на питання, якій саме із множин S чи $\mathbb{N} \setminus S$ воно належить[44]. □

Теорема 5.16. *Існує перераховна, але нерозв'язна множина натуральних чисел*[45].

Доведення. За твердженням 5.12 перераховна множина — це область значень обчислювальної функції.

Алгоритми обчислення обчислювальних функцій можна ефективно занумерувати: $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, ... Розглядаючи для кожного алгоритму множину значень відповідної функції, одержуємо перелік перераховних множин: A_1, A_2, A_3, ... Припустимо, що кожна перераховна множина є розв'язною. Тоді з теореми Поста випливає, що разом із кожною множиною у списку A_1, A_2, A_3, ... зустрічається і її доповнення.

Розглянемо множину $A = \{n \mid n \in A_n\}$. З означення A випливає, що для кожного n $\ A \cap A_n = A \cup A_n$. За припущенням кожна з множин A_n є розв'язною, тому множина A також буде A розв'язною. Але тоді множина $\overline{A}$ є перераховною і зустрічається в $A_1, A_2, A_3, \ldots$. Нехай $\overline{A} = A_k$. Тоді $A \cap A_k = \varnothing$, звідки $k \notin A \cap A_k$. Але оскільки $A \cap A_k = A \cup A_k$ то $k \notin A \cup A_k$, що суперечить рівності $A \cup A_k = \mathbb{N}$. Отримана суперечність доводить, що не всі перераховні множини є розв'язними. □

Вправа 5.4. *Доведіть, що область визначення функції $g(n)$ із доведення теореми* 5.1 *буде перераховною нерозв'язною множиною.*

Теорема 5.17. *Множина номерів загальнорекурсивних функцій є неперераховною.*

[44] Першим цей надзвичайно примітивний перебірковий алгоритм запропонував, мабуть, іспанський схоласт Луллій, якого пізніше жорстоко висміяв Свіфт у своєму описі Лапутянської академії в "Мандрах Гуллівера". А нині цей алгоритм має дуже солідну назву "алгоритму британського музею".

[45] Теорема 5.16 є одним із найважливіших фактів теорії алгоритмів.

Доведення. Нехай f_0, f_1, f_2, ... — перелік загальнорекурсивних функцій. Тоді функція $g(n) = f_n(n) + 1$ є загальнорекурсивною, але в цьому переліку не зустрічається. □

Теорема 5.18. *Якщо алфавіт числення предикатів містить зліченну множину символів змінних і не більше ніж зліченні множини індивідних констант, предикатних символів і функціональних символів, то множина загальнозначущих формул буде перераховною.*

Доведення. Елементи кожної з перерахованих в умові теорем множин можна занумерувати натуральними числами, а потім ефективно вказати єдину нумерацію для всіх символів алфавіту. Далі за допомогою діагонального методу можна ефективно занумерувати всі правильно побудовані формули числення предикатів, а потім і всі виводи. Отримуємо ефективну процедуру (так званий *алгоритм Гільберта*), яка дозволяє послідовно виписувати всі теореми даного числення предикатів. Отже, множина всіх теорем виявляється перераховною. За теоремою Ґеделя про повноту (теорема 4.16) множина теорем збігається з множиною загальнозначущих формул. Тому остання також буде перераховною. □

Зауваження. *На жаль, алгоритм Гільберта не дозволяє, взагалі кажучи, з'ясувати, чи буде загальнозначущою конкретна формула F. Якщо в послідовності формул, яку генерує алгоритм, формула F з'явилась, то вона, звісно, є загальнозначущою. Однак, якщо алгоритм працює, а F так і не з'являється, то нічого певного сказати не можна: можливо, вона не є загальнозначущою, а можливо, що до неї ще просто не дійшла черга. Але насправді проблема не в тому, що алгоритм Гільберта недосконалий. Черч* 1936 *р. довів, що коли алфавіт числення предикатів містить хоча б один функціональний символ арності ≥ 1 або ж предикатний символ арності ≥ 2, то множина загальнозначущих формул не є розв'язною.*

Вправа 5.5. *Доведіть, що коли алфавіт числення предикатів містить лише предикати арності* 1*, то множина загальнозначущих формул є розв'язною.*

5.8. Алгоритмічна нерозв'язність

Масову задачу називають *алгоритмічно нерозв'язною*, якщо не існує алгоритму, який би її розв'язував.

Приклад. *З теореми* 5.17 *випливає, що задача розпізнавання за номером частково рекурсивної функції, чи є вона скрізь визначеною, є алгоритмічно нерозв'язною.*

Наступний приклад алгоритмічно нерозв'язної задачі, який ми розглянемо, пов'язаний із проблемою *самозастосовності* алгоритмів. Так називають властивість алгоритму успішно завершувати свою роботу на вхідних даних, що є формальним записом цього алгоритму. *Задача розпізнавання самозастосовності* полягає у знаходженні такого алгоритму, який може за формальним записом довільного алгоритму за скінченну кількість кроків з'ясувати, чи є цей алгоритм самозастосовним.

Теорема 5.19. *Задача розпізнавання самозастосовності є алгоритмічно нерозв'язною.*

Доведення. Припустимо, алгоритм, який розпізнає самозастосовність, існує. На підставі тези Черча можна стверджувати, що існує машина Тюрінга $\mathcal{M}$, яка реалізує цей алгоритм. Уважатимемо, що $\mathcal{M}$ переробляє формальний запис алгоритму $\mathcal{A}$ на 1, якщо $\mathcal{A}$ самозастосовний, і на 0 у протилежному разі.

Розглянемо тепер машину Тюрінга $\mathcal{M}'$, яка переробляє формальні записи несамозастосовних алгоритмів на 0, а у випадку самозастосовних алгоритмів ніколи не зупиняється. Така машина легко будується з $\mathcal{M}$: після того, як $\mathcal{M}$ записала на стрічці символ 1 і мала зупинитися, нова машина не зупиняється, а починає, наприклад, нескінченно рухатися праворуч. Нова машина $\mathcal{M}'$ не є ні самозастосовною, ні несамозастосовною. Справді, якщо $\mathcal{M}'$ самозастосовна, то її можна застосувати до свого формального запису. Але з побудови $\mathcal{M}'$ випливає, що в цьому випадку вона ніколи не зупиниться, тобто є несамозастосовною. Якщо ж $\mathcal{M}'$ несамозастосовна, то вона на своєму формальному записові повинна зупинитися, тобто є самозастосовною. Отримана суперечність доводить, що машини $\mathcal{M}$ не існує. □

Задача розпізнавання самозастосовності є досить штучною. Але її часто використовують для доведення алгоритмічної нерозв'язності більш природних і складніших задач: із припущення про існування алгоритму для розв'язання такої задачі виводиться існування алгоритму для розв'язання задачі розпізнавання самозастосовності. Зокрема і у дослідженні машин Тюрінга виникає природне питання: чи можна за програмою $\Pi(T)$ даної машини Тюрінга T і даним словом w встановити, чи зупиниться машина T, якщо на її вхід подати слово w (так звана *проблема зупинки машини Тюрінга*). Записуючи номери символів у двійковій системі числення, можна як саму програму будь-якої машини Тюрінга, так і будь-яке вхідне слово, записати в алфавіті $A = \{\Lambda, a, q, 0, 1, *\}$ (тут Λ, як зазвичай, порожній символ, а символ $*$ вводиться лише для того, щоб було зручно розділяти слова). Тому чіткіше проблему зупинки машини Тюрінга можна сформулювати так:

чи існує така машина Тюрінга M із зовнішнім алфавітом $A = \{\Lambda, a, q, 0, 1, *\}$, що коли на вхід M подати слово $\Pi(T) * w$, де $\Pi(T)$ і w — записані в алфавіті A відповідно програма і вхідне слово деякої машини Тюрінга T, то M видасть на виході 1, якщо машина T зупиняється на вході w, і 0 у протилежному випадку.

Виявляється, що такої машини Тюрінга не існує. Іншими словами, справджується теорема 5.20.

Теорема 5.20 (Тюрінга)**.** *Проблема зупинки машини Тюрінга алгоритмічно нерозв'язна.*

Доведення. Припустимо, що така машина Тюрінга M існує. Програми машин Тюрінга можна занумерувати (напр., за допомогою нумерації Ґеделя). Нехай $n(T)$ — номер машини Тюрінга T. Тоді M на входах $\Pi(T) * n(T)$ розпізнає самозастосовність машин Тюрінга. Оскільки за тезою Черча кожен алгоритм еквівалентний деякій машині Тюрінга, то M розв'язує задачу розпізнавання самозастосовності. Але це суперечить теоремі 5.19. □

Можна розглянути значно вужчий варіант проблеми зупинки машини Тюрінга: коли на вхід подають порожнє слово Λ. Однак і в такому варіанті вона лишається алгоритмічно нерозв'язною.

Твердження 5.5. *Задача зупинки машини Тюрінга на порожньому вході є алгоритмічно нерозв'язною.*

Доведення. Для кожного слова w в певному алфавіті розглянемо машину Тюрінга T_w, яка на порожній стрічці виписує слово w і зупиняється перед його початком. Тоді для кожної машини Тюрінга T і вхідного слова w в її алфавіті композиція $T_w \circ T$ поводить себе на порожньому вході так само, як машина T на вході w. Тому наявність машини M, яка розв'язувала б проблему зупинки машини Тюрінга на порожньому вході, дозволяє легко побудувати машину M', яка розв'язувала б загальну проблему зупинки машини Тюрінга. Оскільки такої машини M' не існує, то проблема зупинки машини Тюрінга на порожньому вході є алгоритмічно нерозв'язною. □

Властивість обчислювальних функцій називається *нетривіальною*, якщо можна вказати як обчислювальну функцію, що має цю властивість, так і обчислювальну функцію, що її не має. Наприклад, властивість бути обчислювальною за Тюрінгом є тривіальною, а бути скрізь визначеною — це нетривіальна властивість.

Дуже важливим прикладом застосування теореми 5.19 є теорема Райса.

Теорема 5.21 (Райса)**.** *Будь-яка нетривіальна властивість обчислювальних функцій є алгоритмічно нерозв'язною* (*точніше — не існує алгоритму, який би за описом функції встановлював, чи має ця функція дану нетривіальну властивість*).

Доведення. Незалежно від того, як задають обчислювальні функції — за допомогою машин Тюрінга, як частково рекурсивні функції тощо, — їх (точніше — їхні описи) можна ефективно перенумерувати:

$$f_0,\ f_1,\ \ldots,\ f_n,\ \ldots. \tag{5.11}$$

Тому теорему можна переформулювати так: якщо $\mathcal{C}$ — непорожній клас обчислювальних функцій, який не збігається з усім класом обчислювальних функцій, то не існує алгоритму, який за номером n функції f_n визначав би, чи належить функція f_n класу $\mathcal{C}$. Іншими словами, треба довести, що множина $M = \{n \mid f_n \in \mathcal{C}\}$ є нерозв'язною.

Припустимо, що множина M є розв'язною, і розглянемо її характеристичну функцію

$$\chi_M(n) = \begin{cases} 1, & \text{якщо } f_n \in \mathcal{C}; \\ 0, & \text{якщо } f_n \notin \mathcal{C}. \end{cases}$$

Нехай g — ніде не визначена функція. Маємо два випадки.

1. $g \notin \mathcal{C}$. Виберемо якусь конкретну функцію $h \in \mathcal{C}$ і визначимо функцію

$$F(n,m) = \begin{cases} h(m), & \text{якщо значення } f_n(n) \text{ визначене;} \\ g(m), & \text{якщо значення } f_n(n) \text{ не визначене.} \end{cases}$$

Функція $F(n,m)$ є обчислювальною. Справді, для її обчислення спочатку треба знайти $f_n(n)$. Якщо це обчислення колись завершиться, то далі слід перейти до обчислення $h(m)$. Якщо ж процес обчислення $f_n(n)$ ніколи не завершиться, то це рівносильне обчисленню ніде не визначеної функції $g(m)$.

Якщо зафіксувати n, то $F(n,m)$ стає обчислювальною функцією від однієї змінної m. Отже, її опис має зустрічатися у послідовності f_0, f_1, …, f_n, … і, перебираючи послідовно всі описи, ми можемо його знайти. Нехай це $f_{t(n)}(m)$. Отже,

$$f_{t(n)}(m) = \begin{cases} h(m), & \text{якщо значення } f_n(n) \text{ визначене;} \\ g(m), & \text{якщо значення } f_n(n) \text{ не визначене.} \end{cases}$$

Функція $t(n)$ є скрізь визначеною обчислювальною функцією. Оскільки $h \in \mathcal{C}$ і $g \notin \mathcal{C}$, то $f_{t(n)} \in \mathcal{C}$ тоді й лише тоді, коли значення $f_n(n)$ визначене. Тому

$$\chi_M(t(n)) = \begin{cases} 1, & \text{якщо значення } f_n(n) \text{ визначене;} \\ 0, & \text{якщо значення } f_n(n) \text{ не визначене.} \end{cases}$$

Але тоді функція $\chi_M(t(n))$ розв'язує проблему розпізнавання самозастосування, що неможливо.

2. $g \in \mathcal{C}$. У цьому разі вибираємо функцію $h \notin \mathcal{C}$ і міркуємо аналогічно попередньому випадку. Проблему розпізнавання самозастосування розв'язуватиме функція $1 - \chi_M(t(n))$. □

Зауваження. *У теоремі Райса фігурують як функції, так і їхні описи, і йдеться про алгоритмічну нерозпізнавальність властивостей саме функцій, а не їхніх описів чи текстів програм. Зрозуміло, що для текстів програм така, наприклад, властивість, як містити не більше 1000 символів, є нетривіальною і алгоритмічно розв'язною. Але це властивість саме текстів програм. А от такі властивості обчислювальних цими програмами функцій, як скінченність чи нескінченність множини значень, періодичність, обмеженість, порядок зростання тощо, є алгоритмічно нерозв'язними. Тому теорема Райса для програмістів має приблизно таке ж значення, як для фізиків твердження про неможливість вічного двигуна. Якщо сказати більш грубо, то теорема Райса стверджує, що за описом алгоритму нічого не можна сказати про властивості функції, яку він обчислює. Точніше, не існує жодного систематичного способу з'ясувати, що саме обчислює цей алгоритм. Щоразу це творча задача, і часто дуже важка. Кожен, хто мав справу із чужою програмою, та ще й без коментарів, знає це з власного досвіду.*

Дуже важливим із погляду обчислювальної практики є наслідок з теореми Райса. Нагадаємо, що алгоритм називають *поліноміальним*, якщо існують такі додатні цілі числа k, C_1 і C_2, що для довільного входу w кількість тактів $n(w)$ роботи відповідної машини Тюрінга на вході w довжини $|w|$ не перевищує $C_1|w|^k + C_2$. Наведемо вказаний наслідок.

Наслідок 5.2. *Множина поліноміальних алгоритмів є перераховною, але нерозв'язною.*

Доведення. Нерозв'язність множини поліноміальних алгоритмів випливає з теореми Райса. Для доведення її перераховності спочатку перерахуємо множину $\mathcal{M}$ усіх машин Тюрінга:

$$M_1,\ M_2,\ \ldots,\ M_n,\ \ldots\ .$$

Кожній четвірці (M_i, k, C_1, C_2) поставимо у відповідність машину Тюрінга $\widetilde{M}(M_i, k, C_1, C_2)$, яка на вході w працює так само, як і машина M_i, якщо остання на цьому вході працює не більше ніж $t(|w|) = C_1|w|^k + C_2$ тактів, і зупиняється через $t(|w|)$ тактів, якщо машина M_i працює на вході w більше

ніж $t(|w|)$ тактів. Зауважимо, що $\widetilde{M}(M_i, k, C_1, C_2)$ завжди є поліноміальною і збігається з M_i, якщо M_i є поліноміальною (для довільного входу w час роботи M_i на вході w не перевищує $C_1|w|^k + C_2$). Тому множина $\widetilde{M}(M_i, k, C_1, C_2)$ збігається з множиною всіх поліноміальних алгоритмів.

Кожен із параметрів k, C_1 і C_2 пробігає множину $\mathbb{N}^+$ додатних цілих чисел. Оскільки множину всіх $\widetilde{M}(M_i, k, C_1, C_2)$ можна ототожнити з множиною $\mathcal{M} \times \mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+$, то її перераховність випливає з твердження 5.4, п.a). $\square$

Список літератури

[1] Дрозд Ю. А. Основи математичної логіки / Ю. А. Дрозд. – К. : ВПЦ "Київський університет", 2005. – 126 с.

[2] Cutland N. Computability, an Introduction to Recursive Function Theory / N. Cutland. – Cambridge University Press, 1980. – X+251 p.

[3] Kleene S. C. Mathematical logic / S. C. Kleene. – Wiley, New York, 1967. – 398 p.

[4] Lyndon Roger C. Notes on Logic / Roger C. Lyndon. – New York, 1966. – vi+97 p.

[5] Mendelson E. Introduction to Mathematical logic. – 4th edition / E. Mendelson. – Chapman & Hall, 2–6 Boundary Row, London SE1 8HN, UK, 1997. – 447 p.

Предметний покажчик

Навчальне видання

Безущак Оксана Омелянівна
Ганюшкін Олександр Григорович

МАТЕМАТИЧНА ЛОГІКА

Навчальний посібник

Редактор *Л. В. Магда*

Оригінал-макет виготовлено ВПЦ "Київський університет"

Формат 70x100$^{1/16}$. Ум. друк. арк. 11,61. Наклад 100. Зам. № 223-10734.
Гарнітура Tempora. Папір офсетний. Друк офсетний. Вид. № М3.
Підписано до друку 29.11.23

Видавець і виготовлювач
ВПЦ "Київський університет"
Б-р Тараса Шевченка, 14, м. Київ, 01601, Україна
☎ (38044) 239 32 22; (38044) 239 31 58; (38044) 239 31 28
e-mail: vpc@knu.ua; vpc_div.chief@univ.net.ua; redaktor@univ.net.ua
http: vpc.knu.ua
Свідоцтво суб'єкта видавничої справи ДК № 1103 від 31.10.02