Zbigniew Huzar

Elementy logiki i teorii mnogości dla informatyków

Oficyna Wydawnicza Politechniki Wrocławskiej
Wrocław 2007

Recenzenci
Marian Adamski
Leszek Pacholski
Wiesław Szwarc

Opracowanie redakcyjne i korekta
Alicja Kordas

Projekt okładki
Zofia i Dariusz Godlewscy

Niniejszy podręcznik jest poprawioną i uzupełnioną wersją, wydanej w 2002 roku, książki „Elementy logiki dla informatyków"

ISBN 978-83-7493-349-0

Oficyna Wydawnicza Politechniki Wrocławskiej
Wybrzeże Wyspiańskiego 27, 53-370 Wrocław
http://www.oficyna.pwr.wroc.pl
oficwyd@pwr.wroc.pl

Drukarnia Oficyny Wydawniczej Politechniki Wrocławskiej. Zam. nr 710/2007

Spis treści

Przedmowa

Informatyka jest dyscypliną młodą, liczącą około pięćdziesiąt lat. Sam termin *informatyka* pojawił się w języku polskim na początku lat siedemdziesiątych, a termin *komputer* zadomowił się na dobre dopiero w końcu lat siedemdziesiątych ubiegłego wieku. Rozwój informatyki był i pozostaje stymulowany dwoma czynnikami. Pierwszym jest rozwój technologii, głównie elektronicznej. Dzięki postępowi w tej dziedzinie stało się możliwe technicznie zrealizowanie najpierw urządzeń liczących, których koncepcje były rozważane znacznie wcześniej, a później zbudowanie uniwersalnych urządzeń liczących – współczesnych komputerów. Drugim czynnikiem jest potencjalnie ogromne pole zastosowań informatyki. Oba te wzajemnie sprzężone czynniki doprowadziły do sytuacji, że komputer staje się powszechnym narzędziem pracy niemal w każdej dziedzinie.

Miarą tempa rozwoju obecnie dominującej technologii półprzewodnikowej jest fakt, że wydajność sprzętu komputerowego, mierzona częstotliwością zegara sterującego pracą komputera, i – podobnie – rozmiar pamięci operacyjnej komputera podwaja się co półtora roku. Przewiduje się, że taka tendencja może się utrzymać do lat 2015–2020. Wyznacznikiem rozpowszechnienia zastosowań informatyki jest obecnie nie tylko Internet – globalna sieć komputerowa, stanowiąca federację setek tysięcy sieci komputerowych – ale również rozpoczynający się proces integracji Internetu, telefonii komórkowej oraz telewizji cyfrowej.

Informatyka dostarcza specyficznych narzędzi i metod, które można wykorzystywać do rozwiązywania problemów w różnych dziedzinach. Do zrozumienia tej specyfiki i możliwości zastosowań informatyki potrzeba trwałych i niezawodnych podstaw. Tak jak w przypadku innych nauk ścisłych, podstawy informatyki są oparte na matematyce, a dokładniej na wybranych jej działach, odpowiednio przystosowanych do potrzeb informatyki. Podstawy informatyki są wprawdzie ciągle kształtowane, ale pewne ich elementy można obecnie uznać za ustabilizowane. W historii matematyki był okres na przełomie XIX i XX wieku, gdy uświadomiono sobie konieczność ustalenia podstaw matematyki, bez których nie byłby możliwy spójny rozwój różnych działów: algebry, analizy matematycznej, rachunku prawdopodobieństwa, topologii itp. Podstawy matematyki, które uformowały się w pierwszej połowie ubiegłego wieku, objęły dwa niezależne wcześniej działy – teorię mnogości i logikę. Podobnie jest z podstawami informatyki, za które również można uważać teorię mnogości i logikę, z położeniem

akcentu, silniejszego niż w podstawach matematyki, na logikę. Wynika to także z tego, że informatykę traktuje się niekiedy jako dyscyplinę wyrosłą z podstaw matematyki.

Szczególna rola matematyki w podstawach informatyki nie jest jednak uzasadniona wyłącznie względami historycznymi. Zasadniczy powód wynika z roli, jaką pełni informatyka w zastosowaniach praktycznych. Rozwiązywanie problemów, przed którymi staje informatyk, wymaga od niego – po pierwsze – zrozumienia danej dziedziny zastosowań i zrozumienia na czym dany problem polega, po wtóre – informatyk musi znać i rozumieć narzędzia i metody, którymi może dysponować, wreszcie – po trzecie – musi zaproponować jak, za pomocą posiadanych środków, dany problem rozwiązać. Opis problemu na tle specyficznej dziedziny jest początkowo wyrażany w języku naturalnym. Precyzyjny jego opis wymaga wyrażenia go w języku sformalizowanym, czyli języku o ściśle określonej składni i semantyce. Potrzeba taka wynika stąd, że przedstawione ostatecznie komputerowi do policzenia rozwiązanie problemu musi być wyrażone w języku programowania, czyli również pewnym sformalizowanym języku. Komputer, w odróżnieniu od człowieka, nie potrafi bowiem podjąć żadnych innych akcji niż te, które są wyrażone w pewnym języku sformalizowanym.

Konieczność formalizacji informatyki wynika więc z dwóch powodów. Po pierwsze – ze specyfiki komputera i tego, co potrafi, a to – co potrafi – można sprowadzić do umiejętności przetwarzania symboli. Po drugie – wynika z potrzeby zrozumienia abstrakcji, czyli procesu budowy formalnego opisu problemu na podstawie opisu wyrażonego w języku naturalnym. Formalny opis problemu jest pewnym modelem rzeczywistego problemu występującego w realnej dziedzinie. Model jest opisem uproszczonym, to znaczy koncentruje się tylko na wybranych aspektach rzeczywistego problemu. Na jakich aspektach i jak szczegółowo ma skupiać się model zależy oczywiście od celu jego budowy. Sposób budowy modelu, ocena zgodności modelu z opisywaną rzeczywistością, związek pomiędzy modelem opisu a modelem rozwiązania są typowymi zagadnieniami, wokół których wyrastają teorie i działy matematyki. Klasycznym przykładem są analiza matematyczna i teoria równań różniczkowych, które rozwinęły się na skutek zapotrzebowania dziewiętnastowiecznej techniki i fizyki.

Na początku XX wieku z formalizacją matematyki wiązano zbyt wielkie nadzieje. Program formalizacji matematyki, związany głównie z nazwiskiem Dawida Hilberta, załamał się w latach trzydziestych ubiegłego wieku, po odkryciach Kurta Gödla, który wskazał na swoistą ograniczoność metod formalnych. Gdyby się okazało, że program Hilberta jest realizowalny, wówczas można byłoby przypuszczać, że wszystko to, co potrafi człowiek, mógłby również zrealizować komputer. Tak jednak nie jest, dlatego intuicja i kreatywność są tymi właściwościami człowieka, które stanowią o jego przewadze nad komputerem. Oznacza to, że informatyk w swojej pracy powinien traktować metody formalne jako uzupełnienie i wsparcie własnej pomysłowości i twórczości.

W informatyce metody formalne stanowią fundament podstawowych pojęć, takich jak: pojęcie algorytmu, obliczalności czy złożoności obliczeniowej.

Logika dostarcza języka do przedstawiania i badania własności modeli informatycznych, w tym systemów komputerowych i języków programowania, a zwłaszcza środków do definiowania składni i semantyki języków programowania. W języku logiki można specyfikować wymagania stawiane projektowanym systemom oprogramowania. Język logiki może ponadto być bezpośrednio używany jako język programowania. Ogromną rolę odgrywa logika w zastosowaniach informatyki, na przykład w tworzeniu i funkcjonowaniu baz wiedzy i systemów ekspertowych.

Szczególną rolę odgrywa logika w procesie wytwarzania oprogramowania. Na gruncie logiki stało się możliwe sformułowanie pojęcia poprawności programów, a następnie opracowanie metod weryfikacji ich poprawności. Proces wytwarzania oprogramowania, jak na przykład w inżynierii oprogramowania, jest obecnie w coraz większym zakresie wspomagany przez komputer. Budowa narzędzi wspomagających ten proces opiera się na formalnych metodach, mających oparcie na gruncie logiki.

Niniejszy podręcznik pt. *Elementy logiki i teorii mnogości dla informatyków* jest poprawioną i rozszerzoną wersją wydania z 2002 roku: *Elementy logiki dla informatyków*, Oficyna Wydawnicza Politechniki Wrocławskiej. Tak jak wydanie poprzednie, obejmuje głównie logikę klasyczną wraz z krótkimi informacjami o logikach nieklasycznych.

W języku polskim jest wiele bardzo dobrych podręczników logiki, pisanych przede wszystkim dla matematyków – na przykład: [Adamowicz, Zbierski 1991], [Grzegorczyk 1975], [Hunter 1982], [Rasiowa 1998], [Słupecki, Hałkowska, Piróg-Rzepecka 1978], a w ostatnim okresie: skrypt [Tiuryn 2003] oraz [Guzicki, Zakrzewski 2005], [Kraszewski 2007], przy czym dwie ostatnie pozycje ograniczają się tylko do teorii mnogości. Warto wspomnieć o krótkiej i dawno wydanej książce o przeglądowym charakterze [Lyndon 1978]. Natomiast, oprócz nielicznych – na przykład: [Mostowski, Pawlak 1970], [Szałas 1992] – praktycznie nie ma takich podręczników dla informatyków. Zwraca uwagę fakt, że w ostatnim okresie powstają różne podręczniki logiki dla informatyków w języku angielskim – na przykład: [Fitting 1990], [Kelly 1997], [Nissanke 1999], [Socher-Ambrosius, Johann 1996], [Ben-Ari 2001]. Ostatnia pozycja – pod wieloma względami podobna do niniejszego podręcznika – ukazała się w 2005 roku w tłumaczeniu na język polski. Jedną z istotnych różnic między tymi dwiema kategoriami podręczników jest sposób przedstawiania systemów dowodzenia. Dla matematyków zwykle jako podstawowy wybiera się system dowodzenia Hilberta, który dobrze oddaje praktykę dowodową „klasycznego" matematyka, w informatyce natomiast większą rolę odgrywają systemy dowodowe, które – inaczej niż system Hilberta – pozwalają na automatyzowanie procesu dowodzenia. W niniejszym podręczniku omawia się za-

tem – jako podstawowe systemy dowodzenia – system sekwentów Gentzena i system oparty na regule rezolucji.

Pierwszy rozdział podręcznika jest ogólnym wprowadzeniem, wyjaśniającym czym jest logika. Pozostały materiał można podzielić na cztery części.

Pierwsza część, obejmująca rozdziały od 2. do 8., jest krótką prezentacją elementów teorii mnogości, algebr abstrakcyjnych i języków formalnych. W obecnym wydaniu książki wprowadzono dodatkowe informacje na temat liczb kardynalnych oraz nieco poszerzono rozdział dotyczący języków formalnych.

W drugiej części, obejmującej rozdziały od 9. do 12., omówiono rachunek zdań i kwantyfikatorów – ich składnię, semantykę oraz związane z nimi systemy dowodzenia oparte na sekwentach Gentzena i regule rezolucji.

W trzeciej części, obejmującej rozdziały 13. i 14. o charakterze informacyjnym, krótko omówiono inne systemy dowodzenia oraz dokonano przeglądu innych, nieklasycznych logik. W obecnym wydaniu zamieszczono opis systemu dowodzenia opartego na tablicach analitycznych, jako systemu alternatywnego w stosunku do systemu dowodzenia opartego na regule rezolucji.

Część czwarta – dołączona do tego wydania książki, obejmująca rozdziały 15. i 16. – przedstawia zastosowanie metod logiki do definiowania składni i semantyki języków programowania oraz klasyczną logikę programów Hoare'a, służącą dowodzeniu poprawności programów.

Prezentacja materiału jest sformalizowana tylko częściowo i odnosi się w zasadzie do definiowania pojęć oraz do sformułowania i udowodnienia wybranych twierdzeń. Zwrócono uwagę przede wszystkim na twierdzenia o poprawności i zupełności systemu dowodzenia opartego na rachunku sekwentów Gentzena, w przypadku pozostałych przedstawianych systemów dowodzenia ograniczono się tylko do sformułowania odpowiednich twierdzeń.

Do każdego rozdziału są dołączone ćwiczenia. Zebrane tu zadania pochodzą z różnych źródeł: część stanowi opracowania autora lub współpracowników, część jest zaczerpnięta z podręczników przedstawionych w spisie literatury: [Fitting 1990], [Gabbay 1998], [Gerstig 1993], [Kelly 1997], [Marek, Onyszkiewicz 1975], [Nissanke 1999], [Stanosz 2002], [Pacholski 2004], [Ławrow, Maksimowa 2004], [Guzicki, Zakrzewski 2005].

W celu czytelnego wyodrębnienia przykładów wprowadzono w tekście linie rozdzielające na początku i końcu odpowiednich akapitów. Zakończenia dowodów są zaznaczone czarnym kwadracikiem ■.

Podręcznik jest przeznaczony w zasadzie dla studentów pierwszego roku informatyki na studiach politechnicznych. Zakres materiału jest jednak szerszy i dlatego mogą skorzystać z niego, jako z lektury uzupełniającej, także studenci lat starszych.

Składam serdeczne podziękowania moim kolegom z Instytutu Informatyki Stosowanej Politechniki Wrocławskiej za pomoc podczas przygotowywania tego podręcznika. Szczególnie gorąco dziękuję profesorowi Iwanowi Tabakowowi oraz doktorowi Zdzisławowi Spławskiemu za uważną lekturę rękopisu, wskazanie usterek, cenne wskazówki oraz propozycje ulepszenia tekstu, a doktor Bogumile Hnatkowskiej – za wnikliwe uwagi dotyczące dwóch ostatnich rozdziałów.

Oddzielne podziękowanie kieruję do recenzentów – profesora Mariana Adamskiego, profesora Leszka Pacholskiego i doktora hab. Wiesława Szwarca, którzy – oprócz wskazania usterek – przedstawili sugestie i propozycje dotyczące sposobu prezentacji materiału.

Dziękuję też studentom Wydziału Informatyki i Zarządzania Politechniki Wrocławskiej, którzy wychwycili usterki edytorskie w poprzednim wydaniu książki.

Niezależnie od wszelkich uwag i sugestii, podręcznik, jak każdy obszerniejszy tekst, może zawierać przeoczenia bądź nieścisłości. Czytelników, którzy spostrzegą takie usterki, autor prosi o przekazanie odpowiedniej informacji pocztą elektroniczną na adres: zbigniew.huzar@pwr.wroc.pl

Wrocław, maj 2007 Zbigniew Huzar

1. Elementarne pojęcia logiczne

1.1. Czym jest logika?

Słowo *logika*[1] bywa używane przez filozofów, matematyków i w mowie potocznej w licznych znaczeniach i kontekstach. Długotrwała tradycja terminologiczna określa logikę jako analizę języka pod kątem jego wykorzystania do:

- definiowania,
- klasyfikowania,
- wnioskowania.

Celem takiej analizy jest podanie reguł posługiwania się językiem, aby był on skuteczny.

Logika pojmowana jako narzędzie poprawnego myślenia, czyli wnioskowania lub rozumowania, była już przedmiotem zainteresowania starożytnych[2] [Kotarbiński 1985], [Murawski 1995]. Drugie jej narodziny przypadają na wiek XIX. Traktowana jako pomocniczy dział matematyki, wyodrębniła się na początku XX wieku w samodzielną dyscyplinę matematyki. Obecnie zakres pojęcia logiki jest szeroki i obejmuje trzy odrębne dziedziny [Bocheński 1992]:

- logikę formalną,
- metodologię,
- filozofię logiki.

Przedmiotem zainteresowania *logiki formalnej* są *wypowiedzi* w danym języku, a dokładniej to, czy są one prawdziwe czy fałszywe. Daną wypowiedź można oceniać albo jako prawdziwą, albo jako fałszywą, gdyż żadna wypowiedź nie może być jednocześnie prawdziwa i fałszywa. *Prawda* i *fałsz*, jako własności wypowiedzi, są zatem podstawowymi pojęciami logiki.

Pojęcie prawdy, chociaż używane powszechnie, nie jest łatwe do określenia. Klasyczne rozumienie prawdy opiera się na związku pomiędzy wypowiedzią a rzeczywistością, do której dana wypowiedź się odnosi. Ten sens oddają słowa wypowiedziane blisko dwa tysiące lat temu przez Sekstusa Empiryka [Turski 1988]:

> *O każdym bowiem zdaniu rozstrzyga się, że jest prawdziwe albo że jest fałszywe ze względu na jego odniesienie do rzeczy, o której zostało orzeczone. Jeżeli bowiem okazuje się ono zgodne z rzeczą, o której zostało orzeczone, wydaje się prawdziwe, jeżeli niezgodne – fałszywe.*

Zadaniem logiki formalnej jest ustalanie prawdziwości wypowiedzi. Pierwszym zadaniem jest ustalanie prawdziwości wypowiedzi złożonych na podstawie prawdziwości wypowiedzi, które stanowią ich składowe. Szczególnym rodzajem są takie wypowiedzi złożone, które są zawsze prawdziwe, niezależnie od prawdziwości swoich wypowiedzi składowych. Wypowiedzi takie nazywa się prawami logicznymi. Głównym zadaniem logiki jest jednak wnioskowanie, czyli badanie tego, co na podstawie danego zestawu prawdziwych wypowiedzi – przesłanek – można sądzić o prawdziwości innych wypowiedzi. Chodzi tu o wnioskowanie niezawodne, to znaczy takie, które na podstawie prawdziwych przesłanek gwarantuje zawsze wyprowadzenie prawdziwych wniosków. Przedmiotem logiki są różnego rodzaju schematy niezawodnego wnioskowania, ich formułowanie, porządkowanie i uzasadnianie.

Metodologia zajmuje się stosowaniem logiki do różnych dziedzin [Bocheński 1992], [Wójcicki 1982]. W praktyce okazuje się, że te same prawa logiczne mogą być stosowane w różny sposób. Inną rzeczą są schematy wnioskowania, a inną przeprowadzanie wnioskowania na podstawie tych schematów. Znany na przykład podział wnioskowania na metody dedukcyjne i indukcyjne nie polega na użyciu różnych praw logiki, lecz na różnym użyciu tych samych praw. Celem metodologii – nie wnikając w szczegóły – są ogólne sposoby zdobywania i formułowania wiedzy prawdziwej albo przynajmniej dobrze uzasadnionej.

Filozofia logiki obejmuje analizę podstawowych pojęć logiki [Bocheński 1992]. Próbuje odpowiadać na przykład na pytania: *Co to jest prawda? Co to jest prawo logiczne? Skąd wiadomo, że jest ono prawdziwe?*

Książka obejmuje tylko *logikę formalną*, nazywaną inaczej *logiką matematyczną* lub *logiką symboliczną*. Logika formalna wprowadza język symbolicznego zapisu wypowiedzi i określa jak można takim symbolicznym zapisom przypisywać pewne znaczenie, czyli – w jaki sposób można określać ich *semantykę*? Zakres wypowiedzi, które można zapisywać w języku logiki formalnej, nie obejmuje oczywiście wszystkich wypowiedzi, które można sformułować w języku naturalnym. Język logiki formalnej jest natomiast całkowicie wystarczający do przedstawiania i analizy wypowiedzi dowolnych działów matematyki. Nic w tym dziwnego, gdyż narodziny współczesnej logiki wiążą się właśnie z potrzebą precyzyjnego sformułowania i analizy zagadnień z zakresu podstaw matematyki, które pojawiły się pod koniec XIX i na początku XX wieku. Dlatego logikę formalną określa się niekiedy jako *metamatematykę*, czyli jako naukę dostarczającą języka do opisu wszystkich pozostałych działów matematyki.

[1] Słowo *logika* pojawia się po raz pierwszy w tytule dzieła Demokryta (460–371 p.n.e.).

[2] Problematyka logiczna była rozważana przez Sokratesa (469–399 p.n.e.) i Platona (427–347 p.n.e.), ale za pierwszego twórcę systemu logiki uważa się Arystotelesa (384–322 p.n.e.).

Celem logiki formalnej jest ujęcie procesu rozumowania, albo wnioskowania, w postaci *przekształcania napisów* reprezentujących wypowiedzi. Chodzi o to, aby na podstawie pewnych napisów, reprezentujących wypowiedzi uznane za prawdziwe, uzyskiwać prawdziwe wnioski – nowe napisy, reprezentujące nowe, prawdziwe wypowiedzi. Inaczej: chodzi o to, aby przekształcania napisów reprezentowały niezawodne schematy wnioskowania.

Przekształcanie napisów opiera logika formalna na *systemie dedukcyjnym*, czyli na ustalonym zbiorze reguł mechanicznego przekształcania tekstów. Pewne napisy przyjmuje się za poprawne z założenia. Traktuje się je jako *aksjomaty* systemu dedukcyjnego. Inne napisy przyjmuje się za poprawne tylko wtedy, gdy daje się je wyprowadzić z aksjomatów przez stosowanie ustalonych *reguł* systemu dedukcyjnego. Reguła jest mechanicznym sposobem przekształcania jednych napisów w inne napisy. Napisy wyprowadzone w wyniku stosowania przyjętych reguł powinny być poprawne nie tylko w sensie zgodności z przyjętymi regułami przekształcania, ale również powinny być poprawne w sensie semantycznym, to znaczy powinny być wypowiedziami prawdziwymi.

Logik formalnych jest wiele [Marciszewski 1987, 1988]. Różnią się one klasą obiektów, do których odnoszą się wypowiedzi, rodzajami wypowiedzi (np. wypowiedzi oznajmujące, przypuszczające, pytające, nakazujące) oraz stosowanymi systemami dedukcyjnymi – systemami wnioskowania. Szczególną rolę – zarówno ze względu na historię, a także zastosowania – pełni *logika klasyczna*. Logika klasyczna jest jądrem wszystkich innych logik formalnych, w tym również różnych specjalistycznych logik stosowanych w informatyce.

1.2. Język logiki formalnej

Jak wspomniano, przedmiotem logiki formalnej są *wypowiedzi* w danym języku, a dokładniej to: czy są prawdziwe czy fałszywe.

Nie wszystkie wypowiedzi mogą być jednak oceniane jako prawdziwe albo fałszywe. Nie sposób tak ocenić wypowiedzi rozkazującej czy pytającej, można tak oceniać co najwyżej wypowiedzi oznajmujące, ale nawet co do nich mogą powstawać wątpliwości. Na przykład, czy wypowiedź:

W 2100 roku bardzo popularną formą wypoczynku będą wakacje na Marsie.

jest prawdziwa czy fałszywa? Trudno to osądzić, przynajmniej w obecnej dobie. Z powodu braku wiedzy historycznej nie można natomiast stwierdzić, czy prawdziwa jest wypowiedź:

Król Bolesław Chrobry urodził się w poniedziałek.

Wypowiedzi, którym można przypisać prawdziwość albo fałszywość, będą nazywane *zdaniami*. Zdania mogą być proste, na przykład:

Książka leży na stole.
W programie koncertu jest symfonia Mahlera.

Warto zwrócić uwagę, że tego rodzaju zdania są formułowane w pewnym kontekście sytuacyjnym i tylko w tym kontekście można rozstrzygać, czy są prawdziwe czy fałszywe. W języku naturalnym spotyka się też wypowiedzi, których prawdziwość, nawet po ustaleniu kontekstu, może być trudna do określenia. Rozpatrzmy zdania:

On jest dosyć wysokim mężczyzną.
Samochód jechał dosyć wolno.

Powodem trudności w pierwszym zdaniu jest rozumienie zwrotu *dosyć wysoki*. Czy jest dosyć wysoki mężczyzna, który ma 180 cm wzrostu, czy dopiero taki, który ma 185 cm? Podobnie w drugim zdaniu problem stwarza rozumienie zwrotu *jechać dosyć wolno*.

Ze zdań prostych można budować zdania złożone, na przykład:

*Pójdę do kina **lub** pójdę do teatru.*
***Jeżeli** wykonawcą koncertu będą filharmonicy berlińscy, **to** zwalą się tłumy.*
*W 1939 roku Hitler napadł na Czechosłowację **i** – w roku następnym – na Polskę.*

Zdania złożone powstają przez połączenie zdań prostych za pomocą spójników logicznych. Spójnikami logicznymi (zdaniowymi) są na przykład słowa i zwroty: *nie*, *lub*, *i* (*oraz*), *jeżeli ..., to ...*, *... wtedy i tylko wtedy, gdy ...* .

W języku naturalnym zwroty te mają ustalone znaczenie. Poniżej przedstawia się prostą formalizację uściślającą ich znaczenie. Formalizacja spójników logicznym polega na:

- nadaniu im pewnej symbolicznej notacji,
- przypisaniu im znaczenia w terminach tablic prawdziwościowych.

Zdania będą oznaczane symbolami p, q, r, ... Spójniki logiczne będą oznaczane następująco:

- Spójnik *nie* – nazywany *negacją* – jest oznaczany symbolem $\neg$. Negację zdania p zapisujemy: $\neg p$.
- Spójnik *i* (*oraz*) – nazywany *koniunkcją* – jest oznaczany symbolem $\wedge$. Koniunkcję zdań p, q zapisujemy: $p \wedge q$.
- Spójnik *lub* – nazywany *dysjunkcją* lub *alternatywą* – jest oznaczany symbolem $\vee$. Dysjunkcję (alternatywę) zdań p, q zapisujemy: $p \vee q$.
- Spójnik *jeżeli ..., to ...* – nazywany *implikacją* – jest oznaczany symbolem $\Rightarrow$. Implikację zdań p, q zapisujemy: $p \Rightarrow q$.

  Implikację $p \Rightarrow q$ można także czytać w inny sposób, na przykład:

  p jest warunkiem dostatecznym do tego, że q.
  q pod warunkiem, że p.
  q wtedy, gdy p.
  q jest warunkiem koniecznym do tego, że p.

- Spójnik *wtedy i tylko wtedy, gdy* – nazywany *równoważnością* – jest oznaczany symbolem $\Leftrightarrow$. Równoważność zdań p, q zapisujemy: $p \Leftrightarrow q$.

  Równoważność $p \Leftrightarrow q$ można także czytać w inny sposób, na przykład:

  p jest warunkiem koniecznym i wystarczającym do tego, że q.
  p dokładnie wtedy, gdy p.
  p jest równoważne p.

Uwaga

Czasem negacją, koniunkcją itd. nazywa się zdania złożone, które powstają z innych zdań przez łączenie spójnikami negacji, koniunkcji itd.

Zapisując zdania złożone w postaci symbolicznej, będziemy używać nawiasów, grupując w odpowiedni sposób zdania składowe. Nawiasy będą opuszczane, gdy przyjmie się następującą kolejność stosowania (wiązania) spójników (od najsilniejszego do najsłabszego):

$$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow.$$

Zamiast na przykład

$$((\neg p) \wedge q) \vee (r \wedge s)) \Rightarrow t$$

można pisać

$$\neg p \wedge q \vee r \wedge s \Rightarrow t$$

Zakłada się ponadto, że występujące obok siebie spójniki $\wedge$, $\vee$ łączą w lewo, a występujące obok siebie spójniki $\Rightarrow$, $\Leftrightarrow$ łączą w prawo. Na przykład:

$p \wedge q \wedge r$ znaczy $(p \wedge q) \wedge r$,

$p \Rightarrow q \Rightarrow r$ znaczy $p \Rightarrow (q \Rightarrow r)$.

Prawdziwość zdania złożonego zależy tylko od prawdziwości jego zdań składowych i od tego, jakim spójnikiem są one połączone. Taką własność nazywa się *ekstensjonalnością*.

Tablica 1.1

p	$\neg p$
P	F
F	P

p	q	$p \wedge q$
F	F	F
P	F	F
F	P	F
P	P	P

p	q	$p \vee q$
F	F	F
P	F	P
F	P	P
P	P	P

p	q	$p \Rightarrow q$
F	F	P
P	F	F
F	P	P
P	P	P

p	q	$p \Leftrightarrow q$
F	F	P
P	F	F
F	P	F
P	P	P

Rolę spójników logicznych daje się prosto wyrazić za pomocą tablic prawdziwościowych – tablica 1.1. Tablica prawdziwościowa jest tabelarycznym zestawieniem wszystkich wartościowań zdań składowych oraz odpowiadających im wartościowaniom zdania złożonego połączonego danym spójnikiem logicznym. W celu zmniejszenia rozmiarów tablicy zamiast *prawda* lub *fałsz* pisze się symbole **P** oraz **F**. Tablica uściśla znaczenie, które się wiąże ze spójnikami logicznymi w języku naturalnym.

Własność ekstensjonalności, czyli abstrahowanie od wewnętrznych treści zdań składowych przy ocenie prawdziwości zdań złożonych, może powodować kolizję z potocznym rozumieniem prawdziwości zdań. Typowym przykładem są zdania połączone spójnikiem implikacji. Zdanie

Jeżeli księżyc ma kształt sześcianu, to dzisiaj mamy dzień rektorski.

uznalibyśmy za bezsensowne. Formalnie jest to zdanie poprawnie zbudowane, a ponadto jest to zdanie prawdziwe. Chociaż zdanie *dzisiaj mamy dzień rektorski* nie musi być zdaniem prawdziwym, ale fałszywość zdania *księżyc ma kształt sześcianu* pociąga prawdziwość całej wypowiedzi. Pojęcie sensowności, do którego często się odwołujemy w języku naturalnym, nie ma bezpośredniego odpowiednika w języku logiki klasycznej. Wynika to z tego, że język logiki klasycznej jest znacznie uboższy od języka naturalnego – jest tylko pewnym jego przybliżeniem.

Zdania są wypowiedziami, którym – w danym kontekście wypowiedzi – jednoznacznie przypisuje się prawdziwość lub fałsz. Znane są też inne rodzaje wypowiedzi, którym prawdziwość lub fałsz można przypisać dopiero po dodatkowych uściśleniach dotyczących elementów wypowiedzi. Na przykład o prawdziwości zdania

Mężczyzna jest wyższy od kobiety.

można jednoznacznie się wypowiedzieć dopiero wtedy, gdy wiadomo, o którego mężczyznę i o którą kobietę chodzi. Mężczyzna i kobieta stanowią tu argumenty wypowiedzi. Wskazując na konkretnego mężczyznę i na konkretną kobietę, można stwierdzać o prawdziwości lub fałszu tego zdania.

Zdania tego rodzaju nazywa się *funkcjami zdaniowymi* albo *formami zdaniowymi*. Można je traktować jako pewien sposób wyrażania *własności* elementów pewnego zbioru. Zdania takie będą zapisywane $P(a)$, gdzie a jest argumentem wypowiedzi.

Funkcji zdaniowych często się używa w powiązaniu z charakterystycznymi zwrotami, na przykład:

Możliwe, że zachodzi $P(a)$.
Dla każdego elementu a ze zbioru A zachodzi $P(a)$.

Pierwszy ze zwrotów to rodzaj zwrotu *modalnego*. Taki zwrot występuje na przykład w zdaniach:

Możliwe, że Piotr wypożyczył już potrzebną mu książkę.
Możliwe, że prezes spóźni się na spotkanie.
Możliwe, że w 2100 roku bardzo popularną formą wypoczynku będą wakacje na Marsie.

Drugi ze zwrotów to rodzaj zwrotu *kwantyfikacyjnego*. Przykłady wypowiedzi z tym zwrotem:

Każdy student otrzymuje indeks.
Każdy dorosły ponosi pełną odpowiedzialność za swoje czyny.

Dalej zajmiemy się przede wszystkim zwrotami kwantyfikacyjnymi. Będą rozważane tylko dwa zwroty:

Dla każdego elementu a zachodzi P(a).
Istnieje element a, dla którego zachodzi P(a).

Pierwszy zwrot jest nazywany *kwantyfikacją ogólną* albo *uniwersalną*, a drugi – *kwantyfikacją szczegółową* albo *egzystencjalną*.

Istnieją jeszcze inne rodzaje zwrotów kwantyfikacyjnych, które nie będą rozważane, na przykład:

Dla większości elementów a ze zbioru A zachodzi P(a).
Dla nieskończenie wielu elementów a ze zbioru A zachodzi P(a).

Jeżeli symbolem $P(a)$ oznaczyć funkcję zdaniową, której dla ustalonego elementu a ze zbioru A można w jednoznaczny sposób przyporządkować prawdę albo fałsz, to wypowiedź z kwantyfikatorem ogólnym dla $P(a)$ zapisuje się symbolicznie w postaci

$$\forall a \in A \bullet P(a),$$

a wypowiedź z kwantyfikatorem szczegółowym w postaci

$$\exists a \in A \bullet P(a).$$

Wypowiedzi z kwantyfikatorami można czytać w różny sposób.

Zapis z kwantyfikatorem ogólnym $\forall a \in A \bullet P(a)$ można czytać:

Dla każdego $a \in A$ [zachodzi] $P(a)$.
Dla dowolnego $a \in A$ [zachodzi] $P(a)$.
Dla wszystkich $a \in A$ [zachodzi] $P(a)$.
Wszystkie $a \in A$ mają własność $P(a)$.
$P(a)$ dla wszystkich $a \in A$.

Zapis z kwantyfikatorem szczegółowym $\exists a \in A \bullet P(a)$ można czytać:

Dla pewnego $a \in A$ [zachodzi] $P(a)$.
Dla jakiegoś $a \in A$ [zachodzi] $P(a)$.
Jakieś $a \in A$ ma własność $P(a)$.
$P(a)$ dla pewnego $a \in A$.

Zapis [zachodzi] oznacza tu, że słowo 'zachodzi' występuje opcjonalnie – można je czytać albo pomijać.

Uwaga

Oprócz wprowadzonej, używa się również innych notacji na wypowiedzi z kwantyfikatorami, na przykład:

$\forall a \in A.P(a)$	$\forall a \in A:P(a)$	$(\forall a \in A)P(a)$	$\bigwedge_{a \in A} P(a)$
$\exists a \in A.P(a)$	$\exists a \in A:P(a)$	$(\exists a \in A)P(a)$	$\bigvee_{a \in A} P(a)$

1.3. Wnioskowanie

Logika formalna zajmuje się schematami wnioskowania, które pozwalają na to, aby na podstawie prawdziwości jednych wypowiedzi – *przesłanek* – wnioskować o prawdziwości innych wypowiedzi – *wniosków*. Historycznie najstarsze schematy wnioskowania, nazywane *sylogizmami*, pochodzą od Arystotelesa.

Przykładem wnioskowania opartego na jednym z sylogizmów jest następujące rozumowanie:

Wszyscy bogowie greccy są zazdrośni.
Zeus jest greckim bogiem.
Zatem: *Zeus jest zazdrosny.*

Dwa pierwsze zdania są tu przesłankami, a ostatnie – wnioskiem (konkluzją).

Ogólnie *schemat wnioskowania* można przedstawić w postaci „ułamka", w którego „liczniku" będą zapisywane przesłanki, a w „mianowniku" będą zapisywane wnioski. Rozpatrzmy kilka schematów wnioskowań, które można odnieść do wielu codziennych sytuacji.

Znanym schematem jest *modus ponendo ponens*, mający postać

$$\frac{p \Rightarrow q \qquad p}{q}$$

gdzie: p oraz q oznaczają dowolne wypowiedzi.

Na podstawie takiego schematu wnioskuje się na przykład:

***Jeżeli** dzisiaj jest niedziela, **to** jutro jest poniedziałek*
Dzisiaj jest niedziela.
Jutro jest poniedziałek.

Schemat ten jest niezawodny, co oznacza, że jeżeli przesłanki są prawdziwe, to także prawdziwy jest wyprowadzony na ich podstawie wniosek.

Inny przykład również niezawodnego schematu wnioskowania to *modus ponendo tollens*

$$\frac{\textit{Albo } p, \textit{ albo } q \qquad p}{\neg q}$$

Zwrot ***albo*** ..., ***albo*** ... nie był wcześniej omówiony. Zgodnie z potocznym rozumieniem zdanie złożone postaci ***albo*** p, ***albo*** q jest prawdziwe tylko wtedy, gdy jest prawdziwe dokładnie jedno ze zdań składowych p, q.

Przykład wnioskowania:

*Albo pójdę do kina, **albo** pójdę do teatru.*
Pójdę do kina.
Nie pójdę do teatru.

Jeszcze inny przykład niezawodnego schematu wnioskowania to *modus tollendo ponens*:

$$\frac{p \vee q \qquad \neg p}{q}$$

Według tego schematu wnioskuje się w przykładzie:

*Pójdę do kina **lub** pójdę do teatru.*
Nie pójdę do kina.
Pójdę do teatru.

Często korzysta się ze schematów wnioskowania nazywanych *sylogizmem warunkowym*. Przykładem jest schemat:

$$\frac{p \Rightarrow q \qquad q \Rightarrow r}{p \Rightarrow r}$$

Schematy wnioskowania, którymi zajmuje się logika formalna, są w pewien sposób ograniczone. Nie biorą pod uwagę treści, lecz tylko prawdziwość zdań, dlatego mechaniczne stosowanie przedstawionych schematów wnioskowania, jeżeli się nie wnika w treść zdań, może prowadzić do absurdalnych wniosków. Jako przykład posłuży następujące rozumowanie: Niech będą dane dwie wypowiedzi:

***Jeżeli** Cezar pozostanie w domu, **to** Cezar nie zostanie zabity przez spiskowców.*

oraz

***Jeżeli** Cezar nie zostanie zabity przez spiskowców, **to** Cezar wygłosi przemówienie w Senacie.*

Wprowadźmy oznaczenia. Niech:

p oznacza: *Cezar pozostanie w domu.*
q oznacza: *Cezar nie zostanie zabity przez spiskowców.*
r oznacza: *Cezar wygłosi przemówienie w Senacie.*

Z zastosowaniem tych oznaczeń wnioskowanie oparte na schemacie sylogizmu warunkowego przebiega następująco: Na podstawie przesłanek $p \Rightarrow q$ oraz $q \Rightarrow r$ otrzymuje się wniosek $p \Rightarrow r$, czyli:

***Jeżeli** Cezar pozostanie w domu, **to** Cezar wygłosi przemówienie w Senacie.*

Wniosek ten jest całkowicie sprzeczny ze zdrowym rozsądkiem. Wynika to z tego, że w zastosowanym schemacie wnioskowania uwzględnia się tylko prawdziwość przesłanek, a nie uwzględnia się treściowego powiązania przesłanek i konkluzji: pozostawanie w domu w pewnym okresie wyklucza przebywanie w Senacie w tym samym okresie i, tym samym, wygłoszenie tam przemówienia.

To, że zdanie jest wynikiem zastosowania pewnego schematu wnioskowania do innych zdań-przesłanek nazywa się *konsekwencją dowodową* albo *konsekwencją składniową*. W rozpatrywanym przykładzie wyraża się to zapisem

$$\{p \Rightarrow q, q \Rightarrow r\} \vdash p \Rightarrow r$$

Ogólnie, jeżeli $\{p_1, ..., p_n\}$ jest pewnym zbiorem zdań-przesłanek, a q jest zdaniem, które wyprowadzono z tego zbioru na podstawie jedno- lub wielokrotnego stosowania pewnych schematów wnioskowania, to zapisuje się to w postaci

$$\{p_1, ..., p_n\} \vdash q$$

Symbol $\vdash$ nazywa się symbolem *konsekwencji składniowej*. Powyższy zapis czyta się: q jest konsekwencją składniową zbioru zdań $\{p_1, ..., p_n\}$.

Rozpatrzmy teraz rozumowanie, które nie opiera się na przedstawionych schematach wnioskowania. Niech dany będzie przykład:

***Jeżeli** znany pianista da recital, **to** przyjdą tłumy, **gdy** ceny biletów nie będą zbyt wygórowane.*
***Jeżeli** znany pianista da recital, **to** ceny biletów nie będą zbyt wygórowane.*
Zatem: ***Jeżeli** znany pianista da recital, **to** przyjdą tłumy.*

W pierwszym z powyższych zdań występuje zwrot ***gdy***. Zgodnie z potocznym rozumieniem, zdanie złożone postaci p ***gdy*** q jest równoważne zdaniu ***jeżeli*** q, ***to*** p.

Czy jeżeli przesłanki w przykładzie – dwa pierwsze zadania – są prawdziwe, to czy prawdziwa jest również konkluzja – trzecie zdanie? Wprowadźmy oznaczenia. Niech:

p oznacza: *Znany pianista da recital.*
q oznacza: *Przyjdą tłumy.*
r oznacza: *Ceny biletów będą zbyt wygórowane.*

Po zastosowaniu tych oznaczeń nasze wnioskowanie ma postać:

$p \Rightarrow (\neg r \Rightarrow q)$
$p \Rightarrow \neg r$
zatem: $p \Rightarrow q$

Można przytoczyć dwa sposoby uzasadnienia poprawności wnioskowania. Pierwszy sposób można zilustrować tablicą prawdziwościową (tablica 1.2). Podobnie jak w poprzedniej tablicy, zamiast *prawda* i *fałsz* pisze się **P** i **F**.

Tablica 1.2

	p	q	r	$\neg r$	$\neg r \Rightarrow q$	$p \Rightarrow \neg r$	$p \Rightarrow (\neg r \Rightarrow q))$	$p \Rightarrow q$
1	F	F	F	P	F	P	P	P
2	F	F	P	F	P	P	P	P
3	F	P	F	P	P	P	P	P
4	F	P	P	F	P	P	P	P
5	P	F	F	P	F	P	F	F
6	P	F	P	F	P	F	P	F
7	P	P	F	P	P	P	P	P
8	P	P	P	F	P	F	P	P

Sposób uzasadniania jest tu następujący: wnioskowanie ma być niezawodne, to znaczy konkluzja ma być prawdziwa zawsze wtedy, gdy prawdziwe są przesłanki. Wystarczy rozpatrzyć wszystkie wartościowania zdań prostych p, q, r, przy których prawdziwe są zdania złożone stanowiące przesłanki, i sprawdzić, czy przy tych wartościowaniach prawdziwe jest również zdanie stanowiące konkluzję. Przypadki takich wartościowań reprezentują wiersze 1, 2, 3, 4 i 7 w tablicy 1.2. Analiza tych przypadków potwierdza poprawność wyprowadzonego wniosku.

Drugi sposób opiera się na następującym rozumowaniu nie wprost: jeżeli założyć, że nasz wniosek jest poprawny, to czy jest możliwe, aby jednocześnie były prawdziwe przesłanki i negacja konkluzji? Inaczej – czy zdanie

$$(p \Rightarrow (\neg r \Rightarrow q)) \wedge (p \Rightarrow \neg r) \wedge \neg (p \Rightarrow q)$$

może być prawdziwe dla dowolnych wartościowań zdań prostych p, q, r? Okazuje się, co pokazuje tablica 1.3, że przy wszystkich wartościowaniach zdanie to jest fałszywe. Nie może być tak, że jednocześnie są prawdziwe przesłanki i negacja konkluzji. Nie jest zatem możliwe, aby zdania stanowiące przesłanki mogły być niezgodne ze zdaniem stanowiącym konkluzję.

Oba sposoby nie polegały na tekstowym przekształcaniu przesłanek, ale opierały się na analizie znaczenia zdań, dokładniej – na analizie ich prawdziwości. Oba sposoby potwierdziły, że konkluzja jest *konsekwencją logiczną*, albo *konsekwencją semantyczną*, zbioru przesłanek. Fakt ten, w odniesieniu do przykładu, zapisuje się w postaci

$$\{p \Rightarrow (\neg r \Rightarrow q), p \Rightarrow \neg r\} \models p \Rightarrow q$$

Ogólnie, jeżeli $\{p_1, ..., p_n\}$ jest pewnym zbiorem zdań-przesłanek, a q jest jego logiczną konsekwencją, zapisuje się to w postaci

$$\{p_1, ..., p_n\} \models q$$

Symbol $\models$ nazywa się symbolem *konsekwencji semantycznej*. Zapis ten czyta się: q jest konsekwencją semantyczną zbioru zdań $\{p_1, ..., p_n\}$.

Tablica 1.3

	p	q	r	$\neg r$	$\neg r \Rightarrow q$	$p \Rightarrow (\neg r \Rightarrow q))$	$p \Rightarrow \neg r$	$p \Rightarrow q$	$\neg(p \Rightarrow q)$	$(p \Rightarrow (\neg r \Rightarrow q)) \wedge (p \Rightarrow \neg r) \wedge \neg(p \Rightarrow q)$
1	F	F	F	P	F	P	P	P	F	F
2	F	F	P	F	P	P	P	P	F	F
3	F	P	F	P	P	P	P	P	F	F
4	F	P	P	F	P	P	P	P	F	F
5	P	F	F	P	F	F	P	F	P	F
6	P	F	P	F	P	P	F	F	P	F
7	P	P	F	P	P	P	P	P	F	F
8	P	P	P	F	P	P	F	P	F	F

Mając pojęcia konsekwencji składniowej i konsekwencji semantycznej, można sprecyzować niezawodność schematów wnioskowania. Schemat wnioskowania jest niezawodny (albo poprawny), gdy dla dowolnego zbioru zdań $\{p_1, ..., p_n\}$ oraz zdania q, jeżeli

$$\{p_1, ..., p_n\} \vdash q$$

to

$$\{p_1, ..., p_n\} \models q$$

Inaczej: schemat wnioskowania jest niezawodny, gdy dla dowolnego zbioru przesłanek konsekwencja składniowa pociąga konsekwencję semantyczną.

Schemat wnioskowania, który nie jest niezawodny, jest praktycznie bezużyteczny. Wszystkie przedstawiane wcześniej schematy są schematami niezawodnymi (poprawnymi), zawodnym natomiast schematem wnioskowania jest na przykład schemat postaci

$$\frac{p \Rightarrow q}{q}$$

Aby to stwierdzić, wystarczy rozpatrzyć wartościowanie, w którym obie wypowiedzi p oraz q są fałszywe. Przy takim wartościowaniu przesłanka schematu $p \Rightarrow q$ jest prawdziwa, ale wniosek q jest fałszywy.

1.4. Indukcja matematyczna

Zasada indukcji matematycznej jest jednym z bardziej użytecznych schematów wnioskowania. Bezpośrednio odnosi się ona do badania własności wyrażanych w terminach liczb naturalnych, czyli do własności postaci $P(n)$, gdzie $n \in Nat$. Zasada indukcji opiera się na prostej obserwacji, że cały zbiór liczb naturalnych można uporządkować, zaczynając od 0, a następnie można przechodzić do kolejnych liczb przez dodawanie 1. Z obserwacji tej wynika, że udowodnienie, iż pewna własność $P(n)$ zachodzi dla każdej liczby naturalnej n wymaga pokazania, że zachodzi ona dla $n = 0$, oraz że zachodzi $P(n + 1)$, gdy zachodzi $P(n)$. Czy zachodzi $P(0)$ wymaga zatem bezpośredniego zbadania, natomiast $P(1)$ zachodzi, ponieważ zachodzi $P(0)$, podobnie $P(2)$ zachodzi, ponieważ zachodzi $P(1)$ itd.

Twierdzenie 1.1 (*Zasada indukcji matematycznej*)

Niech $P(n)$ będzie pewną własnością, która odnosi się do liczby naturalnej n. Aby pokazać, że własność $P(n)$ zachodzi dla każdej liczby naturalnej $n \in Nat$, wystarczy pokazać, że:

krok początkowy: P zachodzi dla $n = 0$, czyli zachodzi $P(0)$,
krok indukcyjny: jeżeli zachodzi $P(n)$, to również zachodzi $P(n + 1)$.

Dowodzenie zgodnie z zasadą indukcji składa się z dwóch kroków. Krok początkowy wymaga zbadania zachodzenia własności P dla $n = 0$. Drugi krok wymaga udowodnienia implikacji: jeżeli $P(n)$, to $P(n + 1)$. Założenie $P(n)$ w tej implikacji nazywa się *hipotezą indukcyjną*.

Przykład 1.1

Niech $P(n)$ oznacza własność, że $2^n > n^2$. Własność ta nie zachodzi dla wszystkich liczb naturalnych, ale zachodzi dla $n \geq 5$.
Łatwo sprawdzić, że zachodzi $P(5)$, gdyż $2^5 > 5^2$, ale nie zachodzi $P(4)$, gdyż nieprawda, że $2^4 > 4^2$, i podobnie nie zachodzi $P(0)$, $P(1)$, $P(2)$, $P(3)$.
Jeżeli zachodzi $P(n)$ oraz $n \geq 5$, to zachodzi również $P(n + 1)$.
Istotnie, jeżeli $2^n > n^2$, to – że $2^{n+1} > (n+1)^2$ – wynika z następującego wnioskowania:

$$
\begin{aligned}
2^{n+1} &= 2 \times 2^n \\
&> 2n^2 && \text{– na mocy hipotezy indukcyjnej, że } 2^n > n^2 \\
&= n^2 + n^2 \\
&\geq n^2 + 5n && \text{– na mocy założenia, że } n \geq 5 \\
&= n^2 + 2n + 3n \\
&> n^2 + 2n + 1 && \text{– własność trywialna: } 3n > 1 \text{ dla } n \geq 1 \\
&= (n + 1)^2
\end{aligned}
$$

Czasem własności, o których się sądzi, że można je łatwo udowodnić metodą indukcji, mogą się okazać niemożliwe do bezpośredniego wykorzystania zasady indukcji. Ilustruje to przykład.

Przykład 1.2

Należy pokazać, że suma n początkowych liczb nieparzystych jest kwadratem pewnej liczby naturalnej, to znaczy że dla każdej liczby naturalnej n istnieje taka liczba naturalna k, że

$$\sum_{i=0}^{n-1}(2i+1) = k^2$$

Dla $n = 0$ własność $P(0)$ zachodzi trywialnie dla $k = 1$. Załóżmy teraz, że istnieje $k > 1$ takie, że zachodzi powyższy wzór, wówczas

$$\sum_{i=0}^{n}(2i+1) = \sum_{i=0}^{n-1}(2i+1) + (2n+1) = k^2 + 2n + 1$$

Niestety, nie ma gwarancji, że wyrażenie $k^2 + 2n + 1$ jest kwadratem pewnej liczby naturalnej i tym samym nie można dowodu kontynuować. Może to sugerować, że indukcja jest zbyt słabym schematem dla dowodzenia tego typu własności. Tak jednak nie jest. Należy zauważyć, że gdyby w rozważanym wyrażeniu przyjąć, że $k = n$, wówczas zachodziłaby równość

$$k^2 + 2n + 1 = n^2 + 2n + 1 = (n + 1)^2$$

Obserwacja ta sugeruje rozważenie mocniejszej własności, mianowicie

$$\sum_{i=0}^{n-1}(2i+1) = n^2$$

Jej udowodnienie metodą indukcji jest już proste i pozostawia się je Czytelnikowi.

Udowodnienie pewnych własności wymaga silniejszej formy indukcji. Mianowicie, w indukcyjnym kroku, aby udowodnić $P(n + 1)$ wymaga się założenia, że nie tylko zachodzi $P(n)$, ale również, że zachodzą $P(1)$, ..., $P(n - 1)$.

Twierdzenie 1.2 (*Zasada indukcji matematycznej – silna forma*)

Niech $P(n)$ będzie pewną własnością, która odnosi się do liczby naturalnej n. Aby pokazać, że własność $P(n)$ zachodzi dla każdej liczby naturalnej $n \in Nat$, wystarczy pokazać, że:

krok początkowy: P zachodzi dla $n = 0$, czyli zachodzi $P(0)$, oraz
krok indukcyjny: zachodzi $P(n + 1)$, gdy zachodzi $P(k)$ dla każdego $k = 0, ..., n$.

Przykład 1.3

Należy pokazać, że każda liczba naturalna $n \geq 2$ jest iloczynem liczb pierwszych. Własność ta zachodzi oczywiście dla $n = 2$. Dalej załóżmy, że własność ta zachodzi dla 2, 3, ..., n. Na tej podstawie należy pokazać, że zachodzi ona również dla $n + 1$. Jeżeli $n + 1$ jest liczbą pierwszą, to własność jest oczywiście prawdziwa. W przeciwnym razie, gdy $n + 1$ nie jest liczbą pierwszą, oznacza to, że $n + 1$ może być wyrażone jako iloczyn km dwóch liczb, gdzie $2 \leq k \leq n$ oraz $2 \leq m \leq n$. Na mocy hipotezy indukcyjnej liczby k oraz m są iloczynami liczb pierwszych, zatem $n + 1$ wyraża się również jako iloczyn liczb pierwszych.

W przykładzie nie korzysta się bezpośrednio z hipotezy indukcyjnej dla n, ale dla pewnych liczb k, m mniejszych od $n + 1$. Ogólnie może zachodzić potrzeba wykorzystania wielu takich liczb.

1.5. Logika w informatyce

Logika klasyczna znajduje w informatyce szerokie zastosowanie. W pierwszej kolejności dostarcza ona języka do przedstawiania i badania własności modeli informatycznych, w tym systemów komputerowych i języków programowania. Szczególną rolę odegrała logika w formowaniu pojęcia algorytmu i obliczalności [Arbib 1968], [Davis, Hersh 1994], [Mostowski, Pawlak 1970], [Penrose 1996].

Oprócz logiki klasycznej są wykorzystywane logiki specjalne, przeznaczone na przykład do specyfikacji oprogramowania, a także jako języki programowania.

Ważną rolę w informatyce odgrywają różnego rodzaju logiki nieklasyczne, między innymi w systemach eksperckich (doradczych). Zadaniem takich systemów jest wspomaganie człowieka podczas podejmowania decyzji, na przykład postawienie przez lekarza diagnozy o stanie zdrowia pacjenta na podstawie wyników badań. Proces podejmowania decyzji opiera się w takich przypadkach na informacji niepewnej lub niepełnej, a wnioski z przeprowadzonego wnioskowania nie muszą być niezawodne.

Oto wybrane działy informatyki, w których logika znajduje bezpośrednie zastosowanie:

- Specyfikacja i weryfikacja programów – np. [Bicaregui, Fitzgerald, Lindsay 1994], [Dembiński, Małuszyński 1981], [Shepard 1995].

  Formuły logiczne służą do wyrażenia tego, co program ma obliczać, czyli do wyrażania specyfikacji programu. Stwierdzenie czy dany program oblicza to, co powinien, czyli czy spełnia zadaną specyfikację, polega na odpowiednim manipulowaniu na tekście formuł specyfikacji i na tekście programu. Inaczej: stwierdzanie poprawności programu względem danej specyfikacji polega na przeprowadzeniu dowodu w odpowiedniej logice programów.

- Przetwarzanie rozproszone, współbieżność i sterowanie, systemy komunikacji w czasie rzeczywistym – np. [Apt, Olderog 1991], [Huzar 1989].

  Odpowiednie formy logiki zostały opracowane w celu wyrażania i wnioskowania o zjawiskach, które występują w przestrzeni i trwają w czasie. Są one wykorzystywane na przykład do wnioskowania o współpracy pomiędzy mobilnymi równoległymi procesami.

- Zarządzanie bazami wiedzy – np. [Bolc, Borodziewicz, Wójcik 1991], [Tyugu 1989].

  Zadaniem odpowiednich logik jest umożliwienie udzielenia odpowiedzi na pytania kierowane do bazy wiedzy. Udzielanie odpowiedzi na pytania sprowadza się do zbadania, czy jest ono konsekwencją semantyczną nagromadzonej wiedzy.

- Projektowanie układów logicznych – np. [Harrison 1973].

  Projektowanie układów elektronicznych komputerów, na przykład układów scalonych, spowodowało powstanie specjalistycznych logik, między innymi logik wielowartościowych i progowych.

- Systemy ekspertowe, planowanie i sztuczna inteligencja – np. [Bolc, Borodziewicz, Wójcik 1991], [Banerji 1990], [Bubnicki 1990], [Huzar, Kurzyński, Sas 1994].

  Dział ten wykształcił nową grupę logik, których istotą jest prowadzenie wnioskowania w warunkach informacji niepełnej lub niepewnej.

- Przetwarzanie języka naturalnego (lingwistyka informatyczna) – np. [Carnap 1990], [Marciszewski 1987].

  W celu automatycznej analizy tekstu, czy też automatycznego przekładu z jednego języka na inny, powstały różne logiki służące przede wszystkim do wyrażania znaczenia tekstu.

- Programowanie logiczne – np. [Kowalski 1989], [Wójcik 1991].

  Język logiki może być traktowany bezpośrednio jako język programowania. To, co w podejściu klasycznym jest logiczną specyfikacją programu – przy zachowaniu pewnych ograniczeń – może być interpretowane jako wykonywalny program.

Ćwiczenia

1. Która z wypowiedzi jest zdaniem, a która funkcją zdaniową:
   a) *Księżyc jest zrobiony z żółtego sera.*
   b) *On faktycznie jest wysokim mężczyzną.*
   c) *Słońce krąży dookoła Ziemi.*
   d) *W ciągu wieków składniki te uformowały rafy.*
   e) *Niech żyje przyjaźń między narodami!*

f) *Dwa jest liczbą parzystą.*
g) *Która drużyna zdobędzie mistrzostwo kraju w piłce nożnej?*
h) *Oczekuje się, że w przyszłym roku obroty na giełdzie znacznie wzrosną.*
i) $x^2 - 4 = 0$.
j) *Długi honorowe należy spłacać w ciągu 24 godzin.*
k) *Mężczyzna jest wyższy od kobiety.*
l) *Należy raczej zapobiegać niż leczyć.*
m) *Kalifornię co roku nawiedza trzęsienie ziemi o sile 7 stopni w skali Richtera.*
n) *Król Jagiełło był raczej wysokim mężczyzną.*

2. Jaką wartość logiczną mają zdania:

a) *8 jest liczbą nieparzystą lub 6 jest liczbą parzystą.*
b) *8 jest liczbą nieparzystą oraz 6 jest liczbą parzystą.*
c) *Jeżeli 8 jest liczbą nieparzystą, to 6 jest liczbą parzystą.*
d) *Jeżeli 8 jest liczbą nieparzystą oraz 6 jest liczbą parzystą, to 6 jest większe od 8.*

3. Które ze zdań jest negacją danego zdania:

a) *Wynikiem obliczeń jest albo 2, albo 3.*
(i) *Wynikiem nie jest ani 2, ani 3.*
(ii) *Wynikiem nie jest 2 lub nie jest 3.*
(iii) *Wynikiem nie jest 2 i nie jest 3.*

b) *Ogórek jest zieloną rośliną nasienną.*
(i) *Ogórek nie jest zielony, ale jest rośliną nasienną.*
(ii) *Ogórek nie jest zielony lub nie jest rośliną nasienną.*
(iii) *Ogórek nie jest zielony i nie jest rośliną nasienną.*

4. Wskaż poprzednik i następnik implikacji w zdaniach:

a) *Pomyślny wzrost roślin jest uwarunkowany prawidłowym nawadnianiem.*
b) *W przypadku modyfikacji programu pojawią się w nim błędy.*
c) *Błędy w programie pojawią się tylko w przypadku jego modyfikacji.*
d) *Oszczędność energii jest związana z dobrą izolacją ścian i szczelnością okien.*

5. W podanych zdaniach złożonych rozpoznaj zdania proste i łączące je spójniki:

a) *Edmund Hillary i Tenzing Norgay są pierwszymi zdobywcami Mont Everestu.*
b) *Indochiny leżą w strefie tropikalnej i mają gorące lata, ale zimy w części północnej są chłodne.*
c) *Niezależnie od tego, jak wysoko skaczesz, księżyca nie osiągniesz, chyba że polecisz tam rakietą.*

6. Przedstaw tablice prawdziwościowe wyrażające znaczenie występujących w języku polskim, następujących zwrotów:

a) *co najwyżej jedno z dwojga,*
b) *dokładnie jedno z dwojga,*

c) *o ile ...,*
d) *ani ..., ani,*
e) *pod warunkiem, że ...,*
f) *chyba że,*
g) *choćby nawet,*
h) *zawsze wtedy, gdy.*

7. Za pomocą dowolnych dwóch spośród spójników: negacji, koniunkcji, alternatywy i implikacji, zdefiniuj znaczenie wyrażeń podanych w zadaniu 6.

8. Rozpatrz następujące wnioskowanie oparte na sylogizmie warunkowym:

Jeżeli dzisiaj jest wtorek, to jutro jest środa.
Jeżeli dzisiaj jest środa, to jutro jest czwartek.
Zatem: *Jeżeli dzisiaj jest wtorek, to jutro jest czwartek.*

Wyjaśnić przyczyny paradoksalnego wniosku.

9. Oto fragment raportu policji sporządzonego przez młodego aspiranta:

Świadek nie był zastraszony lub też, jeśli Henry popełnił samobójstwo, to testament odnaleziono. Jeśli świadek był zastraszony, to Henry nie popełnił samobójstwa. Jeśli testament odnaleziono, to Henry popełnił samobójstwo. Jeśli Henry nie popełnił samobójstwa, to testament odnaleziono.

Co komendant policji może wywnioskować z powyższego raportu (poza oczywistym faktem, że należy zwolnić aspiranta)? Odpowiedz na pytania:

Czy świadek był zastraszony?
Czy Henry popełnił samobójstwo?
Czy testament odnaleziono?

10. Pośród członków pewnego klubu lingwistycznego każdy uczy się francuskiego, niemieckiego lub hiszpańskiego. Wiadomo, że 20 uczy się francuskiego, 12 francuskiego i hiszpańskiego, 16 niemieckiego, 16 hiszpańskiego, 4 francuskiego i niemieckiego, 7 niemieckiego i hiszpańskiego, 3 wszystkich trzech języków. Ilu członków liczy klub? Ilu z nich uczy się dokładnie dwóch języków?

11. Oto przykłady wnioskowań przez indukcję:

a) Pokażę, że wszystkie liczby naturalne są parzyste. Oczywiście 0 jest liczbą parzystą. Niech n będzie dowolną liczbą naturalną i załóżmy, że dla wszystkich $k < n$, k jest parzyste. Niech n_1 i n_2 będzie dowolnym rozbiciem liczby n na sumę liczb mniejszych (tzn. $n = n_1 + n_2$). Ponieważ n_1 oraz n_2 są mniejsze od n, zatem n_1 i n_2 są parzyste, a więc n jest parzyste jako suma dwóch liczb parzystych.

b) Pokażę, że wszystkie dodatnie liczby naturalne są nieparzyste. Oczywiście 1 jest liczbą nieparzystą. Niech n będzie dowolną liczbą naturalną i załóżmy, że dla wszystkich $k < n$, k jest nieparzyste. Niech 1, n_1 *i* n_2 będzie dowolnym rozbiciem liczby n na sumę trzech liczb mniejszych (tzn. $n = n_1 + n_2 + 1$). Ponieważ

n_1 oraz n_2 są mniejsze od n, zatem n_1 i n_2 są nieparzyste, a więc n jest parzyste jako suma dwóch liczb nieparzystych i liczby 1.

c) Pokażę, że wszystkie proste na płaszczyźnie są równoległe. Rozważmy jednoelementowy zbiór prostych na płaszczyźnie. Oczywiście wszystkie proste należące do tego zbioru są do siebie równoległe. Załóżmy, że w każdym n-elementowym zbiorze prostych wszystkie proste są do siebie równoległe. Rozważmy teraz $(n + 1)$-elementowy zbiór prostych. Ustalmy w nim jedną prostą p. Na mocy założenia indukcyjnego wszystkie pozostałe n prostych są do siebie równoległe. Ustalmy teraz inną prostą q. Na mocy założenia indukcyjnego wszystkie pozostałe n prostych są również do siebie równoległe. Ponieważ relacja równoległości prostych jest przechodnia, wszystkie $n + 1$ proste są równoległe. Na mocy zasady indukcji matematycznej każdy zbiór prostych na płaszczyźnie zawiera wyłącznie proste równoległe. Dotyczy to zbioru wszystkich prostych na płaszczyźnie.

Które z tych rozumowań jest poprawne? Wskaż błędy popełnione w błędnym rozumowaniu.

12. Rozważyć uogólnienie problemu przedstawionego w przykładzie 1. Dla jakich liczb naturalnych n, k zachodzi nierówność: $2^n > n^k$?

13. O własności $P(n)$ wiadomo, że jest prawdziwe $P(1)$, a ponadto dla każdej liczby naturalnej n zachodzi implikacja $P(n) \Rightarrow P(n + 10)$. Czy wynika stąd, że prawdziwe są implikacje:
    a) $P(32) \Rightarrow P(62)$,
    b) $P(33) \Rightarrow P(61)$,
    c) $P(34) \Rightarrow P(63)$,
    d) $P(31) \Rightarrow P(64)$.

14. O własności $P(n)$ wiadomo, że jest prawdziwe $P(1)$, natomiast $P(100)$ jest fałszywe, a ponadto dla każdej liczby naturalnej n zachodzi implikacja $P(n) \Rightarrow P(n + 2)$. Czy wynika stąd, że:
    a) $P(101)$ jest prawdziwe,
    b) $P(300)$ jest prawdziwe,
    c) $P(50)$ jest fałszywe,
    d) $P(200)$ jest fałszywe.

15. Przeanalizuj prawdziwość zdania: *To, co mówię w tej chwili, jest kłamstwem.*

16. Oto rozmowa czterech krasnoludków A, B, C oraz D, z których każdy zawsze mówi prawdę albo zawsze kłamie:
    A mówi do B: *Jesteś kłamcą.*
    C mówi do A: *Ty sam jesteś kłamcą.*
    D mówi do C: *Oni obaj są kłamcami. I ty także jesteś kłamcą.*

    Który z nich mówi prawdę?

2. Elementarne pojęcia mnogościowe

2.1. Zbiór i element zbioru

Podstawą wszelkiej komunikacji pomiędzy ludźmi, a także pomiędzy ludźmi i komputerami, jest język, który używa wspólnie ustalonych symboli i jednoznacznie skojarzonych z nimi pojęć. Symbole służą do reprezentacji pojęć. Mogą one mieć różną postać – mogą to być dźwięki, obrazy, znaki graficzne. Każdy symbol powinien reprezentować pojęcie, które jest jednakowo rozumiane przez obiekty (podmioty) uczestniczące w komunikacji. Wprowadzenie języka wymaga zdefiniowania odpowiedniego zestawu symboli oraz zdefiniowania przypisywanego im znaczenia. Z wprowadzaniem nowego języka wiąże się pewien problem: definicje elementów języka wymagają opisu, wyrażanego w pewnym innym języku. Oznacza to, że przed wprowadzeniem pewnego języka należy dysponować innym językiem służącym do opisu nowego języka. W celu odróżnienia tych języków, język definiowany nazywa się *językiem przedmiotowym*, krótko językiem, a język służący do opisu języka przedmiotowego nazywa się *metajęzykiem*.

Uwaga

Pojęcie metajęzyka funkcjonuje w wielu sytuacjach. Na przykład podczas nauki języka obcego, załóżmy angielskiego, język ten opisujemy i wyjaśniamy za pomocą języka polskiego. Każdy metajęzyk ma swój metajęzyk – można pisać po niemiecku o kimś, kto pisze po polsku o angielskim. Istnieje niekończąca się hierarchia metajęzyków.

Elementy języka formalnego (symbolicznego) zawierają pojęcia odnoszące się do dwóch obszarów – obszaru teorii mnogości i logiki. Elementy tego języka wyjaśnia się w języku naturalnym. Język naturalny odgrywa tu rolę metajęzyka. Z języka naturalnego wykorzystuje się oczywiście tylko te pojęcia, co do których nie ma wątpliwości interpretacyjnych. Sytuacja jest podobna do tej, z którą spotyka się, studiując na przykład encyklopedię. Encyklopedia służy do wyjaśniania pewnych pojęć-haseł i czyni to za pomocą innych pojęć-haseł, o których się zakłada, że powinny być powszechnie znane i jednoznacznie rozumiane. Czasem wprawdzie jest tak, że w wyjaśnianiu pewnych haseł występują inne hasła, ale w odniesieniu do encyklopedii jako całości przyjmuje się, że istnieje pewien zestaw pojęć pierwotnych, których encyklo-

pedia używa, ale ich nie wyjaśnia i które się uważa za powszechnie zrozumiałe. Postępowanie takie nie jest całkowicie ścisłe i czasem może być źródłem niejednoznaczności lub nawet sprzeczności, ale praktycznie – w większości przypadków – pozwala na wprowadzanie i wyjaśnianie potrzebnych pojęć.

Obszary teorii mnogości i logiki silnie się przenikają. Formułując pojęcia należące do obszaru teorii mnogości, korzysta się z pojęć logicznych, ale też i odwrotnie – formułowanie własności logicznych wymaga odwołania się do pojęć mnogościowych. Można się zatem spotkać z dwoma podejściami do opisu logiki klasycznej. Pierwsze podejście polega na przyjęciu pewnych elementów teorii mnogości i wyprowadzaniu na ich podstawie pojęć logicznych. Drugie podejście, odwrotnie, polega na przyjęciu podstawowych pojęć logicznych i na wyprowadzaniu na ich podstawie pojęć mnogościowych. W książce przyjęto pierwsze podejście – przed pełnym opisem pojęć logicznych wyjaśnia się elementarne pojęcia z zakresu teorii mnogości.

Do podstawowych pojęć mnogościowych zalicza się:

- pojęcie *zbioru*,
- pojęcie *elementu* zbioru,
- pojęcie *należenia* bądź *nienależenia* elementu do zbioru.

Pojęcie zbioru – intuicyjnie zrozumiałe – okazało się bardzo trudne do precyzyjnego zdefiniowania. Poprzestaniemy tu na intuicyjnym albo naiwnym rozumieniu zbioru, tak jak czynił to w XIX wieku Cantor[3] – twórca teorii mnogości – który zbiór określał jako

> *ujęcie w całość określonych, dobrze wyróżnionych obiektów, zwanych elementami zbioru.*

W określeniu tym nie wskazuje się, czym mogą być elementy zbioru. Podane określenie zbioru nie jest precyzyjne, gdyż przy próbie odpowiedzi na pewne pytania mogą się pojawić sprzeczności. Próby uściślenia pojęcia zbioru prowadziły do powstania sformalizowanej teorii mnogości.

Należy podkreślić, że w podanym określeniu kładzie się akcent na rozróżnialność bytów stanowiących elementy zbioru. Nie określa się natomiast jak osiągać tę rozróżnialność, czy na przykład przez jednoznaczną identyfikację bytów, czy przez określenie unikalnych ich własności. Oznacza to jednak, że mając dwa byty, potrafi się stwierdzić, czy są one identyczne czy różne.

Jeżeli symbolem A oznacza się pewien zbiór oraz symbolem a oznacza się pewien element, to zapis $a \in A$ czyta się: *a jest elementem zbioru A*, natomiast zapis $a \notin A$ czyta się: *a nie jest elementem zbioru A*.

Jeżeli $a \in A$ oraz $b \in A$, to zapisuje się to skrótowo: $a, b \in A$.

[3] Georg Cantor (1845–1918).

Na ogół zbiory będziemy oznaczać napisami zaczynającymi się wielkimi literami lub pojedynczymi literami greckimi, a elementy zbiorów odpowiednimi małymi literami, z ewentualnymi indeksami.

Pewne zbiory przyjmuje się jako znane. Będą to zbiory: liczb naturalnych *Nat*, liczb całkowitych *Całkowite*, liczb wymiernych – *Wymierne*, liczb rzeczywistych – *Rzeczywiste*.

Szczególnym zbiorem jest *zbiór pusty* – zbiór, który nie ma żadnego elementu. Będzie on oznaczany symbolem $\emptyset$.

W celu wyeliminowania pewnej klasy paradoksów (zob. dalej – paradoks Russella), które mogą powstać podczas definiowania zbiorów, zakłada się, że żaden zbiór nie może być swoim elementem, to znaczy dla dowolnego zbioru A zachodzi: $A \notin A$.

2.2. Definiowanie zbiorów

Zbiory można definiować w różny sposób. Przedstawia się trzy sposoby definiowania zbiorów:

- enumeracyjny,
- rekursywny,
- ekstensjonalny.

Najprostszym sposobem definiowania zbioru jest jawne wskazanie wszystkich jego elementów. Sposób ten nazywa się *enumeracją* lub *wyliczeniem* elementów zbioru. Schemat takiej definicji ma postać

$$A =_{\text{def}} \{a_1, a_2, ..., a_n\}$$

Zapis ten czytamy: *A jest nazwą zbioru, którego elementami są* $a_1, a_2, ..., a_n$. Symbol $=_{\text{def}}$ czytamy: *równy z definicji*.

Przedstawiony wyżej schemat definicji zbioru zawiera dwa elementy:

- wprowadza symbol A jako *nazwę* zbioru,
- określa *znaczenie* (inaczej *interpretację*), które przypisujemy temu symbolowi; jest nim zestaw elementów $a_1, a_2, ..., a_n$, które należą do zbioru A. Elementy te, oddzielone przecinkami, tworzą skończony ciąg. Występujące tu trzy kropki są tylko zaznaczeniem, że liczba tych elementów może być dowolna, ale skończona. Definicja konkretnego zbioru musi oczywiście wymienić jawnie wszystkie jego elementy.

W związku z rozróżnieniem pojęcia symbolu oraz pojęcia znaczenia symbolu należy zwrócić uwagę na nazwę zbioru. Możliwe są dwa spojrzenia na nazwę.

W pierwszym spojrzeniu nazwę traktuje się tylko jako symbol – nazwa nie wyraża żadnego znaczenia, jest tylko znakiem lub ciągiem znaków z ustalonego repertuaru

znaków. Taką rolę ma symbol A występujący po lewej stronie w podanej poprzednio definicji.

W drugim spojrzeniu nazwę traktuje się jako zbiór, którego elementy są wymienione w nawiasach. Aby na przykład odpowiedzieć na pytanie czy $a \in A$, należy widzieć A jako zestaw konkretnych elementów.

Często dalej używanym zbiorem będzie zbiór wartości logicznych

$Logiczne =_{def} \{prawda, fałsz\}$

Rozpatrzmy dalsze przykłady enumeracyjnej definicji zbiorów:

$Kreski =_{def} \{|, -\}$
$Strzałki =_{def} \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$
$DniTygodnia =_{def} \{poniedziałek, wtorek, środa, czwartek, piątek, sobota, niedziela\}$
$LiteryMałe =_{def} \{a, b, ..., z\}$
$LiteryDuże =_{def} (A, B, ..., Z\}$

Elementami pierwszego i drugiego zbioru są symbole graficzne, elementami pozostałych zbiorów są litery lub napisy. W definicjach dwóch ostatnich zbiorów występują takie same kropki, ale z uwagi na kontekst, w którym występują, potrafimy nadać im odpowiednie różne znaczenia.

Bezpośrednio z definicji zbiorów wynika, że na przykład:

$| \in Kreski$
$\rightarrow, \uparrow \in Strzałki$

Używane pojęcie zbioru nie narzuca ograniczeń na to, czym mogą być jego elementy. W szczególności elementami zbioru mogą być inne zbiory. Rozpatrzmy przykłady zbiorów:

$\{a\}$
$\{\{a\}\}$
$\{\{a, b\}, \{a\}\}$
$\{\{\{a\}\}, \{a\}, a\}$

Są to zbiory *anonimowe*, to znaczy niemające nazw. Pierwszy zbiór składa się tylko z jednego elementu a. Drugi zbiór składa się również z jednego elementu, ale elementem tym jest zbiór jednoelementowy $\{a\}$. Trzeci zbiór ma dwa elementy, którymi są zbiory $\{a, b\}$ oraz $\{a\}$. Ostatni zbiór ma trzy elementy, z których każdy ma różną strukturę – pierwszy jest zbiorem postaci $\{\{a\}\}$, drugi jest zbiorem postaci $\{a\}$, a trzeci jest pojedynczym elementem a.

Enumeracyjne definiowanie zbioru nie jest możliwe, gdy zbiór zawiera nieskończenie wiele elementów. W tym przypadku można stosować podejście *rekursywne*. Rekursywna definicja zbioru składa się z dwóch części:

- *części bazowej*, w której jawnie wskazuje się na pewne obiekty jako elementy definiowanego zbioru,
- *części rekursywnej*, w której wskazuje się na nowe obiekty jako elementy definiowanego zbioru, przez odpowiednie odwołanie się do tych obiektów, o których już wiadomo, że należą do definiowanego zbioru (inaczej: część rekursywna określa jak za pomocą obiektów, o których wcześniej wiemy, że są elementami zbioru, można wyrazić inne obiekty, które są również elementami definiowanego zbioru).

Szczególnie ważnym i potrzebnym zbiorem nieskończonym jest zbiór liczb naturalnych *Nat*. Można zdefiniować go rekursywnie następująco:

- $0 \in Nat$
- jeżeli $n \in Nat$, to $n + 1 \in Nat$.

Elementy zbioru w części rekursywnej definicji są wyrażane przez napisy n oraz $n + 1$. Napisy te reprezentują pewne liczby, przy czym to, jakie są to liczby, zależy od tego, jaką liczbę przypisze się symbolowi n. Stosując część rekursywną po raz pierwszy, symbolowi n przypisuje się 0 i na tej podstawie wnioskuje się, że 1 jest również elementem zbioru *Nat*. Stosując część rekursywną po raz drugi, symbolowi n przypisze się liczbę 1 itd.

W podanej definicji zakłada się, że wiadomo jest, czym jest liczba i co oznacza dodanie jedynki do liczby. Bez rozumienia tych pojęć nie można zrozumieć, czym jest zbiór *Nat*. Pojęcia te należą do metajęzyka, którego używamy do zdefiniowania zbioru liczb naturalnych. Inna, formalna definicja liczb naturalnych, która nie odwołuje się do pojęcia liczby i dodawania, jest podana dalej.

Uwaga

Podana definicja zbioru liczb naturalnych przyjmuje, że liczba 0 jest najmniejszą liczbą naturalną. Spotyka się też definicje, które przyjmują, że najmniejszą liczbą naturalną jest 1. Konwencja ta wynika z historii powstawania liczb naturalnych, kiedy – do odkrycia zera – za liczby naturalne uważano tylko 1, 2, 3 itd.

Podobnie można zdefiniować zbiór dodatnich liczb parzystych:

- $2 \in ParzysteDodatnie$
- jeżeli $n \in ParzysteDodatnie$, to $n + 2 \in ParzysteDodatnie$.

Ponownie należy zwrócić uwagę, że w części rekursywnej definicji zbioru użyto konstrukcji $n + 2$, która należy do metajęzyka służącego do definiowania zbioru, i o której zakładamy, że jest dla Czytelnika jednoznacznie zrozumiała.

Inny przykład rekursywnej definicji pewnego zbioru liczb *PewneLiczby* jest następujący:

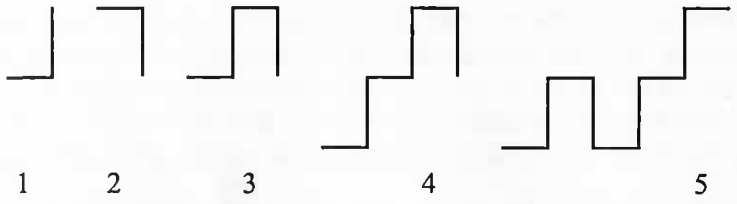
- $5, 7 \in PewneLiczby$
- jeżeli $n, m \in PewneLiczby$, to $n + m \in PewneLiczby$

Część bazowa określa, że elementami zbioru *PewneLiczby* są liczby 5 i 7. Analizując część rekursywną, łatwo się przekonać, że elementami tego zbioru będą także liczby 10, 12, 14, 15, 17, 20 itd.

Rekursywna definicja zbioru *LinieŁamane*, którego elementami są symbole graficzne – linie łamane, złożone z elementów zbioru *Kreski* – ma następującą postać:

- $|, - \in LinieŁamane$
- jeżeli $a, b \in LinieŁamane$, to linia powstająca z połączenia a oraz b w taki sposób, że jeden z końców a był połączony z jednym końcem b tak, aby poza miejscem połączenia a oraz b nie miały innych punktów wspólnych, należy również do zbioru *LinieŁamane*.

Łatwo się przekonać, że elementami zbioru *LinieŁamane* będą m.in. następujące linie:



Rys. 2.1. Elementy zbioru *LinieŁamane*

Linie o numerach 1 i 2 powstają przez różne powiązania elementów zbioru *Kreski*, linia 3 jest wynikiem połączenia linii 1 i 2, linia 4 – linii 1 i 3, a linia 5 – linii 3 i 4.

Rekursywna definicja zbioru ma charakter konstruktywny, to znaczy określa jak można skonstruować nowe elementy zbioru z innych elementów, o których już wiemy, że są elementami definiowanego zbioru. Inaczej można powiedzieć, że definicja rekursywna wyznacza pewien *algorytm* konstrukcji elementów zbioru. Algorytm jest w tym momencie rozumiany nieformalnie jako ciąg pewnych kroków obliczeniowych prowadzących do rozwiązania danego problemu. Z tego względu rekursywny sposób definiowania zbiorów jest bardzo często wykorzystywany w informatyce. Rekursywne podejście pozwala wprawdzie na definiowanie zbiorów nieskończonych, ale nie dowolnych zbiorów, lecz tylko zbiorów przeliczalnych, tzn. takich, których wszystkie elementy można zestawić w jeden ciąg (pojęcie przeliczalności zbioru jest zdefiniowane w dalszej części książki). Oczywiście, w skończonej liczbie kroków można wyznaczyć tylko skończoną liczbę elementów zbioru.

Najogólniejszy sposób definiowania zbiorów opiera się na podejściu *ekstensjonalnym*. Podejście to polega na definiowaniu zbioru przez określenie własności jego elementów. Schemat definicji zbioru ma postać

$$A =_{\text{def}} \{a \mid P(a)\}$$

Zapis ten czytamy: *do zbioru o nazwie A należą wszystkie te i tylko te elementy a, które mają własność P(a)*, czyli takie elementy, dla których wypowiedź $P(a)$ jest prawdziwa. $P(a)$ jest funkcją zdaniową, dlatego też ten sposób definiowania zbiorów nazywa się także *definiowaniem przez funkcję zdaniową*. Formalna postać funkcji zdaniowych będzie precyzyjnie określona w dalszej części książki.

Uwaga

Definicja ekstensjonalna nie określa skąd brać te elementy, które mają własność $P(a)$. Ogólne pytanie o to, które byty należy rozważać przy takim definiowaniu zbioru, wiąże się ze znanym problemem filozoficznym, określanym jako problem *powszechników* albo *uniwersaliów*.

Rozpatrzmy poprzedni przykład zbioru dodatnich liczb parzystych

$$ParzysteDodatnie =_{\text{def}} \{x \mid (x \textit{ jest liczbą naturalną}) \wedge (x > 0) \wedge (x \textit{ jest podzielne przez } 2)\}$$

Własność

$$(x \textit{ jest liczbą naturalną}) \wedge (x > 0) \wedge (x \textit{ jest podzielne przez } 2)$$

ma postać wypowiedzi złożonej. Poszczególne jej człony należą do metajęzyka – języka arytmetyki. Aby rozumieć sens całej wypowiedzi, należy rozumieć jej części składowe:

x *jest liczbą naturalną*, czyli $x \in Nat$

$x > 0$

x *jest liczbą naturalną podzielną przez* 2

oraz łączący je spójnik logiczny $\wedge$. Pierwsza z wypowiedzi wymaga rozumienia przynależności elementu do zbioru, a pozostałe wymagają elementarnej wiedzy z zakresu arytmetyki. Znaczenie spójnika $\wedge$ zostało wyjaśnione w poprzednim rozdziale.

Czasem, gdy definiujemy nowy zbiór A, wygodne jest odniesienie do innego, wcześniej ustalonego zbioru B. Piszemy wtedy

$$A =_{\text{def}} \{x \in B \mid P(x)\},$$

co jest skrótem od

$$\{x \mid x \in B \wedge P(x)\}.$$

Możemy więc napisać

$$ParzysteDodatnie =_{\text{def}} \{x \in Nat \mid (x>0) \wedge (x \textit{ jest podzielne przez } 2)\}$$

Uwaga

Czasem, definiując zbiór, zamiast symbolu $=_{\text{def}}$ używa się również innych oznaczeń, na przykład $\stackrel{\text{def}}{=}$, $\hat{=}$, a nawet $=$. Ostatnim symbolem należy się posługiwać

ostrożnie, gdyż jego znaczeniem podstawowym jest stwierdzanie równości (identyczności) elementów należących do pewnego zbioru.

Podejście ekstensjonalne do definiowania zbiorów jest wygodne i uniwersalne, ale nieostrożny sposób formułowania własności może prowadzić do absurdu. Znany przykład takiego absurdu jest nazywany *paradoksem Russella*[4]. Russel wykorzystał w skrajnej postaci rozumowanie, stosowane w początkowym okresie rozwoju teorii mnogości, mianowicie: niech Z będzie zbiorem zdefiniowanym następująco:

$$Z =_{\text{def}} \{X \mid X \notin X\}$$

to znaczy Z jest zbiorem – rodziną zbiorów – którego elementami są wszystkie zbiory X, mające tę własność, że nie są swoimi elementami. Odpowiedzmy teraz na pytanie: czy $Z \in Z$? Jeżeli Z jest swoim elementem, czyli $Z \in Z$, to oznacza, że ma taką samą własność jak wszystkie elementy zbioru Z, czyli $Z \notin Z$. Jeżeli natomiast Z nie jest swoim elementem, czyli $Z \notin Z$, to z definicji należy do rodziny zbiorów Z, czyli $Z \in Z$. W obu przypadkach zachodzi sprzeczność.

Paradoks ten uzasadnia dlaczego na początku rozdziału wprowadzono ograniczenie, że dla dowolnego zbioru A zachodzi $A \notin A$.

Warto zwrócić uwagę, że przypuszczenie, iż zbiór może być swoim elementem, wcale nie jest absurdalne. Rozważmy bowiem zbiór Z, którego elementami są zbiory nieskończone, to znaczy zbiory o nieskończenie wielu elementach. Z pewnością istnieje nieskończenie wiele zbiorów nieskończonych, a zatem zbiór Z jest nieskończony, czyli jest swoim elementem!

Rozpatrzmy jeszcze jeden przykład. W większości praktycznie spotykanych przypadków mamy do czynienia ze zbiorami, które nie są swoimi elementami – nazwijmy je zbiorami zwyczajnymi, w odróżnieniu od pozostałych zbiorów, które nazwiemy niezwyczajnymi. Utwórzmy teraz zbiór Z, którego elementami są wszystkie zbiory zwyczajne. Zapytajmy: czy zbiór Z jest zbiorem zwyczajnym czy niezwyczajnym? Jeśli Z jest zbiorem zwyczajnym, to wchodzi w skład swoich elementów, lecz wówczas – zgodnie z określeniem – jest on zbiorem niezwyczajnym. Jeśli natomiast Z jest zbiorem niezwyczajnym, to – zgodnie z określeniem niezwyczajności – powinien być swoim własnym elementem, a przecież elementami zbioru Z są tylko zbiory zwyczajne. Ponownie, jak w poprzednim przykładzie, w obu przypadkach zachodzi sprzeczność.

Uwaga

Potrzeba wyeliminowania sprzeczności wynikających ze zbyt swobodnego definiowania bardzo „obszernych", prowadzących do antynomii, zbiorów doprowadziła do aksjomatycznego ujęcia teorii zbiorów (zob. podrozdział 4.1). Niektóre z takich koncepcji wiązały się z wprowadzeniem pojęcia klasy. Zasadnicza idea

[4] Bertrand Russell (1872–1970).

polegała na odróżnieniu klasy od zbioru: zbiory mogą być elementami innych zbiorów, klasy zaś nie, dlatego – zamiast operować niejasnym pojęciem zbiór wszystkich zbiorów – mówi się o klasie wszystkich zbiorów.

Pojęcie klasy w wyżej przedstawionym znaczeniu należy odróżniać od, mającego zupełnie inne znaczenie, pojęcia klasy używanego w informatyce – w programowaniu i projektowaniu systemów informatycznych.

Przykład 2.1

Rozpatrzmy przykłady zbiorów używanych w językach programowania. W zasadzie wszystkie takie zbiory są zbiorami skończonymi. Wyróżnia się między innymi predefiniowane zbiory wartości związane z typami danych.

Zbiór wartości logicznych

$$Boolean =_{\text{def}} \{false, true\}$$

Zbiór całkowitoliczbowy

$$Integer =_{\text{def}} \{-N, ..., 0, ..., N\},$$

gdzie N jest liczbą naturalną określoną przez daną implementację języka.

Zbiór liczb rzeczywistych

$$Real =_{\text{def}} \{-N*\delta, ..., 0, ..., N*\delta\}$$

gdzie N jest liczbą naturalną, a δ jest liczbą wymierną określoną przez daną implementację języka; jest to tzw. *stałoprzecinkowa* reprezentacja liczb (w reprezentacji *zmiennoprzecinkowej* kolejne liczby są oddalone od siebie o zmienną różnicę). Warto podkreślić, że wbrew temu, co sugeruje nazwa, zbiór ten zawiera skończoną ilość liczb wymiernych.

Zbiór napisów

$$String =_{\text{def}} \{s \mid s \text{ jest skończonym ciągiem znaków ustalonego repertuaru znaków}\}$$

W praktycznej implementacji typu napisowego długość takich ciągów jest ograniczona konkretną liczbą. Przykładem takiego repertuaru znaków są na przykład znaki kodów stosowane w reprezentacji maszynowej, na przykład jednobajtowy kod ASCII (*American Standard Code for Information Interchange*) czy dwubajtowe kody *Unicode* lub BMP (*Basic Multilingual Plane*).

Definiowany przez programistę zbiór wyliczeniowy to na przykład

$$DniTygodnia =_{\text{def}} \{pon, wt, sr, czw, pt, sob, nd\}$$

gdzie *pon*, *wt*, ..., *nd* są pewnymi ustalonymi napisami. Napisy te – w odróżnieniu od napisów, które należą do zbioru *String* – są nierozkładalne, to znaczy ich fragmenty nie są elementami zbioru *DniTygodnia*.

Specyficznym dla wielu języków, nie tylko języków programowania, jest zbiór identyfikatorów. Zbiór ten będzie dalej często wykorzystywany i oznaczymy go symbolem *Ident*. Może on być definiowany na przykład tak

$$Ident =_{def} \{s \mid s \text{ jest niepustym ciągiem składającym się z liter lub cyfr, którego pierwszym elementem jest litera}\}$$

Należy zwrócić uwagę, że w treści własność definiujących zbiory występują pojęcia, o których się zakłada, że są pojęciami zrozumiałymi – są to pojęcia metajęzyka, w którym opisujemy dane własności. Na przykład w definicji zbiorów *String* oraz *Ident* takim pojęciem jest ciąg, a w definicji zbioru *DniTygodnia* takim pojęciem jest napis.

2.3. Podzbiory, równość zbiorów, zbiory potęgowe

Mówimy, że A jest *podzbiorem* zbioru B, co oznaczamy $A \subseteq B$, wtedy i tylko wtedy, gdy dla dowolnego elementu a: jeżeli $a \in A$, to także $a \in B$. Symbol $\subseteq$ nazywa się symbolem *zawierania* lub symbolem *inkluzji*. Podaną definicję zawierania zbiorów można również wyrazić formalnie

$$A \subseteq B \Leftrightarrow (\forall a \bullet a \in A \Rightarrow a \in B)$$

Z definicją wiąże się następujący komentarz: Jest to definicja w postaci *normalnej*. Składa się ona z dwóch części przedzielonych symbolem równoważności $\Leftrightarrow$, który czytamy: *wtedy i tylko wtedy*. Część po lewej stronie symbolu równoważności jest wyrażeniem zawierającym pojęcie definiowane – *definiendum*, a część po prawej stronie zawiera pojęcie definiujące – *definiens*. Poprawność definicji wymaga, aby po prawej stronie nie występowało pojęcie definiowane, gdyż byłby to przypadek „błędnego koła". Oczywiście, aby rozumieć sens definicji, pojęcia występujące w części definiującej muszą być znane. Podana definicja spełnia przedstawione wymogi, gdyż w wyrażeniu definiującym po prawej stronie nie występuje pojęcie podzbioru, a pojęcia należenia elementu do zbioru, spójnika implikacji i kwantyfikatora ogólnego były wyjaśnione wcześniej. Definicja normalna pozwala przełożyć każdy zwrot językowy zawierający wyrażenie definiowane na zwrot niezawierający tego wyrażenia. Większość definicji podawanych w książce ma postać definicji normalnej.

Łatwo zauważyć, że zachodzą własności:

$$\varnothing \subseteq A$$
$$A \subseteq A$$
$$(A \subseteq B \wedge B \subseteq C) \Rightarrow (A \subseteq C)$$

Używa się też symbolu inkluzji właściwej $\subset$. Zapis $A \subset B$ czytamy: *zbiór A zawiera się właściwie w zbiorze B*. Oznacza to, że A zawiera się w B, czyli $A \subseteq B$, oraz zbiór B zawiera przynajmniej jeden element, który nie należy do zbioru A. Formalnie

$$A \subset B \Leftrightarrow (A \subseteq B) \wedge (\exists a \bullet a \notin A \wedge a \in B)$$

Dwa zbiory A i B są *identyczne* albo *równe*, co oznacza się

$$A = B$$

wtedy i tylko wtedy, gdy mają dokładnie te same elementy, czyli gdy $A \subseteq B$ oraz $B \subseteq A$. Formalnie

$$A = B \Leftrightarrow (A \subseteq B) \wedge (B \subseteq A)$$

Symbol = jest tu symbolem *równości* lub *identyczności* zbiorów.

W zbiorze nie odróżnia się kolejności ani powtórzeń elementów. Na przykład zbiory:

$$A =_{def} \{1, 2, 3\}$$
$$B =_{def} \{1, 3, 2\}$$
$$C =_{def} \{1, 2, 3, 2\}$$

są identyczne, czyli $A = B = C$.

Jeżeli A jest zbiorem, to przez 2^A oznacza się zbiór, którego elementami są wszystkie podzbiory zbioru A. Zbiór 2^A jest nazywany *zbiorem potęgowym* zbioru A. Zbiór potęgowy jest więc rodziną zbiorów.

Uwaga

Zbiór potęgowy zbioru A oznacza się również przez $\mathbb{P}(A)$ albo $\mathcal{P}(A)$.

Przykład 2.2

Dla zbioru $A =_{def} \{a, b, c\}$ jego zbiór potęgowym 2^A jest równy zbiorowi

$$\{\varnothing, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$$

Postać oznaczenia zbioru potęgowego 2^A wynika z następującej własności: Jeżeli A jest zbiorem skończonym, to przez $card(A)$ oznaczamy liczbę jego elementów. Łatwo pokazać, że dla dowolnego skończonego zbioru A zachodzi

$$card(2^A) = 2^{card(A)}$$

Należy zwrócić uwagę na to, że użyty powyżej symbol równości = odnosi się do równości liczb całkowitych, podczas gdy użyty wcześniej ten sam symbol odnosił się do równości zbiorów. Symbol równości w różnych kontekstach może być używany do porównywania obiektów należących do różnych kategorii.

Uwaga

Na określenie liczności elementów skończonego zbioru A używa się również innych oznaczeń, na przykład: $\#(A)$, $|A|$.

W przypadku zbiorów nieskończonych nie można mówić o liczbie ich elementów. Można natomiast porównywać dwa zbiory pod względem równoliczności. Pojęcie równoliczności zbiorów jest zdefiniowane dalej, po wprowadzeniu pojęcia funkcji.

2.4. Operacje na zbiorach

Mając dane pewne zbiory, można z nich budować nowe zbiory. Na zbiorach wykonuje się operacje (działania), których wynikiem są nowe zbiory. Podstawowymi operacjami są: suma, przekrój, różnica i różnica symetryczna dwóch zbiorów. Działania te są zdefiniowane przez podanie własności zbiorów wynikowych.

Suma zbiorów

$$A \cup B =_{\text{def}} \{a \mid a \in A \vee a \in B\}$$

Przekrój zbiorów

$$A \cap B =_{\text{def}} \{a \mid a \in A \wedge a \in B\}$$

Różnica zbiorów

$$A \setminus B =_{\text{def}} \{a \mid a \in A \wedge a \notin B\}$$

Uwaga

Innym oznaczeniem różnicy zbiorów jest $A - B$.

Przykład 2.3

Rozpatrzmy zbiory:

$A =_{\text{def}} \{\{a, b\}, c\}$
$B =_{\text{def}} \{c, d\}$
$C =_{\text{def}} \{\{a, \{a\}\}, a\}$
$D =_{\text{def}} \{a, \{a\}\}$

Łatwo sprawdzić, że:

$A \cup B = \{\{a, b\}, c, d\}$ $A \cap B = \{c\}$ $A \setminus B = \{\{a, b\}\}$
$C \cup D = \{\{a, \{a\}\}, \{a\}, a\}$ $C \cap D = \{a\}$ $C \setminus D = \{\{a, \{a\}\}\}$

Gdy interesujące nas zbiory są podzbiorami pewnego wyróżnionego zbioru, nazywanego *zbiorem uniwersum*, używa się operacji dopełnienia zbioru. Jeżeli U jest uniwersum oraz A jest pewnym jego podzbiorem, to przez A' oznaczamy operację dopełnienia zbioru A, którą definiujemy jako

$$A' =_{\text{def}} U \setminus A$$

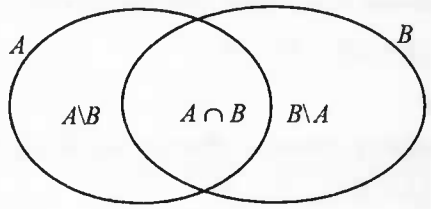
Dwa zbiory A, B nazywa się zbiorami *rozłącznymi*, jeżeli ich przekrój jest pusty, czyli gdy

$$A \cap B = \emptyset$$

Często stosowanym sposobem ilustracji operacji mnogościowych są *wykresy Venna*[5]. Zakłada się w nich, że uniwersum jest zbiór punktów na płaszczyźnie, a rozważanymi

[5] John Venn (1834–1923).

zbiorami są dowolne obszary na płaszczyźnie. Przykład takiego wykresu przedstawiono na rysunku 2.2. Dwa przecinające się owale reprezentują zbiory A oraz B. Poszczególne podobszary oznaczają odpowiednio podzbiory $A\setminus B$, $A \cap B$ oraz $B\setminus A$.



Rys. 2.2. Związki pomiędzy zbiorami

Wprowadzone operacje mają różne własności. Łatwo sprawdzić, że zachodzą następujące własności, nazywane też prawami mnogościowymi:

1. *Prawa przemienności*

$$A \cup B = B \cup A$$
$$A \cap B = B \cap A$$

2. *Prawa łączności*

$$(A \cup B) \cup C = A \cup (B \cup C)$$
$$(A \cap B) \cap C = A \cap (B \cap C)$$

3. *Prawa rozdzielności*

$$(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$$
$$(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$$

4. *Prawa de Morgana*

$$(A \cap B)' = A' \cup B'$$
$$(A \cup B)' = A' \cap B'$$

5. *Prawa dla zbioru pustego*

$$A \cap \emptyset = \emptyset$$
$$A \cup \emptyset = A$$
$$A \setminus \emptyset = A$$
$$\emptyset \setminus A = \emptyset$$
$$\emptyset' = U$$

6. *Prawa dla zbioru uniwersum*

$$A \cap U = A$$
$$A \cup U = U$$
$$A \setminus U = \emptyset$$
$$U' = \emptyset$$

Przykładowo pokażemy jak uzasadnić jedną z tych własności – pierwsze z praw de Morgana.

Własność

Zachodzi następująca równość zbiorów

$$(A \cap B)' = A' \cup B'$$

Dowód

Z definicji równości zbiorów wynika, że należy pokazać dwie inkluzje:

$$(A \cap B)' \subseteq A' \cup B'$$
$$A' \cup B' \subseteq (A \cap B)'$$

Z kolei, z definicji podzbioru wynika, że należy pokazać dwie implikacje:

$$x \in (A \cap B)' \Rightarrow x \in A' \cup B'$$
$$x \in A' \cup B' \Rightarrow x \in (A \cap B)'$$

albo – co oznacza to samo – równoważność

$$x \in (A \cap B)' \Leftrightarrow x \in A' \cup B'$$

W naszym przypadku pokażemy właśnie ostatnią równoważność. Dowód ma postać ciągu zdań połączonych spójnikami równoważności. Po prawej stronie wiersza, w którym występuje równoważność, po symbolu dwóch kresek, jest podany odpowiedni komentarz uzasadniający.

$x \in (A \cap B)' \Leftrightarrow$	-- z definicji operacji dopełnienia
$x \in U \wedge x \notin (A \cap B) \Leftrightarrow$	-- z definicji operatora nienależenia do zbioru
$x \in U \wedge \neg(x \in (A \cap B)) \Leftrightarrow$	-- z definicji iloczynu zbiorów
$x \in U \wedge \neg(x \in A \wedge x \in B) \Leftrightarrow$	-- z prawa de Morgana dla rachunku zdań
$x \in U \wedge (\neg x \in A \vee \neg\, x \in B) \Leftrightarrow$	-- z prawa operatora nienależenia do zbioru
$x \in U \wedge (x \notin A \vee x \notin B) \Leftrightarrow$	-- z prawa rozdzielności koniunkcji względem dysjunkcji
$x \in U \wedge x \notin A \vee x \in U \wedge x \notin B \Leftrightarrow$	-- z definicji operacji dopełnienia
$x \in A' \vee x \in B' \Leftrightarrow$	-- z definicji sumy zbiorów
$x \in A' \cup B'$	■

Uwaga

Przedstawimy kilka komentarzy związanych z dowodem. Pojęcie dowodu będzie omawiane dalej. Schemat przedstawionego tu dowodu ma postać ciągu równoważności

$$p_1 \Leftrightarrow p_1 \ldots \Leftrightarrow p_n$$

Poszczególne równoważności zachodzą na mocy definicji lub wynikają z pewnych reguł wnioskowania.

Równoważność ma własność przechodniości, co oznacza, że można stosować regułę wnioskowania:

$$\frac{p \Leftrightarrow q \quad q \Leftrightarrow r}{p \Leftrightarrow r}$$

gdzie: p, q, r są zdaniami. Na podstawie tej reguły wnioskowania można zatem stwierdzić, że

$$p_1 \Leftrightarrow p_n$$

W treści dowodu, poza powołaniem się na odpowiednie definicje występujących pojęć, powołaliśmy się na dwie własności rachunku zdań: prawa de Morgana i rozdzielność koniunkcji względem dysjunkcji. Własności rachunku zdań będą omówione w dalszej części książki, ale – dla czytelności – przedstawimy je również tutaj. Prawa de Morgana dla rachunku zdań mają postać:

$$\neg (p \wedge q) \Leftrightarrow \neg p \vee \neg q$$
$$\neg (p \vee q) \Leftrightarrow \neg p \wedge \neg q$$

Prawa rozdzielności mają natomiast postać:

$$p \wedge (q \vee r) \Leftrightarrow p \wedge q \vee p \wedge r$$
$$p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$$

Wymienione równoważności noszą miano praw logicznych, co oznacza, że równoważności te są prawdziwe niezależnie od wartościowań zmiennych p, q, r. Można się o tym przekonać, budując i sprawdzając odpowiednie tablice prawdziwościowe (zob. rozdz. 1.).

Zdefiniowane dla dwóch zbiorów operacje sumy i przekroju uogólnia się na dowolne rodziny zbiorów. Niech I będzie dowolnym zbiorem, nazywanym *zbiorem indeksów*, oraz niech $\{A_i \mid i \in I\}$ będzie indeksowaną rodziną zbiorów, wtedy:

$$\bigcup_{i \in I} A_i =_{\text{def}} \{a \mid \exists\, i \in I \bullet\ a \in A_i\}$$

$$\bigcap_{i \in I} A_i =_{\text{def}} \{a \mid \forall i \in I \bullet\ a \in A_i\}$$

są *uogólnioną sumą* i *uogólnionym przekrojem* zbiorów. Uogólnioną sumę i przekrój rodziny zbiorów $\{A_i \mid i \in I\}$ będziemy też zapisywać w postaci:

$$\bigcup \{A_i \mid i \in I\}$$

$$\bigcap \{A_i \mid i \in I\}$$

Przykład 2.4

Niech $A_i =_{def} \{1, 2, ..., i\}$ będzie rodziną zbiorów, gdzie $i \in ParzysteDodatnie$. Łatwo sprawdzić, że:

$$\bigcap_{i \in ParzysteDodatnie} A_i = \{1, 2\}$$

$$\bigcup_{i \in ParzysteDodatnie} A_i = Nat \backslash \{0\}$$

Ćwiczenia

1. Przeanalizować jednoznaczność podanych niżej określeń zbiorów. Jakie związki zachodzą pomiędzy tymi zbiorami?
   a) zbiór ludzi żyjących na jednym kontynencie,
   b) zbiór narodów europejskich,
   c) zbiór zbiorów ludzi posługujących się tym samym językiem,
   d) zbiór obywateli polskich,
   e) zbiór Polaków.

2. Wskazać elementy następujących zbiorów:
   a) $\{a\}$
   b) $\{\{a\}\}$
   c) $\{\{a, b\}, \{a\}\}$
   d) $\{\{\{a\}\}, \{a\}, a\}$
   e) $\{x \in Nat \mid x^2 < 7\}$
   f) $\{x \in Wymierne \mid x^2 = 2\}$
   g) $\{x \in Wymierne \mid (x + 1)^2 < 0\}$

3. Niech A, B, C, D będą parami rozłącznymi, niepustymi zbiorami. Jakie warunki powinny spełniać te zbiory, aby zachodziły następujące równości:
   a) $\{B, C\} = \{B, C, D\}$
   b) $\{\{A, B\}, C\} = \{\{A\}, C\}$
   c) $\{\{A, B\}, \{D\}\} = \{\{A\}\}$
   d) $\{\{A, \varnothing\}, B\} = \{\{\varnothing\}\}$

4. Wykazać, że równość zbiorów $\{\{A\}, \{A, B\}\} = \{\{C\}, \{C, D\}\}$ zachodzi wtedy i tylko wtedy, gdy $A = C$ oraz $B = D$.

5. Obliczyć $A \cap B$, $A \cup B$, $A \backslash B$, $B \backslash A$ dla następujących zbiorów A i B:
   a) $A = \{\{a, b\}, c\}$ $\quad B = \{c, d\}$
   b) $A = \{\{a, \{a\}\}, a\}$ $\quad B = \{a, \{a\}\}$

6. Sprawdzić i uzasadnić, które spośród niżej podanych równości zachodzą bądź nie zachodzą dla dowolnych zbiorów A, B, C, D:
   a) $(A \cup B) \backslash C = (A \backslash C) \cup (B \backslash C)$
   b) $(A \backslash B) \cap (C \backslash D) = (A \cap C) \backslash (B \cup D)$
   c) $(A \cup B) \cap B = B$
   d) $(A \cap B) \cup (A \backslash B) = A$
   e) $(A \backslash B) = A \backslash (A \cap B)$
   f) $(A \backslash B) \cup B = A$

7. Niech $\boldsymbol{U}$ będzie pewnym ustalonym zbiorem, zwanym uniwersum. Jeżeli $A \subseteq \boldsymbol{U}$, to $A' =_{def} \boldsymbol{U} \backslash A$ nazywa się *dopełnieniem* zbioru A. Pokazać, że dla podzbiorów z uniwersum $\boldsymbol{U}$ zachodzą prawa de Morgana:
   a) $(A \cup B)' = A' \cap B'$
   b) $(A \cap B)' = A' \cup B'$

8. Dane są podzbiory A, B, C pewnego uniwersum $\boldsymbol{U}$. Ile co najwyżej różnych zbiorów można otrzymać ze zbiorów A, B, C za pomocą operacji $\cup$, $\cap$, $\backslash$?

9. Różnica symetryczna zbiorów jest zdefiniowana następująco:

   $A \div B =_{def} A/B \cup B/A$

   Pokazać, że zachodzą następujące równości:
   a) $A \div B = B \div A$
   b) $(A \div B) \div C = A \div (B \div C)$
   c) $A \cap (B \div C) = (A \cap B) \div (A \cap C)$
   d) $A \cup (B \div C) = (A \div B) \div (A \cap B)$

10. Zdefiniować operacje $\cup$, $\cap$, $\backslash$ przez:
    a) $\div$, $\cap$
    b) $\div$, $\cup$
    c) $\div$, $/$

11. Ile elementów ma najmniejsza, niepusta rodzina zbiorów $\boldsymbol{A}$ z pewnego uniwersum $\boldsymbol{U}$ taka, że:
    a) jeżeli $A \in \boldsymbol{A}$ i $B \in \boldsymbol{A}$, to $A \cup B \in \boldsymbol{A}$,
    b) jeżeli $A \in \boldsymbol{A}$ i $B \in \boldsymbol{A}$, to $A \cap B \in \boldsymbol{A}$.

12. Udowodnić, że:
    a) $2^{A \cap B} = 2^A \cap 2^B$
    b) $2^{A \cup B} = \{A_1 \cup B_1 \mid A_1 \in 2^A \wedge B_1 \in 2^B\}$

13. Niech $card(A)$ oznacza liczbę elementów zbioru skończonego A. Pokazać, że dla skończonych zbiorów A oraz B zachodzi:
    a) $card(2^A) = 2^{card(A)}$
    b) $card(A \cup B) = card(A) + card(B) - card(A \cap B)$

14. Dla dowolnych zbiorów skończonych A, B i C znaleźć wzory określające:
    a) $card(A \cup B \cup C)$
    b) $card(2^{A \cup B})$
    c) $card(A \setminus B)$

15. Dowieść, że dla dowolnej rodziny zbiorów $A_1, A_2, ..., A_n$, dla $n \in Nat$, zachodzi równość

$$A_1 \cup A_2 \cup ... \cup A_n =$$
$$(A_1 \setminus A_2) \cup (A_2 \setminus A_3) \cup ... \cup (A_{n-1} \setminus A_n) \cup (A_n \setminus A_1) \cup (A_1 \cap A_2 \cap ... \cap A_n)$$

16. Niech $A_1, A_2, ..., A_n$, dla $n > 0$, będą podzbiorami zbioru U. Przez A_i^1 oznaczmy zbiór A_i, a przez A_i^0 oznaczmy dopełnienie tego zbioru, czyli A_i'. Każdy iloczyn postaci:

$$A_1^{i_1} \cap ... \cap A_n^{i_n}$$

    gdzie $i_j \in \{0, 1\}$ dla $j = 1, ..., n$ nazywa się składową.

    a) Pokazać, że różnych składowych jest co najwyżej 2^n.
    b) Pokazać, że różne składowe są rozłączne.
    c) Znaleźć sumę wszystkich składowych.
    d) Udowodnić, że zbiór A_i jest równy sumie tych składowych, w których występuje czynnik postaci A_i^1.

17. Zbadać i udowodnić, które z podanych niżej związków zachodzą dla rodzin zbiorów $\{A_i \mid i \in I\}$, $\{B_i \mid i \in I\}$ oraz $\{C_{i,j} \mid i \in I, j \in J\}$:
    a) $\bigcup_{i \in I} A_i \cup \bigcup_{i \in I} B_i = \bigcup_{i \in I} (A_i \cup B_i)$,
    b) $\bigcup_{i \in I} (A_i \cap B_i) \subseteq \bigcup_{i \in I} A_i \cap \bigcup_{i \in I} B_i$,
    c) $\bigcap_{i \in I} A_i \cup \bigcap_{i \in I} B_i \subseteq \bigcup_{i \in I} (A_i \cup B_i)$,
    d) $\bigcup_{i \in I} \bigcap_{j \in J} C_{i,j} \subseteq \bigcap_{i \in I} \bigcup_{j \in J} C_{i,j}$.

18. Rodzinę $\{A_n \mid n \in Nat\}$ nazywa się *zstępującą* rodziną zbiorów, gdy $A_{n+1} \subseteq A_n$ dla $n \in Nat$. Udowodnić, że jeśli $\{A_n \mid n \in Nat\}$ oraz $\{B_n \mid n \in Nat\}$ są rodzinami zstępującymi, to:

$$\bigcap_{i \in Nat} (A_i \cup B_i) = \left(\bigcap_{i \in Nat} A_i \right) \cup \left(\bigcap_{i \in Nat} B_i \right).$$

    Podać przykłady rodzin zbiorów $\{A_n \mid n \in Nat\}$ oraz $\{B_n \mid n \in Nat\}$, dla których powyższa równość nie zachodzi.

19. Rodzinę $\{A_n \mid n \in Nat\}$ nazywa się *wstępującą* rodziną zbiorów, gdy $A_n \subseteq A_{n+1}$, dla $n \in Nat$. Udowodnić, że jeśli $\{A_n \mid n \in Nat\}$ oraz $\{B_n \mid n \in Nat\}$ są rodzinami wstępującymi, to

$$\bigcup_{i \in Nat} (A_i \cap B_i) = \left(\bigcup_{i \in Nat} A_i \right) \cap \left(\bigcup_{i \in Nat} B_i \right).$$

    Podać przykłady rodzin zbiorów $\{A_n \mid n \in Nat\}$ oraz $\{B_n \mid n \in Nat\}$, dla których powyższa równość nie zachodzi.

20. Funkcja f określona dla liczb rzeczywistych i o wartościach rzeczywistych jest ciągła, jeśli

$$\forall x_0 \in Rzeczywiste \bullet \forall \varepsilon > 0 \bullet \exists \delta > 0 \bullet \forall x \in Rzeczywiste \bullet |x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \varepsilon$$

    Nie używając znaku negacji, zapisać formułę: „funkcja nie jest ciągła".

21. Liczba g jest granicą w punkcie x_0 funkcji f określonej dla liczb rzeczywistych i o wartościach rzeczywistych, jeśli

$$\forall \varepsilon > 0 \bullet \exists \delta > 0 \bullet \forall x \in Rzeczywiste \bullet 0 < |x - x_0| < \delta \Rightarrow |f(x) - g| < \varepsilon$$

    Nie używając znaku negacji, zapisać formułę: „liczba g nie jest granicą funkcji w punkcie x_0".

3. Relacje i funkcje

3.1. Produkty kartezjańskie

Podczas grupowania pewnych elementów w zbiory kolejność ich wyliczenia nie jest istotna. W sytuacji, gdy kolejność jest istotna, elementy się grupuje, używając pojęcia *par uporządkowanych* i *krotek*. Jeżeli $a \in A$ oraz $b \in B$ są dwoma elementami, niekoniecznie różnymi, to zapis

$<a, b>$

oznacza parę uporządkowaną, której komponentami są a oraz b. Uporządkowanie oznacza, że para $<a, b>$ nie jest tym samym, co para $<b, a>$. Dwie pary:

$<a, b>$ oraz $<c, d>$

są *identyczne*, co pisze się $<a, b> = <c, d>$, wtedy i tylko wtedy, gdy:

$a = c$ oraz $b = d$.

Występujący powyżej symbol = ma dwa znaczenia. Gdy pisze się $<a, b> = <c, d>$, oznacza to identyczność dwóch par. Gdy natomiast pisze się $a = b$, oznacza to identyczność dwóch elementów. W obu przypadkach porównuje się ze sobą obiekty różnych kategorii.

Uwaga

Para $<a, b>$ będzie też zapisywana w postaci (a, b).

Symbol identyczności, najczęściej reprezentowany symbolem = lub – rzadziej – symbolem ≈, zasługuje na wyróżnienie, z uwagi na częste użycie w różnych kontekstach. Konteksty te należy odróżniać, a w konkretnym kontekście właściwie rozumieć znaczenie identyczności. Ogólnie, symbol, który może mieć różne znaczenia, nazywa się *symbolem przeciążonym*.

Symbol identyczności, niezależnie od tego, jakie kategorie obiektów porównuje, ma pewne stałe własności. Są to własności *zwrotności*, *symetrii* i *przechodniości*. Niech a, b, c będą obiektami tego samego zbioru. Własność zwrotności oznacza, że dany obiekt jest identyczny ze sobą samym, czyli $a = a$. Własność symetrii oznacza, że jeżeli a jest identyczne z b, to b jest identyczne z a, czyli jeżeli $a = b$, to także $b = a$. Własność przechodniości oznacza, że jeżeli a jest identyczne z b oraz

b jest identyczne z c, to a jest identyczne z c, czyli jeżeli $a = b$ oraz $b = c$, to $a = c$. Symbolicznie własności te można przedstawić w postaci:

$\forall a \in A \bullet a = a$

$\forall a, b \in A \bullet (a = b) \Rightarrow (b = a)$

$\forall a, b, c \in A \bullet (a = b) \wedge (b = c) \Rightarrow (a = c)$

Parę uporządkowaną $<a, b>$ można też wyrazić jako zbiór postaci $\{a, \{a, b\}\}$. Równość zbiorów $\{a, \{a, b\}\}$ oraz $\{c, \{c, d\}\}$ odpowiada wówczas równości odpowiadających im par $<a, b>$ oraz $<c, d>$. W szczególności widać, że dwie pary $<a, b>$ oraz $<b, a>$ są różne, gdyż odpowiadające im zbiory $\{a, \{a, b\}\}$ oraz $\{b, \{a, b\}\}$ nie są identyczne. Posługiwanie się parą uporządkowaną $<a, b>$ zamiast zbiorem $\{a, \{a, b\}\}$ jest wygodniejsze i dlatego dalej będzie używana tylko taka notacja.

Przykład 3.1

Para uporządkowana postaci $<x, y>$, gdzie $x, y \in$ *LiczbyRzeczywiste*, może być interpretowana jako punkt na płaszczyźnie, o współrzędnych x, y.

Pary mają też interpretacje w programowaniu. Pary:

<nazwisko, Bach >
<nazwisko, Kant>,

gdzie: *nazwisko* $\in$ *Ident* oraz *Bach,Kant* $\in$ *Nazwiska* są przykładami danych prostych (wartościami pojedynczych pól rekordów). Podobnie, innymi przykładami danych prostych są pary:

<urodziny, XVII >
<urodziny, XVIII >

gdzie *urodziny* $\in$ *Ident* oraz *XVII, XVIII* $\in$ *LiczbyRzymskie*.

Pary postaci:

<<nazwisko, Bach >, <urodziny, XVII >>
<<nazwisko, Kant>, <urodziny, XVIII >>

reprezentują dane złożone (wartości rekordów złożonych z dwóch pól).

Ogólnie, dla dowolnej liczby naturalnej n definiuje się tzw. *n-krotki*. Jeżeli $a_1, ..., a_n$ są elementami, niekoniecznie różnymi, to

$<a_1, ..., a_n>$

jest *n-krotką*, a $a_1, ..., a_n$ są jej komponentami. Wyróżnia się więc: 0-krotkę $<>$, 1-krotkę $<a>$, 2-krotkę lub parę $<a, b>$, 3-krotkę lub trójkę $<a, b, c>$ itd.

Dwie krotki:

$<a_1, ..., a_n>$ oraz $<b_1, ..., b_m>$

są identyczne wtedy i tylko wtedy, gdy $n = m$ oraz $a_i = b_i$ dla każdego $i = 1, ..., n$.

Krotki:

<1, 1, 1>
<<1, 1>, 1>
<1, <1, 1>>

są więc różne. Pierwsza z nich jest trójką, a pozostałe są parami, w których jedna ze składowych jest również parą.

Produkt (*iloczyn*) *kartezjański* zbiorów A, B jest zbiorem par:

$$A \times B =_{\text{def}} \{<a, b> \mid a \in A \wedge b \in B\}.$$

Zauważmy, że jeżeli zbiory A i B są niepuste oraz $A \neq B$, to $A \times B \neq B \times A$.

Ogólnie – *n-krotny produkt kartezjański zbiorów* $A_1, ..., A_n$, dla $n > 1$, jest zbiorem

$$A_1 \times ... \times A_n =_{\text{def}} \{<a_1, ..., a_n> \mid a_i \in A_i \text{ dla } i = 1, ..., n\}$$

Zamiast pisać $A \times ... \times A$, gdzie A powtarza się n razy, dla $n \in Nat$, pisze się A^n. Z definicji:

$$A^0 =_{\text{def}} \{<>\}$$

$$A^1 =_{\text{def}} A.$$

Uogólnionym produktem kartezjańskim na zbiorze A nazywa się zbiór

$$\bigcup_{n \in Nat} A^n = A^0 \cup A^1 \cup A^2 \cup A^3 \cup ...$$

3.2. Relacje

Określona na zbiorach A oraz B *relacja binarna* R jest podzbiorem produktu kartezjańskiego $A \times B$, czyli $R \subseteq A \times B$.

Jeżeli $A = B$, to mówi się o relacji binarnej na A.

Jeżeli para $<a, b>$ jest elementem relacji binarnej R, to pisze się $<a, b> \in R$. Czasem używa się równoważnego zapisu aRb.

Jeżeli $R_1, R_2 \subseteq A \times B$, to równość relacji $R_1 = R_2$ jest równością zbiorów par reprezentowanych przez R_1 oraz R_2.

Ponownie warto zwrócić uwagę na nową rolę symbolu równości =. Tym razem symbol ten oznacza równość relacji, podczas gdy wcześniej oznaczał równość elementów w obrębie zbioru, równość zbiorów oraz równość krotek.

Zbiór wszystkich relacji binarnych określonych na zbiorach A i B będzie oznaczany przez $2^{A \times B}$, tzn.

$$2^{A \times B} =_{\text{def}} \{R \mid R \subseteq A \times B\}$$

Przy wprowadzonych oznaczeniach zapisy:

$R \subseteq A \times B$ oraz $R \in 2^{A \times B}$

są równoważne.

Uwaga

Na określenie zbioru relacji na produkcie kartezjańskim $A \times B$ używa się również innych oznaczeń, na przykład: $A \leftrightarrow B$ lub $\mathrm{P}(A \times B)$.

Zapis postaci

$R \subseteq A \times B$

nazywa się *sygnaturą relacji*. Symbol R jest *nazwą* relacji, a wyrażenie $A \times B$, gdzie A oraz B są nazwami zbiorów, jest *typem* relacji.

Jeżeli R jest relacją binarną na $A \times B$, to jej *dziedziną* jest zbiór

$$dom(R) =_{\text{def}} \{a \in A \mid \exists\, b \in B \bullet <a, b> \in R\}$$

a jej *przeciwdziedziną* jest zbiór

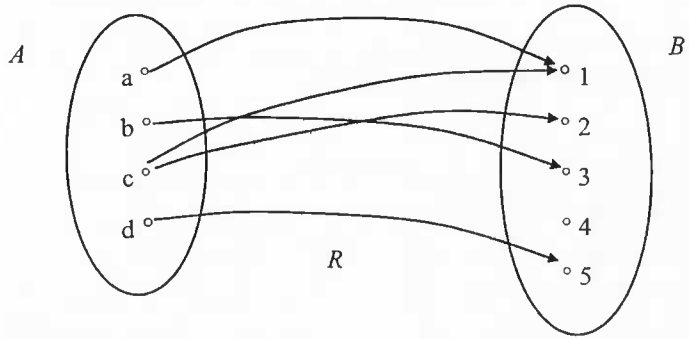
$$ran(R) =_{\text{def}} \{b \in B \mid \exists\, a \in A \bullet <a, b> \in R\}$$

Relacja binarna $R \subseteq A \times B$ ma swoją *relację odwrotną* $R^{-1} \subseteq B \times A$, zdefiniowaną następująco:

$$R^{-1} =_{\text{def}} \{<b, a> \mid <a, b> \in R\}$$

Łatwo zauważyć, że $(R^{-1})^{-1} = R$.

Wprowadzone pojęcia można zilustrować graficznie. Rozpatrzmy przykład relacji binarnej zdefiniowanej na zbiorach $A =_{\text{def}} \{a, b, c, d\}$ oraz $B =_{\text{def}} \{1, 2, 3, 4, 5\}$, przedstawiony na rysunku 3.1.



Rys. 3.1. Graficzna ilustracja relacji

Łuki prowadzące od elementów zbioru A do elementów zbioru B reprezentują pojedyncze pary – elementy relacji R. Z rysunku wynika, że

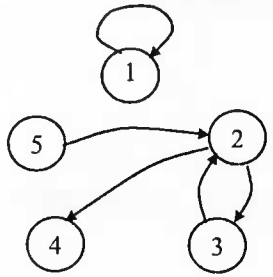
$$R = \{<a, 1>, <b, 3>, <c, 1>, <c, 2>, <d, 5>\}$$

Ponadto $dom(R) = A$ oraz $ran(R) = \{1, 2, 3, 5\} \subset B$. Przedstawienie na rysunku, zgodnie z tą samą konwencją, relacji odwrotnej R^{-1} polegałoby na odwróceniu kierunku strzałek.

Gdy ma się do czynienia z relacjami binarnymi określonymi na jednym zbiorze, czyli relacjami o sygnaturze $R \subseteq A^2$, bardzo przejrzystym sposobem reprezentacji graficznej są grafy. Formalnie grafy są definiowane dalej, tutaj ograniczamy się do przykładu. Niech $A =_{def} \{1, 2, 3, 4, 5\}$ oraz

$$R =_{def} \{<1, 1>, <3, 2>, <2, 3>, <2, 4>, <5, 2>\}$$

Graf reprezentujący relację R jest pokazany na rysunku 3.2.



Rys. 3.2. Graficzna ilustracja relacji R

Wierzchołki grafu reprezentują elementy zbioru A, a łuki grafu reprezentują elementy relacji w taki sposób, że para $<a, b> \in R$ jest reprezentowana przez łuk wychodzący z wierzchołka a i prowadzący do wierzchołka b.

Relacje mają różne zastosowanie w informatyce. Tablice w bazach danych są typowym przykładem relacji.

Przykład 3.2

W pewnej bazie danych dane są dwie tablice:

Tablica 3.1

Nazwisko	Wiek urodzin
Bach	*XVII*
Frege	*XVIII*
Leibnitz	*XVII*
Tarski	*XX*

Tablica 3.2

Nazwisko	Zawód
Bach	*Muzyk*
Frege	*Logik*
Leibnitz	*Filozof*
Tarski	*Matematyk*

Każda z tablic reprezentuje pewną relację. Pierwsza jest relacją typu *Nazwiska* × *Liczby Rzymskie*, druga zaś jest typu *Nazwiska* × *Zawody*. Relacje te można przedstawić w postaci mnogościowej przez wyliczenie odpowiednich par:

$$\{<Bach, XVII>, <Frege, XVIII>, <Leibnitz, XVII>, <Tarski, XX>\}$$

$$\{<Bach, Muzyk>, <Frege, Logik>, <Leibnitz, Filozof>, <Tarski, Matematyk>\}$$

Pojęcie relacji binarnej uogólnia się na *relację n-krotną* R jako dowolny podzbiór n-krotnego, produktu kartezjańskiego

$$R \subseteq A_1 \times ... \times A_n \qquad \text{dla } n \in Nat \setminus \{0, 1\}$$

3.3. Operacje na relacjach

Relacje są zbiorami, można zatem na nich wykonywać wszystkie wcześniej zdefiniowane operacje mnogościowe. Jeżeli na przykład R, $Q \subseteq A \times B$, to zbiory $R \cup Q$, $R \cap Q$, $R \setminus Q$ są również relacjami na produkcie kartezjańskim $A \times B$.

Wprowadza się też specyficzne operacje mnogościowe. Operacje takie występują na przykład w systemach zarządzania bazami danych.

Przykład 3.3

Rozpatruje się operację *złączenia* dwóch relacji $R \subseteq A \times B$ oraz $Q \subseteq A \times C$. Operacja ta, oznaczana tu przez $R \oplus Q$, jest zdefiniowana następująco:

Jeśli $dom(R) = dom(Q)$, to $R \oplus Q \subseteq A \times B \times C$ jest relacją:

$$R \oplus Q =_{def} \{<a, b, c> \mid <a, b> \in R \wedge <a, c> \in Q\}$$

Jeżeli za R oraz Q weźmie się relacje zdefiniowane przez tablicę 3.1 i tablicę 3.2 w poprzednim przykładzie, to widać, że $dom(R) = dom(Q)$, a wynikową relację $R \oplus Q$ przedstawia tablica 3.3.

Tablica 3.3

Nazwisko	Wiek urodzin	Zawód
Bach	*XVII*	*Muzyk*
Frege	*XVIII*	*Logik*
Leibnitz	*XVII*	*Filozof*
Tarski	*XX*	*Matematyk*

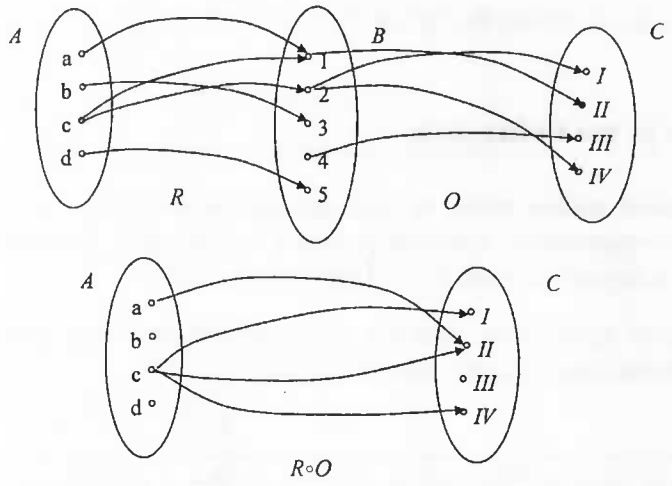
Nową operacją jest *złożenie* (*superpozycja*) dwóch relacji $R \subseteq A \times B$ oraz $Q \subseteq B \times C$. Jest to nowa relacja, zapisywana w postaci $R \circ Q$, zdefiniowana następująco:

$$R \circ Q =_{def} \{<a, c> \mid \exists\, b \in B \bullet <a, b> \in R \wedge <b, c> \in Q\}$$

Graficzną ilustracją złożenia dwóch relacji $R \subseteq A \times B$ oraz $Q \subseteq B \times C$, gdzie

$$A =_{def} \{a, b, c, d\},\ B =_{def} \{1, 2, 3, 4, 5\},\ C =_{def} \{I, II, III, IV\}$$

jest rysunek 3.3.



Rys. 3.3. Graficzna ilustracja złożenia relacji

Górna część rysunku przedstawia relacje R oraz Q, a dolna część – złożenie $R \circ Q$.

Łatwo sprawdzić, że złożenie relacji jest operacją łączną, to znaczy

$$(R \circ Q) \circ S = R \circ (Q \circ S)$$

ale nie jest operacją przemienną, to znaczy

$$R \circ Q \neq Q \circ R$$

Dla dowolnej liczby naturalnej n, *n-krotnym złożeniem* relacji binarej $R \subseteq A^2$ jest relacja R^n, zdefiniowana *indukcyjnie* w sposób następujący:

$$R^0 =_{def} \{<a, a> \mid a \in A\}$$

$$R^{n+1} =_{def} R^n \circ R \text{ dla } n \in Nat$$

Relację R^0 nazywa się *relacją identycznościową* lub *tożsamościową* na A.

Jeżeli $R \subseteq A \times B$, to *obrazem zbioru* $A_1 \subseteq A$ wyznaczonym przez relację R jest zbiór

$$R(A_1) =_{def} \{b \in B \mid \exists\, a \in A_1 \bullet <a, b> \in R\}$$

Łatwo pokazać, że dla $A_1, A_2 \subseteq A$ zachodzą własności:

$$R(A_1 \cup A_2) = R(A_1) \cup R(A_2)$$
$$R(A_1 \cap A_2) \subseteq R(A_1) \cap R(A_2)$$
$$R(dom(R)) = ran(R)$$

Jeżeli $R \subseteq A \times B$, to *przeciwobrazem zbioru* $B_1 \subseteq B$ wyznaczonym przez relację R jest zbiór $R^{-1}(B_1)$.

3.4. Podstawowe rodzaje relacji binarnych

Relacje binarne na A, czyli relacje $R \subseteq A^2$, mogą się charakteryzować różnymi własnościami. Wśród podstawowych własności wyróżnia się między innymi własności *zwrotności*, *przeciwzwrotności*, *symetrii*, *przeciwsymetrii*, *antysymetrii*, *przechodniości* i *spójności*. Własności te są definiowane następująco:

zwrotność
dla dowolnego $a \in A$ zachodzi: $<a, a> \in R$
– symbolicznie: $\forall a \in A \bullet <a, a> \in R$

przeciwzwrotność
dla dowolnego $a \in A$ zachodzi: $<a, a> \notin R$
– symbolicznie: $\forall a \in A \bullet <a, a> \notin R$

symetria
dla dowolnych $a, b \in A$ zachodzi:
jeżeli $<a, b> \in R$, to również $<b, a> \in R$
– symbolicznie: $\forall a, b \in A \bullet <a, b> \in R \Rightarrow <b, a> \in R$

przeciwsymetria
dla dowolnych $a, b \in A$ zachodzi:
jeżeli $<a, b> \in R$, to $<b, a> \notin R$
– symbolicznie: $\forall a, b \in A \bullet <a, b> \in R \Rightarrow <b, a> \notin R$

antysymetria
dla dowolnych $a, b \in$ A zachodzi:
jeżeli $<a, b> \in R$ oraz $<b, a> \in R$, to $a = b$
– symbolicznie: $\forall a, b \in A \bullet <a, b> \in R \wedge <b, a> \in R \Rightarrow a = b$

przechodniość
dla dowolnych $a, b, c \in A$ zachodzi:
jeżeli $<a, b> \in R$ oraz $<b, c> \in R$, to $<a, c> \in R$
– symbolicznie: $\forall a, b, c \in A \bullet <a, b> \in R \wedge <b, c> \in R \Rightarrow <a, c> \in R$

spójność
dla dowolnych $a, b \in A$ zachodzi:
jeżeli $a \neq b$, to $<a, b> \in R$ lub $<b, a> \in R$
– symbolicznie: $\forall a, b \in A \bullet a \neq b \Rightarrow <a, b> \in R \vee <b, a> \in R$

Niekiedy mamy do czynienia z sytuacjami, gdy zachodzi potrzeba takiego rozszerzenia relacji, aby uzyskały pewne z przedstawionych własności. Niech R będzie dowolną relacją binarną na A. *Zwrotnym domknięciem* relacji R jest relacja zdefiniowana jako

$$R \cup R^0$$

gdzie R^0 jest *relacją identycznościową* na A.

Symetrycznym domknięciem relacji R jest relacja zdefiniowana jako

$$R \cup \{<b, a> \mid <a, b> \in R\} \text{ albo } R \cup R^{-1}$$

Przechodnim domknięciem relacji R jest relacja oznaczana symbolem R^+, zdefiniowana jako

$$R^+ =_{\text{def}} \bigcup_{n \in Nat \setminus \{0\}} R^n$$

Zwrotne, przechodnie (*tranzytywne*) *domknięcie* relacji R na zbiorze A jest to relacja, oznaczana symbolem R^*, zdefiniowana następująco:

$$R^* =_{\text{def}} \bigcup_{n \in Nat} R^n$$

Podane wyżej definicje domknięcia relacji oznaczają, że po domknięciu relacja ma odpowiednio własność zwrotności, symetrii i przechodniości. Domknięcie relacji uogólnia się ze względu na dowolną własność w sposób następujący: Niech P będzie pewną własnością oraz R pewną relacją binarną. Mówimy, że relacja R_P jest *domknięciem relacji R względem własności P* wtedy i tylko wtedy, gdy:

1. relacja R_P ma własność P,
2. $R \subseteq R_P$,
3. nie istnieje inna relacja Q, która ma własność P taką, że $Q \subseteq R_P$.

3.5. Relacja równoważności

Bardzo ważna jest relacja *równoważności*, będąca dowolną relacją binarną, która jest zwrotna, symetryczna i przechodnia.

Dla relacji równoważności R określonej na zbiorze A definiuje się zbiory nazywane *klasami abstrakcji*. Dla dowolnego elementu a zbioru A definiuje się mianowicie zbiór

$$\{b \in A \mid <a, b> \in R\}$$

Zbiór taki oznacza się przez $[a]_R$ i nazywa się *klasą abstrakcji generowaną przez element a względem relacji R*.

Twierdzenie 3.1

Zbiór klas abstrakcji ma następujące własności:

1. $\bigcup_{a \in A} [a]_R = A$
2. $<a, b> \in R$ wtedy i tylko wtedy, gdy $[a]_R = [b]_R$
3. jeżeli $[a]_R \neq [b]_R$, to $[a]_R \cap [b]_R = \varnothing$

Dowód

Własność 1

Z definicji równości zbiorów wynika, że należy pokazać:

a) $\bigcup_{a \in A} [a]_R \subseteq A$

b) $A \subseteq \bigcup_{a \in A} [a]_R$

Przypadek a)

Niech $x \in \bigcup_{a \in A} [a]_R$. Należy pokazać, że $x \in A$.

Z definicji uogólnionej sumy zbiorów wynika, że istnieje takie $a \in A$, że $x \in [a]_R$.

Z definicji klasy abstrakcji wynika, że $[a]_R \subseteq A$.

Z obu tych faktów, na podstawie definicji podzbioru, wynika, że $x \in A$, czyli pokazaliśmy, że $x \in \bigcup_{a \in A} [a]_R \Rightarrow x \in A$.

Przypadek b)

Niech $x \in A$. Należy pokazać, że $x \in \bigcup_{a \in A} [a]_R$.

Z definicji klasy abstrakcji i własności zwrotności relacji R wynika, że $x \in [x]_R$.

Z tego faktu zatem, na mocy definicji uogólnionej sumy zbiorów, wynika, że $x \in \bigcup_{a \in A} [a]_R$

Własność 2

Z definicji równoważności wynika, że należy pokazać dwie implikacje:

a) jeżeli $<a, b> \in R$, to $[a]_R = [b]_R$,

b) jeżeli $[a]_R = [b]_R$, to $<a, b> \in R$.

Przypadek a)

Niech $<a, b> \in R$. Należy pokazać, że $[a]_R = [b]_R$.

Pokażemy, że jeśli $<a, b> \in R$, to $[a]_R \subseteq [b]_R$ oraz, że jeśli $<a, b> \in R$, to $[b]_R \subseteq [a]_R$.

Z założenia, że $<a, b> \in R$ i z definicji klasy abstrakcji wynika, że $b \in [a]$.

Niech $x \in [a]_R$. Z definicji klasy abstrakcji wynika, że $<a, x> \in R$, co z własności symetrii relacji równoważności pociąga, że $<x, a> \in R$.

Z własności przechodniości relacji równoważności, na podstawie stwierdzeń, że $<x, a> \in R$ oraz $<a, b> \in R$ wynika, że $<x, b> \in R$. Ponownie, z własności symetrii relacji R oraz z definicji klasy abstrakcji wynika, że $x \in [b]_R$.

Pokazaliśmy, że $x \in [b]_R \Rightarrow x \in [b]_R$, czyli $[a]_R \subseteq [b]_R$.

Wykazanie, że jeśli $<a, b> \in R$, to $[b]_R \subseteq [a]_R$ przebiega analogicznie do pokazanego wyżej.

Przypadek b)

Niech $[a]_R = [b]_R$. Należy pokazać, że $<a, b> \in R$.

Z definicji klasy abstrakcji i zwrotności relacji wynika, że $a \in [a]_R$ i $b \in [b]_R$. Na podstawie równości zbiorów $[a]_R = [b]_R$ stwierdzamy, że $a, b \in [a]_R$, stąd – na podstawie definicji klasy abstrakcji – wynika, że $<a, b> \in R$.

Własność 3

Należy pokazać, że jeżeli $[a]_R \neq [b]_R$, to $[a]_R \cap [b]_R = \emptyset$.

Dowód przeprowadzimy metodą nie wprost. Metodę tę stosuje się do twierdzeń, które mają postać implikacyjną. W rozważanym przypadku założeniem – poprzednikiem implikacji – jest $[a]_R \neq [b]_R$), a tezą – następnikiem implikacji – jest $[a]_R \cap [b]_R = \emptyset$. Dowód nie wprost polega na przyjęciu założenia, nazywanego założeniem dowodu nie wprost, które jest negacją tezy. Następnie należy pokazać, że z tego założenia wynika sprzeczność z założeniem twierdzenia.

W rozpatrywanym przypadku należy pokazać, że

jeżeli $[a]_R \cap [b]_R \neq \emptyset$, to $[a]_R = [b]_R$.

Jeżeli $[a]_R \cap [b]_R \neq \emptyset$, to oznacza, że istnieje element $x \in [a]_R \cap [b]_R$. Z definicji przekroju zbiorów wynika, że $x \in [a]_R$ i $x \in [b]_R$, co na podstawie definicji klasy abstrakcji oznacza, że $<a, x> \in R$ oraz $<b, x> \in R$. Z własności symetrii i przechodniości relacji R wynika, że $<a, b> \in R$, stąd – na mocy udowodnionej wyżej własności 2 – wynika, że $[a]_R \cap [b]_R = \emptyset$, co oznacza sprzeczność z założeniem twierdzenia. ■

Z udowodnionych własności wynika, że jeżeli zbiorze A jest zdefiniowana relacja równoważności, to relacja ta wyznacza podział zbioru A na rozłączne podzbiory (klasy abstrakcji). Podział taki nazywa się też *partycją*. Zbiór, którego elementami są wszystkie klasy abstrakcji, nazywa się *zbiorem ilorazowym* zbioru A względem relacji R i oznacza się A/R, czyli

$$A/R =_{def} \{[a]_R \mid a \in A\}$$

Przykład 3.4

Niech $A =_{def} \{1, 2, 3, 4, 5\}$ oraz relacja $R \subseteq A^2$ jest zdefiniowana następująco:

$$R =_{def} \{<1, 1>, <2, 2>, <3, 3>, <4, 4>, <5, 5>, <3, 2>, <2, 3>, <2, 5>, <5, 2>, <3, 5>, <5, 3>\}$$

Łatwo sprawdzić, że relacja jest zwrotna, symetryczna i przechodnia, czyli jest relacją równoważności. Wyznaczone przez poszczególne elementy zbioru A klasy równoważności są następujące:

$[1]_R = \{1\}$
$[2]_R = [3]_R = [5]_R = \{2, 3, 5\}$
$[4]_R = \{4\}$

Zbiór ilorazowy A/R wyznaczony przez relację R ma postać

$$A/R = \{\{1\}, \{2, 3, 5\}, \{4\}\}$$

3.6. Relacje porządku

Oprócz relacji równoważności ważną grupę stanowią *relacje porządku*. Wyróżnia się:

- relację *quasi-porządkującą*, gdy jest zwrotna i przechodnia,
- relację *częściowo porządkującą w ścisłym sensie*, gdy jest antysymetryczna i przechodnia,
- relację *częściowo porządkującą*, gdy jest zwrotna, antysymetryczna i przechodnia,
- relację *liniowego porządku w ścisłym sensie*, gdy jest antysymetryczna, przechodnia i spójna,
- relację *liniowego porządku* (czasem też krótko: *relację porządku*), gdy jest zwrotna, antysymetryczna, przechodnia i spójna, czyli, gdy jest relacją częściowego porządku i relacją spójną.

Zbiór, na którym jest określona pewna relacja porządku częściowego, nazywa się *zbiorem częściowo uporządkowanym*, a zbiór, na którym określono relację porządku liniowego – *zbiorem liniowo* (albo *całkowicie*) *uporządkowanym*.

Przykład 3.5

Rozważmy rodzinę wszystkich podzbiorów dowolnego zbioru U, czyli zbiór potęgowy 2^U. Zbiór potęgowy 2^U jest zbiorem częściowo uporządkowanym przez relację $R \subseteq 2^U \times 2^U$, która jest określona następująco:

$$R = \{<A, B> \in 2^U \times 2^U \mid A \subseteq B\}$$

Relacja R jest relacją częściowego porządku. Istotnie, R jest relacją zwrotną, gdyż dla dowolnego $A \in 2^U$ zachodzi $<A, A> \in R$, dlatego, że $A \subseteq A$. R jest relacją antysymetryczną, gdyż – jeżeli $<A, B> \in R$ oraz $<B, A> \in R$ – co oznacza, że $A \subseteq B$ oraz $B \subseteq A$, to $A = B$. R jest też relacją przechodnią, gdyż z faktu, że $<A, B> \in R$ oraz $<B, C> \in R$, co oznacza, że $A \subseteq B$ oraz $B \subseteq C$, wynika, że $<A, C> \in R$, co oznacza, że $A \subseteq C$. Relacja R nie jest natomiast relacją liniowego porządku, gdyż nie jest spójna.

Przykład 3.6

Relacją liniowego porządku jest relacja $\leq$ określona na różnych zbiorach liczbowych, na przykład *Nat*, *Całkowite*, *Wymierne* lub *Rzeczywiste*. Używając tej relacji, posługujemy się notacją $a \leq b$, zamiast $<a, b> \in \leq$. Zachodzą oczywiście własności:

$\forall a \in Rzeczywiste \bullet a \leq b$

$\forall a, b \in Rzeczywiste \bullet a \leq b \wedge a \leq b \Rightarrow a = b$

$\forall a, b, c \in Rzeczywiste \bullet a \leq b \wedge b \leq c \Rightarrow a \leq c$

$\forall a, b \in Rzeczywiste \bullet a \neq b \Rightarrow a \leq b \vee a \leq b$

Nie jest natomiast relacją liniowego porządku relacja $<$, gdyż nie spełnia wymogu zwrotności, to znaczy nie jest prawdą, że $a < a$ dla dowolnego $a \in Rzeczywiste$.

Zestaw relacji częściowego porządku $R_1, ..., R_n$, zdefiniowanych odpowiednio na zbiorach $A_1, ..., A_n$, można wykorzystać do zdefiniowania nowej relacji częściowego porządku R, zdefiniowanej na produkcie kartezjańskim $A_1 \times ... \times A_n$. Przykładem jest *leksykograficzne złożenie* relacji $R_1, ..., R_n$, określone jako relacja R, nazywana *relacją porządku leksykograficznego* (alfabetycznego), zdefiniowana na $A_1 \times ... \times A_n$ w sposób następujący:

$$< a_1, ..., a_n > R < b_1, ..., b_n >$$

wtedy i tylko wtedy, gdy istnieje $i \in \{1, ..., n\}$ takie, że dla każdego $j < i$ zachodzi $a_j = b_j$ oraz $a_i \, R_i \, b_i$.

Niektóre własności relacji częściowego porządku można wyrazić za pomocą tzw. elementów wyróżnionych. Niech $\preccurlyeq$ będzie relacją częściowego porządku na zbiorze A oraz niech $B \subseteq A$.

Uwaga

Symbol $\preccurlyeq$ jest często stosowany na oznaczenie relacji częściowego porządku ze względu na graficzne podobieństwo do symbolu $\leq$ przy jednoczesnym podkreśleniu, że dziedziną relacji nie muszą być tylko zbiory liczbowe.

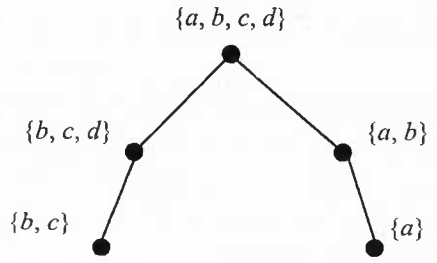
Mówimy, że $a \in A$ jest:

- *elementem najmniejszym* w zbiorze B, jeśli $\forall b \in B \bullet a \preccurlyeq b$,
- *elementem największym* w zbiorze B, jeśli $\forall b \in B \bullet b \preccurlyeq a$,
- *elementem minimalnym* w zbiorze B, jeśli $\forall b \in B \bullet \neg(b \preccurlyeq a)$,
- *elementem maksymalnym* w zbiorze B, jeśli $\forall b \in B \bullet \neg(a \preccurlyeq b)$,
- *ograniczeniem dolnym* zbioru B, jeśli $\forall b \in B \bullet a \preccurlyeq b$ (zauważmy, że a nie musi należeć do zbioru B, element najmniejszy zbioru B jest zatem jego ograniczeniem dolnym),
- *ograniczeniem górnym* zbioru B, jeśli $\forall b \in B \bullet b \preccurlyeq a$ (zauważmy, że a nie musi należeć do zbioru B, element największy zbioru B jest zatem jego ograniczeniem górnym),
- *kresem dolnym* (*infimum*) zbioru B, jeśli a jest elementem największym w zbiorze wszystkich ograniczeń dolnych zbioru B,
- *kresem górnym* (*supremum*) zbioru B, jeśli a jest elementem najmniejszym w zbiorze wszystkich ograniczeń górnych zbioru B.

Przykład 3.7

Rozpatrzmy zbiór potęgowy A zbioru $\{a, b, c, d\}$, czyli $A = 2^{\{a, b, c, d\}}$, z uporządkowaniem częściowym, określonym przez inkluzję. Niech $B = \{\{a\}, \{a, b\}, \{b, c\}, \{b, c, d\}, \{a, b, c, d\}\}$. W zbiorze B są dwa elementy minimalne $\{a\}$ i $\{b, c\}$ oraz jeden element największy $\{a, b, c, d\}$.

Ilustracją związków pomiędzy elementami zbioru częściowo uporządkowanego jest diagram (graf – zob. p. 3.10) Haasego na rysunku 3.4. Wierzchołki diagramu reprezentują elementy zbioru, a krawędzie diagramu łączą ze sobą te wierzchołki x, y, tu elementy zbioru B, pomiędzy którymi zachodzi związek $x \preccurlyeq y$ oraz nie istnieje taki element z, że $x \preccurlyeq z$ oraz $z \preccurlyeq y$.



Rys. 3.4. Diagram Haasego dla zbioru B

W zbiorze B nie ma elementu najmniejszego, element $\{a, b, c, d\}$ jest natomiast jego kresem górnym.

Pomiędzy elementami wyróżnionymi zachodzą następujące związki:

Twierdzenie 3.2

Niech $\preccurlyeq$ będzie relacją częściowego porządku na zbiorze A oraz niech $B \subseteq A$, wtedy:

1. W zbiorze B istnieje co najwyżej jeden element największy i co najwyżej jeden element najmniejszy.
2. Zbiór B ma co najwyżej jeden kres górny i jeden kres dolny.
3. Jeśli b jest największym elementem w zbiorze B, to jest on jedynym elementem maksymalnym w zbiorze B oraz jego kresem górnym.
4. Jeśli b jest najmniejszym elementem w zbiorze B, to jest on jedynym elementem minimalnym w zbiorze B oraz jego kresem dolnym.

Dowód pozostawiamy jako ćwiczenie.

3.7. Funkcje

Niech A oraz B będą dwoma dowolnymi zbiorami. *Funkcją* albo *odwzorowaniem* z A w B nazywa się taką relację binarną $f \subseteq A \times B$, że dla każdego elementu $a \in A$ istnieje co najwyżej jeden element $b \in B$ taki, że $\langle a, b \rangle \in f$.

Inne, równoważne sformułowanie tej samej własności:

dla każdego elementu $a \in A$, jeżeli $\langle a, b \rangle \in f$ oraz $\langle a, c \rangle \in f$, to $b = c$

W symbolicznym zapisie własność ta przyjmuje postać

$$\forall a \in A \bullet \langle a, b \rangle \in f \wedge \langle a, c \rangle \in f \Rightarrow b = c$$

W podanej definicji funkcji f zawiera się możliwość, że dla danego $a \in A$ nie istnieje taki element $b \in B$, że $\langle a, b \rangle \in f$. Oznacza to, że dla $a \in A$ funkcja f jest nieokreślona. Fakt, że para $\langle a, b \rangle \in f$ jest elementem funkcji f, będzie również zapisywany w postaci

$$f(a) = b.$$

Element a nazywa się *argumentem* funkcji f, a b nazywa się *wartością* funkcji f dla argumentu a.

Napis

$$f : A \to B$$

nazywa się *sygnaturą* funkcji; symbol f jest *nazwą* funkcji, a wyrażenie $A \to B$, gdzie A oraz B są nazwami zbiorów, jest *typem* funkcji.

Ogólnie, funkcja może mieć n argumentów ($n \in Nat$). Sygnatura takiej funkcji ma postać

$$f : A_1 \times \ldots \times A_n \to B$$

W skrajnym przypadku, gdy funkcja ma zero argumentów – nazywa się ją funkcją *zeroargumentową* lub *stałą*, a jej sygnaturę zapisujemy

$$f : \to B$$

Dla funkcji o sygnaturze $f : A_1 \times \ldots \times A_n \to B$, fakt, że $\langle a_1, \ldots, a_n, b \rangle \in f$, zapisuje się również w postaci

$$f(a_1, \ldots, a_n) = b.$$

Zapis wartości funkcji w postaci

$$f(a_1, \ldots, a_n)$$

jest zapisem w tak zwanej konwencji *prefiksowej* lub *przedrostkowej*. Inną konwencją, która nie będzie używana, jest notacja *przyrostkowa* (*postfiksowa*), nazywana też *odwrotną notacją polską*, na cześć polskiego logika Łukasiewicza[6], który ją wprowadził. W tej notacji zapisowi wartości funkcji dla argumentów $a_1, \ldots, a_n$ odpowiada zapis

$$(a_1, \ldots, a_n)f$$

Notacja ta jest stosowana w algorytmicznym obliczaniu wartości wyrażeń stanowiących złożenie wielu funkcji.

W przypadku funkcji dwuargumentowych, oprócz podanych notacji, powszechnie stosuje się notację *wrostkową* (*infiksową*). Wartość funkcji $f(a_1, a_2)$, dla argumentów a_1, a_2, zapisuje się w postaci

$$a_1 f a_2$$

Deklarację użycia notacji infiksowej można zaznaczyć w sygnaturze, pisząc

$$\_f\_ : A_1 \times A_2 \to B$$

Podkreślenia po obu stronach symbolu f wskazują na miejsca umieszczania jej argumentów.

Niech $f : A \to B$. Tak jak poprzednio, przez $dom(f)$ i $ran(f)$ oznacza się odpowiednio *dziedzinę* i *przeciwdziedzinę* funkcji f.

Jeżeli $dom(f) = A$, to f nazywa się funkcją *całkowicie określoną*, albo – krótko – *całkowitą*. Zbiór wszystkich funkcji całkowicie określonych z A do B oznacza się $A \to B$. Zbiór A nazywa się zbiorem *źródłowym*, a B – zbiorem *docelowym* funkcji. Zbiór wszystkich funkcji całkowitych z A w B oznacza się też przez B^A.

[6] Jan Łukasiewicz (1878–1956).

W przypadku, gdy $dom(f) \subset A$, funkcję f nazywa się *częściowo określoną*, albo – krótko – *częściową*. Funkcja częściowa f jest nieokreślona dla elementów nienależących do jej dziedziny, czyli do zbioru $A \backslash dom(f)$. Elementowi $a \in A \backslash dom(f)$ nie odpowiada żaden element ze zbioru B. Fakt ten zapisuje się niekiedy, pisząc $f(a) = \perp$, gdzie symbol $\perp$ oznacza *niezdefiniowane*.

Wykorzystując symbol $\perp$, zbiór wszystkich funkcji z A do B oznacza się $(B \cup \perp)^A$.

Jeżeli $ran(f) = B$, to funkcję f nazywa się *surjekcją* albo funkcją „*na*".

Jeżeli dla dwóch różnych argumentów a_1, a_2 funkcja f przyjmuje różne wartości $f(a_1)$, $f(a_2)$, to nazywa się ją funkcją *różnowartościową* albo *injekcją*.

Funkcję f, która jest całkowicie określona, jest surjekcją oraz injekcją, nazywa się funkcją *wzajemnie jednoznaczną* albo *bijekcją*.

Bijekcję, która jest funkcją o tej samej dziedzinie i przeciwdziedzinie, czyli o sygnaturze $f: A \rightarrow A$, nazywa się *permutacją*.

Funkcję f nazywa się funkcją *skończoną*, gdy dziedzina funkcji $dom(f)$ jest zbiorem skończonym.

Jeżeli relacja odwrotna f^{-1} dla funkcji $f: A \rightarrow B$ jest funkcją, to nazywa się ją *funkcją odwrotną* funkcji f.

Przykład 3.8

Niech $A =_{def} \{1, 2, 3, 4, 5\}$ oraz $B =_{def} \{1, 2, 3, 4\}$.

Relacja $R \subseteq A \times B$, zdefiniowana jako zbiór par:

$\{<1, 1>, <2, 3>, <1, 4>, <3, 3>, <4, 4>, <2, 4>, <5, 1>\}$

nie jest funkcją.

Funkcja $f: A \rightarrow B$, zdefiniowana jako zbiór par:

$\{<1, 1>, <3, 5>, <4, 3>, <5, 1>\}$

jest funkcją częściową, gdyż $dom(f) = \{1, 3, 4, 5\} \subset A$.

Funkcja $g: A \rightarrow B$, zdefiniowana jako zbiór par:

$\{<1, 1>, <2, 3>, <3, 4>, <4, 3>, <5, 2>\}$

jest funkcją „*na*", gdyż $ran(f) = B$.

Funkcja $h: A \rightarrow A$, zdefiniowana jako zbiór par:

$\{<1, 1>, <2, 3>, <3, 4>, <4, 3>, <5, 1>\}$

nie ma funkcji odwrotnej.

Szczególną formą enumeracyjnej definicji funkcji jest tablica lub krotka. Na przykład wyżej zdefiniowaną funkcję f można przedstawić w postaci tablicy

$f =$

1	2	3	4	5
1	*	5	3	1

lub krotki

$h = <1, *, 5, 3, 1>$,

gdzie symbol * oznacza, że dla danego argumentu funkcja jest niezdefiniowana.

Przedstawienie funkcji w postaci krotki wymaga dodatkowo, aby dziedzina funkcji była zbiorem liniowo uporządkowanym. Tak jest oczywiście w przypadku dziedziny funkcji f, gdzie porządek w zbiorze A jest wyznaczony przez relację $\leq$.

Funkcje mogą być określane na dowolnych zbiorach. Elementami takich zbiorów mogą być złożone twory, na przykład inne funkcje. W takich przypadkach funkcje nazywa się *funkcjonałami*.

Przykład 3.9

Rozpatrzmy zbiór *FUN*, którego elementami są jednoargumentowe funkcje określone na zbiorze liczb rzeczywistych i o wartościach w zbiorze liczb rzeczywistych *Rzeczywiste*. Z definicji jest to zbiór, którego elementami są funkcje typu

Rzeczywiste → *Rzeczywiste*

Rozpatrzmy operator różniczkowania funkcji *Diff*. Jest to funkcja o sygnaturze

Diff: *FUN* → *FUN*

albo, w postaci rozwiniętej, o sygnaturze

Diff: (*Rzeczywiste* → *Rzeczywiste*) → (*Rzeczywiste* → *Rzeczywiste*)

Operator *Diff* jest funkcją częściową, gdyż istnieją funkcje, które nie mają pochodnej w żadnym punkcie. Istnieją też funkcje całkowicie określone, które mają punkty nieciągłości (są nieróżniczkowalne w tych punktach). Dla takich funkcji operator różniczkowania wyznacza funkcje częściowo określone.

W rozważanych wyżej przykładach funkcje były definiowane enumeracyjnie. Często spotykanym sposobem jest definiowanie funkcji przez wyrażenia funkcyjne. Definicja ma postać *równości*, na przykład

$f(x, y, z) = x * y + 10*z$

Jest to *równość*, po lewej stronie której występuje symbol funkcji z listą zmiennych (argumentów), a po prawej stronie występuje *wyrażenie funkcyjne* (*term*).

W przykładowym wyrażeniu funkcje +, * są znanymi dwuargumentowymi funkcjami arytmetycznymi, 10 jest funkcją zeroargumentową, czyli stałą, a x, y są zmiennymi. Wyrażenie funkcyjne jest więc złożeniem pewnych funkcji. Ogólnie jest ono definiowane następująco:

- stała i zmienna są wyrażeniami funkcyjnymi,
- jeżeli h jest n-argumentową funkcją oraz $g_1, ..., g_n$ są wyrażeniami funkcyjnymi, to $h(g_1, ..., g_n)$ jest wyrażeniem funkcyjnym.

Występujący po lewej stronie symbol funkcji nie może wystąpić po prawej stronie równości. Jedynymi zmiennymi, które mogą występować po prawej stronie równości, są tylko te, które występują po lewej stronie.

Funkcje są pewnymi zbiorami i mogą być definiowane rekursywnie oraz przez określenie własności. Definicję rekursywną, czyli algorytmiczną, funkcji nazywa się też definicją *intensjonalną*, a definicję przez określenie własności – definicją *ekstensjonalną*.

Przykład 3.10

Funkcja *Silnia* jest typu $Nat \rightarrow Nat$. Jej definicja rekursywna ma postać:

$$Silnia(0) = 1$$
$$Silnia(n) = n * Silnia(n-1) \qquad \text{dla } n > 0$$

Definicja składa się z dwóch równości. Po lewej stronie równości występuje symbol definiowanej funkcji wraz z odpowiednimi wartościami argumentu, a po prawej stronie występują wyrażenia funkcyjne. Wyrażenie funkcyjne w pierwszej równości jest stałą (funkcją zeroargumentową), a w drugiej – jest złożeniem funkcji trzech funkcji: odejmowania –, mnożenia * oraz definiowanej funkcji *Silnia*. Pierwsza równość definiuje wartość funkcji dla argumentu o wartości 0, druga – definiuje wartość funkcji dla pozostałych wartości argumentu. Zastosowanie drugiej równości do obliczenia wartości funkcji dla argumentu n wymaga uprzedniego obliczenia wartości funkcji dla argumentu $n - 1$.

W podobny sposób jest zdefiniowana rekursywnie funkcja $M : Nat \rightarrow Nat$:

$$M(1) = 2$$
$$M(2) = 2$$
$$M(n) = 2 * M(n-1) + M(n-2) \qquad \text{dla } n > 2$$

Znacznie bardziej złożony jest sposób definicji funkcji Ackermanna o sygnaturze $Ack: Nat \times Nat \rightarrow Nat$

$$Ack(x, y) = y + 1 \qquad \text{gdy } x = 0$$
$$Ack(x, y) = Ack(x-1, 1) \qquad \text{gdy } y = 0$$
$$Ack(x, y) = Ack(x-1), Ack(x, y-1)$$

Przykład 3.11

Niech, jak poprzednio, *FUN* oznacza zbiór, którego elementami są jednoargumentowe funkcje określone na zbiorze liczb rzeczywistych i o wartościach w zbiorze liczb rzeczywistych, czyli funkcje typu $Rzeczywiste \rightarrow Rzeczywiste$. Dla dowolnej funkcji $f : Rzeczywiste \rightarrow Rzeczywiste$ rozważa się równanie postaci $f(x) = 0$. Równanie to może nie mieć pierwiastków rzeczywistych, może też mieć ich nieskończenie wiele. Przez $Pierwiastki(f)$ oznacza się wartość funkcji, która dla danej funkcji f *wyznacza* podzbiór liczb rzeczywistych P, które są pierwiastkami równania $f(x) = 0$. Funkcja *Pierwiastki* jest typu $FUN \rightarrow 2^{Rzeczywiste}$. Funkcję tę można zdefiniować ekstensjonalnie w sposób następujący:

$$Pierwiastki = \{<f, P> \in FUN \times 2^{Rzeczywiste} \mid x \in P \Leftrightarrow f(x) = 0\}$$

Wprawdzie funkcja *Pierwiastki* jest zdefiniowana jednoznacznie, jednak z definicji tej nie wynika jak dla konkretnej funkcji f określić zbiór jej pierwiastków. Wiadomo, że znajdywanie pierwiastków rzeczywistych równania $f(x) = 0$ jest zadaniem rozwiązywalnym efektywnie tylko dla pewnych klas funkcji.

Przykład 3.12

Rozważmy funkcję *WartośćWielomianu*, która oblicza wartość dowolnego wielomianu dla zadanego argumentu. Jest to funkcja o sygnaturze

$$WartośćWielomianu : Wielomiany \times Rzeczywiste \rightarrow Rzeczywiste$$

Wielomian n-tego stopnia

$$a_n * x^n + ... + a_1 * x + a_0$$

gdzie $n \in Nat$, jest jednoznacznie określony przez zestaw swoich $n + 1$ współczynników $a_n, ..., a_1, a_0 \in Rzeczywiste$. Zbiór *Wielomiany* może zatem być zdefiniowany jako

$$Wielomiany =_{def} \bigcup_{n \in Nat} Rzeczywiste^n$$

stąd, dla dowolnego $<a_n, ..., a_1, a_0> \in Wielomiany$ oraz $x \in Rzeczywiste$

$$WartośćWielomianu(<a_n, ..., a_1, a_0>, x) = a_n * x^n + ... + a_1 * x + a_0$$

Obliczenie tak zdefiniowanej wartości funkcji *WartośćWielomianu* sprowadza się, w oczywisty sposób, do prostego algorytmu obliczeń.

3.8. Operacje na funkcjach

Na funkcjach można wykonywać różne operacje, definiując w ten sposób nowe funkcje. Funkcje są relacjami, w szczególności można wykonywać na nich operacje mno-

gościowe, ale należy zauważyć, że wynikiem takich operacji nie zawsze jest funkcja. Jeżeli na przykład dane są dwie funkcje f, g: $A \to B$, to ich mnogościowa suma $f \cup g$ może nie być funkcją, natomiast przekrój funkcji $f \cap g$ oraz ich różnica $f \setminus g$ są zawsze funkcjami.

Operacja *superpozycji* albo *składania sekwencyjnego* funkcji jest zdefiniowana tak samo jak dla relacji. Jeżeli dane są dwie funkcje:

$$f: A \to B \text{ oraz } g: B \to C$$

to złożeniem sekwencyjnym albo superpozycją funkcji f z funkcją g, oznaczanym przez $f \circ g$, jest funkcja typu $A \to C$, określona następująco:

$$(f \circ g)(a) =_{def} g(f(a))$$

pod warunkiem, że $f(a)$ oraz $g(f(a))$ są określone.

Inne operacje specyficzne dla funkcji to operacje:

- *warunkowego wyboru*,
- *modyfikacji funkcji przez podstawienie*,
- *obcięcia*.

Niech będą dane dwie funkcje $f, g : A \to B$ oraz trzecia funkcja $h : A \to Logiczne$, gdzie $Logiczne =_{def} \{prawda, fałsz\}$. *Warunkowym wyborem* funkcji f, g, h, oznaczanym

$$h \to f, g$$

nazywa się funkcję typu $A \to B$, która jest określona następująco:

$$(h \to f,\ g)(a) = \begin{cases} f(a) \text{ gdy } & h(a) = prawda \\ g(a) \text{ gdy } & h(a) = fałsz \end{cases}$$

Przykład 3.13

Niech

$f = \{<1, 2>, <2, 3>, <3, 4>\}$
$g = \{<1, 3>, <2, 3>, <5, 5>\}$
$h = \{<1, prawda>, <2, fałsz>, <3, prawda>, <4, fałsz>, <5, prawda>\}$

wówczas

$h \to f, g = \{\{<1, 2>, <2, 3>, <3, 4>\}$

Niech $f : A \to B$ będzie funkcją oraz niech $a \in A$, $b \in B$. *Modyfikacją funkcji f przez podstawienie* wartości b dla argumentu o wartości a jest funkcja typu $A \to B$, oznaczana symbolicznie $f[a := b]$, zdefiniowana w sposób następujący:

$$f[a := b](x) =_{def} (x = a) \to b, f(x)$$

W definicji tej wykorzystano poprzednio wprowadzony operator warunkowego wyboru funkcji. Wyrażenie $x = a$ przedstawia funkcję, która przyjmuje wartość logiczną *prawda* wtedy i tylko wtedy, gdy argument x przyjmuje wartość a.

Przykład 3.14

Niech:

$f = \{<1, 2>, <2, 3>, <3, 4>\}$

$g = \{<2, 3>, <5, 5>\}$

wówczas:

$f[1 := 5] = \{<1, 5>, <2, 3>, <3, 4>\}$

$g[1 := 5] = \{<1, 5>, <2, 3>, <5, 5>\}$

Niech $C \subset A$. *Obcięciem* funkcji $f : A \to B$ do podzbioru C zbioru źródłowego A będzie się nazywać funkcję $f|_C : C \to B$, określoną następująco:

dla dowolnego $a \in C : f|_C(a) =_{def} f(a)$.

Łatwo sprawdzić, że równoważną definicją obcięcia funkcji jest

$$f|_C =_{def} f \cap (C \times B)$$

Niech $f : A \to B$ oraz niech $A_1 \subseteq A$, $B_1 \subseteq B$. *Obrazem* zbioru A_1 dla funkcji f nazywa się zbiór

$$f(A_1) =_{def} \{b \in B \mid \exists\, a \in A_1 \bullet f(a) = b\}$$

Przeciwobrazem zbioru B_1 dla funkcji f nazywa się zbiór

$$f^{-1}(B_1) =_{def} \{a \in A \mid \exists\, b \in B_1 \bullet f(a) = b\}$$

Przykład 3.15

Jeżeli

$f = \{<1, 2>, <2, 3>, <3, 4>, <4, 5>\}$

to:

$f|_{\{1, 2\}} = \{<1, 2>, <2, 3>\}$

$f(\{1, 2\}) = \{2, 3\}$

$f^{-1}(\{3, 4\}) = \{2, 3\}$

3.9. Funkcje a relacje

Pomiędzy relacjami a funkcjami zachodzą pewne związki. Każda funkcja jest – z definicji – relacją. Odwrotnie tak nie jest, ale każdej relacji $R \subseteq A \times B$ można przyporządkować przynajmniej jedną taką funkcję $f_R : A \rightarrow B$, że dla każdego $a \in dom(R)$

$$f_R(a) =_{def} b$$

gdzie $b \in B$ jest takim elementem, że $<a, b> \in R$, czyli że

$$f_R \subseteq R \text{ oraz } dom(f_R) = dom(R).$$

Funkcję f_R nazywa się funkcją *zgodną* z relacją R.

Przykład 3.16

Niech

$$R = \{<1, 2>, <1, 3>, <2, 5>, <2, 3>, <3, 4>, <4, 5>\}$$

wówczas:

$$\{<1, 2>, <2, 3>, <3, 4>, <4, 5>\}$$
$$\{<1, 3>, <2, 3>, <3, 4>, <4, 5>\}$$

są funkcjami zgodnymi z R.

Związek zgodności zachodzi pomiędzy programem a jego specyfikacją. Specyfikację programu wyraża się jako pewną relację *Spec*, program zaś jest pewną funkcją *Prog*. Program spełnia specyfikację, gdy pomiędzy specyfikacją *Spec* i programem *Prog* zachodzi związek zgodności, czyli $dom(Prog) = dom(Spec)$ oraz $Prog \subseteq Spec$.

Przykład 3.17

Przypuśćmy, że potrzebna jest funkcja obliczająca pierwiastek kwadratowy z liczby rzeczywistej x, z dokładnością 1. Niech y będzie wartością tej funkcji dla danego x. Związek między x oraz y jest określony zależnością

$$y^2 \leq x \leq (y + 1)^2$$

Formuła ta definiuje relację

$$Spec_{sqrt} =_{def} \{<x, y> \in Rzeczywiste^2 \mid y^2 \leq x \leq (y + 1)^2\}$$

która może być *specyfikacją* programu.

Implementacją dla tej relacji jest dowolna funkcja (realizowana przez program)

$$Impl_{sqrt} : Rzeczywiste \rightarrow Rzeczywiste$$

która spełnia warunki:

$$dom(Impl_{sqrt}) = dom(Spec_{sqrt}) \text{ oraz } Impl_{sqrt} \subseteq Spec_{sqrt}.$$

Czytelnikowi proponuje się samodzielne przedstawienie graficznej ilustracji relacji $Spec_{sqrt}$ oraz funkcji $Impl_{sqrt}$ na płaszczyźnie współrzędnych x, y.

Dowolną relację można przedstawić za pomocą jej funkcji charakterystycznej. Jeżeli dana jest relacja $R \subseteq A_1 \times ... \times A_n$, to jej *funkcją charakterystyczną* jest funkcja

$$f_R : A_1 \times ... \times A_n \rightarrow Logiczne$$

zdefiniowana następująco:

$$f_R(a_1, ..., a_n) = prawda \text{ wtedy i tylko wtedy, gdy } <a_1, ..., a_n> \in R.$$

Funkcja charakterystyczna dla danej relacji R jest wyznaczona jednoznacznie. Odwrotnie – dana funkcja charakterystyczna wyznacza jednoznacznie pewną relację.

3.10. Grafy a relacje

Liczne zastosowania w informatyce mają grafy. Rozpatrzmy najpierw klasę *grafów skierowanych*. Mają one dwie równoważne definicje. W zależności od potrzeb wykorzystuje się jedną z nich.

Pierwsza definicja określa graf skierowany G jako parę

$$G = <V, A>$$

gdzie:

V jest zbiorem *wierzchołków* grafu,
A jest zbiorem *łuków* grafu, określonym jako relacja binarna na zbiorze wierzchołków $A \subseteq V \times V$.

Interpretacja relacji A jest następująca: para $<v_1, v_2> \in A$ reprezentuje łuk grafu prowadzący od wierzchołka v_1 do wierzchołka v_2.

Druga definicja określa graf skierowany G jako parę

$$G = <V, S>$$

gdzie:

V jest zbiorem *wierzchołków* grafu,
S jest funkcją, zwaną *funkcją następników*, określoną na zbiorze wierzchołków, której wartościami są podzbiory wierzchołków S: $V \rightarrow 2^V$.

Interpretacja funkcji S jest następująca: $S(v) = \{v_1, ..., v_k\}$ reprezentuje zbiór wierzchołków-następników wierzchołka v, to znaczy wierzchołków, do których prowadzą łuki wychodzące z wierzchołka v. Jej funkcja odwrotna $P(v)$ określa zbiór wierzchołków-poprzedników wierzchołka v, to znaczy wierzchołków, od których prowadzą łuki do wierzchołka v. Znając dla danego grafu funkcję S, łatwo jest wyznaczyć funkcję P.

Łatwo również zauważyć, że graf zdefiniowany według jednej z tych definicji daje się wyrazić w równoważny sposób według drugiej definicji.

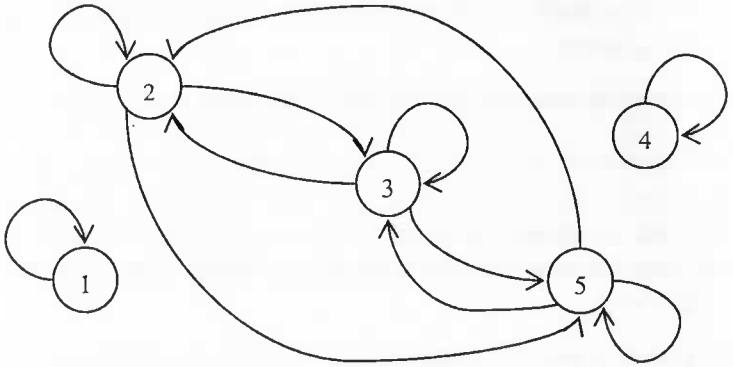
Obie definicje grafów dają podstawę do graficznej ich reprezentacji. Pierwsza definicja pozwala także na graficzną reprezentację relacji binarnych. Pierwszy przykład takiej reprezentacji przedstawiono na rysunku 3.2. Poniżej rozważamy inny przykład, będący graficzną reprezentacją relacji równoważności z przykładu 3.4. W tym przypadku łatwo się przekonać, że analiza niektórych własności relacji, na przykład przechodniości, staje się przejrzysta.

Przykład 3.18

Relacja $R \subseteq A^2$, gdzie $A =_{def} \{1, 2, 3, 4, 5\}$, zdefiniowana następująco:

$$R =_{def} \{<1, 1>, <2, 2>, <3, 3>, <4, 4>, <5, 5>, <3, 2>, <2, 3>, <2, 5>, <5, 2>, <3, 5>, <5, 3>\}$$

ma postać graficzną przedstawioną na rysunku 3.5.



Rys. 3.5. Graficzna reprezentacja relacji

Podgrafem grafu $G = <V, A>$ nazywa się dowolny graf $G' = <V', A'>$ taki, że $V' \subseteq V$ oraz $A' \subseteq V' \times V'$.

Graf nazywa się *nieskończonym*, gdy nieskończony jest zbiór jego wierzchołków.

Ścieżką w grafie $G = <V, A>$ nazywa się niepusty ciąg łuków

$$<v_1, v_2> <v_2, v_3> \ldots <v_{n-1}, v_n> \ldots$$

gdzie $<v_i, v_{i+1}> \in A$, dla $i = 1, \ldots, n-1, \ldots$ Ścieżka przechodzi przez wierzchołki:

$$v_1, v_2, \ldots, v_n \ldots$$

gdzie v_1 jest początkowym wierzchołkiem ścieżki, a jeżeli ścieżka jest skończona, to v_n jest jej wierzchołkiem końcowym. Ścieżkę skończoną, która ma taki sam wierzchołek początkowy i końcowy, nazywa się *cyklem*. Cykl złożony z jednego elementu nazywa się *pętlą*.

Wyróżnia się wiele rodzajów grafów. Określa się między innymi, że graf $G = <V, A>$ jest:

zwrotny, gdy relacja A jest zwrotna (przy każdym wierzchołku jest pętla),

przeciwzwrotny, gdy relacja A jest przeciwzwrotna (graf nie ma pętli),

symetryczny, gdy relacja A jest symetryczna,

przeciwsymetryczny, gdy relacja A jest przeciwsymetryczna,

antysymetryczny, gdy relacja A jest antysymetryczna,

przechodni, gdy relacja A jest przechodnia.

Grafy symetryczne nazywa się także grafami *nieskierowanymi*.

Szczególnym, dalej wykorzystywanym grafem, jest graf nazywany *drzewem*. Jest to graf, który:

- nie ma cykli,
- ma dokładnie jeden wierzchołek v_0, zwany *korzeniem* drzewa, który nie ma poprzedników, to znaczy nie istnieje wierzchołek $v \in V$ taki, że $<v, v_0> \in A$,
- wszystkie pozostałe wierzchołki mają dokładnie jeden poprzednik, to znaczy dla dowolnego wierzchołka $v \neq v_0$ zachodzi $card\{v' \in V \mid <v', v> \in A\} = 1$. Wierzchołek v, który ma następniki, to znaczy $card\{v' \in V \mid <v, v'> \in A\} > 0$, nazywa się wierzchołkiem *rozgałęziającym*. Wierzchołek, który nie ma następników, to znaczy $card\{v' \in V \mid <v, v'> \in A\} = 0$, nazywa się *liściem*, a zbiór wszystkich liści nazywa się *koroną* drzewa.

Lemat 5.1 (*Lemat Königa*)

Jeżeli graf G jest drzewem nieskończonym, w którym każdy wierzchołek ma skończoną liczbę wierzchołków-następników, to w grafie G istnieje ścieżka o nieskończonej długości.

Dowód

Niech v_0 będzie korzeniem grafu G. Zgodnie z założeniem v_0 ma skończoną liczbę wierzchołków-następników. Wśród nich istnieje przynajmniej jeden wierzchołek, niech będzie to v_1, który jest korzeniem nieskończonego poddrzewa G_1 drzewa G, gdyż – w przypadku przeciwnym – gdyby wszystkie wierzchołki-następniki v_0 były korzeniami poddrzew skończonych, to graf G byłby skończony. Powtarzając podobne rozumowanie do następników wierzchołka v_1, znajduje się wśród jego następników wierzchołek v_2, który jest korzeniem nieskończonego podrzewa G_2 drzewa G_1 itd. Jest zatem oczywiste, że ciąg wierzchołków $v_0, v_1, v_2, \ldots$ wyznacza nieskończoną ścieżkę. ■

Ćwiczenia

1. Ile relacji binarnych można zdefiniować na produkcie kartezjańskim $A \times B$, jeżeli A oraz B są zbiorami skończonymi o licznościach $card(A) = n$ oraz $card(B) = m$.

2. Uzupełnij i udowodnij wzory:

   a) $(A \cap B) \times C = (A \times C) \cap (B \times C)$
   b) $(A \cup B) \times C = ?$
   c) $(A \cup B) \times (C \cup D) = ?$

3. Niech $card(A) = n$ oraz $card(B) = m$. Jaka jest liczba funkcji całkowitych oraz częściowych typu $A \to B$?

4. Niech U będzie pewnym zbiorem uniwersum oraz $\subseteq_U$ niech będzie relacją zawierania pomiędzy podzbiorami zbioru U. Które z własności: symetrię, zwrotność, przechodniość ma relacja $\subseteq_U$?

5. Niech $X =_{def} \{a, b, c, d\}$ oraz $R \subseteq X^2$. Zbadać, które spośród własności: symetrii, przeciwsymetrii, zwrotności, przeciwzwrotności, przechodniości, spójności i równoważności mają następujące relacje binarne:

   a) $R = \{<a, a>, <b, b>, <a, b>\}$
   b) $R = \{<a, a>, <b, b>, <c, c>, <d, d>, <a, b>, <b, a>\}$

6. Pomiędzy ludźmi wyróżnia się różne stosunki: koleżeństwo, znajomość, przyjaźń, wrogość, pokrewieństwo itp. Stosunki te można modelować relacjami binarnymi określonymi na zbiorze ludzi, na przykład: $R_{koleżeństwo} =_{def} \{<a, b> \mid a$ jest kolegą $b\}$;

   a) określić, które spośród własności charakteryzujących relacje binarne można przypisać nowo zdefiniowanym relacjom,
   b) jakie związki zachodzą pomiędzy tymi relacjami, chodzi o związki zawierania, na przykład, czy $R_{znajomość} \subseteq R_{koleżeństwo}$, a także o inne, na przykład: czy jeżeli $<a, b> \in R_{wrogość}$ oraz $<b, c> \in R_{wrogość}$, to $<a, c> \in R_{przyjaźń}$?

7. Sprawdzić, czy prawdziwe są następujące stwierdzenia dotyczące relacji binarnych na X:

   a) suma dwóch relacji symetrycznych jest symetryczna,
   b) część wspólna (przekrój) dwu relacji przechodnich jest przechodnia,
   c) jeżeli R jest relacją przechodnią oraz $R \subseteq S \subseteq X^2$, to S jest relacją przechodnią.

8. Sprawdzić, czy prawdziwe są następujące stwierdzenia dotyczące relacji równoważności na X:

   a) suma dwóch relacji równoważności jest relacją równoważności,
   b) przekrój dwóch relacji równoważności jest relacją równoważności.
   c) różnica dwóch relacji równoważności jest relacją równoważności.

9. Niech ID będzie zbiorem identyfikatorów. Czy zdefiniowane poniżej relacje binarne $R_1, R_2 \subseteq ID^2$ są relacjami równoważności? Jeżeli tak, to jakie są wyznaczone przez nie zbiory ilorazowe?

   a) $R_1 =_{def} \{<id_1, id_2> \mid$ pierwsza litera identyfikatora id_1 jest taka sama, jak pierwsza litera identyfikatora $id_2\}$,
   b) $R_2 =_{def} \{<id_1, id_2> \mid$ identyfikator id_1 czytany wspak jest taki sam, jak identyfikator $id_2\}$.

10. Niech $BAZA =_{def} Nazwisko \times Wiek \times Zarobek$, gdzie *Nazwisko* jest zbiorem identyfikatorów, *Wiek* i *Zarobek* są pewnymi podzbiorami nieujemnych liczb całkowitych. Czy relacje binarne $R_1, R_2 \subseteq BAZA^2$ są relacjami równoważności? Jeżeli tak, to jakie są wyznaczone przez nie zbiory ilorazowe?

    a) $R_1 =_{def} \{<<n_1, w_1, z_1>, <n_2, w_2, z_2>> \mid w_1 = w_2 \wedge z_1 = z_2\}$,
    b) $R_2 =_{def} \{<<n_1, w_1, z_1>, <n_2, w_2, z_2>> \mid w_1 = w_2 \wedge |z_1 - z_2| < 1000\}$.

11. Ile jest różnych relacji równoważności na zbiorze n-elementowym?

12. Niech $R, S \subseteq X^2$ będą relacjami równoważności. Czy relacjami równoważności są również:

    a) $R \cup S$,
    b) $R \cap S$,
    c) $R \setminus S$,
    d) $R \circ S$.

13. Niech $R \subseteq X^2$ będzie relacją równoważności oraz $x, y \in X$ będą dwoma ustalonymi elementami zbioru X. Czy relacją równoważności jest relacja:

    $S =_{def} (R \cup \{<x, y>, <y, x>\})^+$

14. Jeśli $R_1 \subseteq X^2$, to $R_2 \subseteq X^2$ takie, że $R_1 \subseteq R_2$ nazywamy rozszerzeniem relacji R_1. Czy każdą relację $R \subseteq X^2$ można rozszerzyć do relacji:

    a) symetrycznej,
    b) przeciwsymetrycznej,
    c) zwrotnej,
    d) przeciwzwrotnej,
    e) przechodniej,
    f) spójnej.

15. Wykazać, że relacja R jest przechodnia wtedy i tylko wtedy, gdy spełniony jest warunek

    $R^2 \subseteq R$

16. Niech S, T będą relacjami binarnymi na X^2. Wskaż, które własności są prawdziwe:

    a) $dom(S \cup T) = dom(S) \cup dom(T)$,

b) $dom(S \cup T) \subseteq dom(S) \cup dom(T)$,
c) $dom(S \cap T) \subseteq dom(S) \cap dom(T)$.

17. Pokazać, że złożenie funkcji różnowartościowych jest funkcją różnowartościową.

18. Niech $f: X \to Y$ oraz $A, B \subseteq X$. Uzupełnij i udowodnij wzory:
a) $f(A \cup B) = f(A) \cup f(B)$
b) $f(A \cap B)$? $f(A) \cap f(B)$
c) $f^{-1}(f(A))$? A

19. Funkcja f jest *zgodna* z relacją i wtedy i tylko wtedy, gdy $f \subseteq R$. Niech $X =_{def} \{a, b, c, d\}$ oraz relacja R będzie zdefiniowana następująco:

$R =_{def} \{<a, b>, <a, d>, <c, c>, <b, b>, <b, d>, <c, c>, <d, b>, <c, d>, <d, c>, <d, a>, <d, d>\}$.

Zdefiniować wszystkie funkcje f zgodne z relacją R takie, że:

a) $dom(f) = dom(R)$,
b) $ran(f) = ran(R)$.

Które spośród tych funkcji mają funkcje odwrotne?

20. Podać warunki konieczne i wystarczające na to, aby mnogościowa suma dwóch funkcji była funkcją.

21. Niech R będzie dowolną relacją równoważności na zbiorze X oraz niech relacja Q będzie zdefiniowana następująco:

$Q =_{def} (R \cup \{<a, b>, <b, a>\})^+$,

gdzie $a, b \in X$. Pokazać, że Q jest relacją równoważności oraz jeśli $<a, b> \notin R$, to

$X/Q = (X/R \setminus \{[a]_R, [b]_R\}) \cup \{[a]_R \cup [b]_R\}$.

22. Niech będzie dany pewien graf $G = <V, A>$, gdzie $A \subseteq V^2$. Jaką interpretację można przypisać złożeniu relacji A? Co oznaczają A^2, ..., A^n? W jaki sposób można zbadać, czy graf ma pętle oraz cykle, to jest drogi o długości większej od 1, które rozpoczynają się i kończą się w tym samym wierzchołku?

23. Pokazać, w jaki sposób na podstawie definicji grafu w postaci $G = <V, A>$ zbudować jego definicję o postaci $G = <V, S>$, gdzie $S: V \to 2^V$ jest funkcją wyznaczającą dla dowolnego wierzchołka $v \in V$ zbiór wierzchołków-następników $S(v)$, to znaczy wierzchołków, do których prowadzą łuki z wierzchołka v.

24. Pokazać, w jaki sposób na podstawie definicji grafu w postaci $G = <V, S>$ wyznaczyć funkcję $P(v)$, określającą zbiór wierzchołków-poprzedników wierzchołka v.

25. Zaproponować sposoby reprezentacji grafu w pamięci komputera.

4. Aksjomatyczna i alternatywne teorie zbiorów

4.1. Aksjomatyczne ujęcie teorii mnogości

Używane dotychczas pojęcie zbioru jest rozumiane na jeden z dwóch sposobów, które spotyka się w rozumieniu potocznym. Jest to tak zwane *dystrybutywne* rozumienie zbioru, czyli zespołu (zestawu) wielu przedmiotów (bytów) połączonych w całość ze względu na pewne wspólne własności. Dystrybutywne rozumienie zbioru wiąże się ze stosunkiem pomiędzy elementem a zbiorem: *element należy do zbioru*. Synonimami tak rozumianego zbioru są spotykane w języku naturalnym takie określenia, jak: 'klasa', 'kolekcja', 'wielość', 'mnogość', 'zbiorowość', 'gatunek', 'rodzaj'.

Drugie, nieużywane tu, pojęcie zbioru, tak zwane *kolektywne*, jest rozumiane jako całość złożona z pewnych przedmiotów, które stanowią części całości. Na przykład las może być traktowany jako całość, której częściami są drzewa, krzewy, runo leśne itp. Kolektywne rozumienie zbioru stosunek pomiędzy elementem a zbiorem określa, że: *element jest częścią zbioru*, dlatego synonimami zbioru w sensie kolektywnym są na przykład określenia: 'całość', 'agregat', 'kompleks', 'konglomerat'.

Można twierdzić, że samochód jest zbiorem w sensie kolektywnym, którego częściami są podwozie, koła, silnik, nadwozie itd. Części te są komponentami zbioru i jednocześnie są również zbiorami w sensie kolektywnym, gdyż składają się z innych, mniejszych komponentów. Zbiory kolektywne są zbiorami tranzytywnymi, ponieważ stosunek 'bycia częścią" jest relacją przechodnią.

Zbiory w sensie kolektywnym są wygodne do przedstawiania wielu pojęć w naukach przyrodniczych, społecznych, a także w lingwistyce komputerowej. Rozważania w książce odnoszą się wyłącznie do zbiorów w sensie dystrybutywnym.

Oprócz rozróżnienia zbiorów w sensie kolektywnym i dystrybutywnym, spotyka się jeszcze inne podejścia do pojęcia zbioru, określane jako podejścia alternatywne. Wśród takich podejść w dalszej części rozdziału omawia się bardzo ogólnie wielozbiory, zbiory rozmyte i zbiory przybliżone, które mają liczne zastosowania w informatyce.

W początkowym okresie swego rozwoju teoria mnogości (teoria zbiorów w sensie dystrybutywnym) była budowana na podstawie intuicyjnego pojęcia zbioru. Droga ta okazała się zawodna, gdyż intuicja nie dawała jednoznacznych odpowiedzi na pewne

subtelne pytania. W konsekwencji pojawiły się sprzeczności, jak na przykład omówiona wcześniej antynomia Russella. W celu ich eliminacji zbudowano różne aksjomatyczne teorie zbiorów. Poniżej przedstawiamy zestawy aksjomatów opracowane przez Zermela[7]. Zestaw ten jest wystarczający do praktyki matematycznej, zwłaszcza do definiowania liczb naturalnych, całkowitych, wymiernych i rzeczywistych ze zwykłymi działaniami arytmetycznymi. Bardziej rozpowszechniona jest nieco silniejsza teoria, zwana teorią Zermela–Fraenkla[8]. Aksjomaty Zermela są tu przedstawiane za pomocą języka naturalnego.

1. *Aksjomat ekstensjonalności*

Dwa zbiory są równe wtedy i tylko wtedy, gdy mają te same elementy.

2. *Aksjomat wyróżniania*

Dla dowolnego zbioru Z i dowolnego jednoargumentowego predykatu (funkcji zdaniowej) P istnieje zbiór T zawierający dokładnie te elementy Z, które spełniają warunek $P(x)$.
Jeżeli żaden element Z nie spełnia predykatu P, na przykład, gdy $P(x)$ jest warunkiem postaci $x \notin Z$, to T jest zbiorem pustym $\varnothing$. Aksjomat wyróżniania zapewnia więc istnienie zbioru pustego $\varnothing$.

3. *Aksjomat par nieuporządkowanych*

Jeżeli Z_1, Z_2 są zbiorami, to para nieuporządkowana $\{Z_1, Z_2\}$ jest zbiorem.

4. *Aksjomat sumy zbiorów*

Niech Z będzie niepustą rodziną zbiorów, tj. zbiorem, którego elementy są zbiorami. Dla każdej takiej rodziny istnieje zbiór S, którego elementami są dokładnie te obiekty, które są elementami zbiorów należących do Z.

5. *Aksjomat nieskończoności*

Istnieje zbiór Z, który zawiera zbiór pusty i jest taki, że jeżeli x należy do Z, to suma x oraz $\{x\}$ także jest w Z.
Rozróżnienie między elementem x a zbiorem jednoelementowym $\{x\}$ ma zasadnicze znaczenie. Aksjomat gwarantuje istnienie zbiorów nieskończonych.

6. *Aksjomat zastępowania*

Niech dla każdego x istnieje dokładnie jedno y takie, że spełniony jest dwuargumentowy predykat (funkcja zdaniowa) $P(x, y)$, wtedy dla każdego zbioru Z istnieje zbiór Z', do którego należą wszystkie i tylko te elementy y, które przy pewnym x ze zbioru Z, spełniają predykat P.

[7] Ernst Zermelo (1871–1953).

[8] Abraham Fraenkel (1891–1965).

7. *Aksjomat zbioru potęgowego*

Dla każdego zbioru Z istnieje rodzina zbiorów, której elementami są wszystkie podzbiory zbioru Z. Rodzinę tę nazywa się zbiorem potęgowym i oznacza 2^Z.

8. *Aksjomat wyboru*

Dla dowolnej rodziny niepustych i rozłącznych zbiorów istnieje zbiór, który z każdym ze zbiorów tej rodziny ma jeden i tylko jeden wspólny element.
Aksjomat wyboru jest z jednej strony intuicyjnie oczywisty, ale z drugiej strony budzi różne kontrowersje. Ich zasadniczym powodem jest to, że w przypadku nieprzeliczalnej rodziny zbiorów nie wiadomo, w jaki sposób tworzyć nowy zbiór, który miałby dokładnie jeden element wspólny z każdym zbiorem tej rodziny. Z całą pewnością proces tworzenia takiego zbioru nie mógłby być postępowaniem efektywnym, to znaczy opartym na realizacji pewnego algorytmu.

9. *Aksjomat regularności (ufundowania)*

W każdym niepustym zbiorze Z istnieje taki element X, że żaden element zbioru X nie jest elementem zbioru Z.
Konsekwencją aksjomatu jest to, że nie istnieją zbiory X, Y, Z o takich własnościach, jak na przykład, że $X \in X$, że zachodzi $X \in Y$ oraz $Y \in X$, że zachodzi $X \in Y$, $Y \in Z$, $Z \in X$ itd. Aksjomat ten ogranicza dziedzinę złożoną ze zbiorów przez wyeliminowanie z niej obiektów o własnościach w rodzaju wyżej wymienionych.

4.2. Definicje zbiorów liczbowych

Na gruncie aksjomatycznego ujęcia teorii mnogości można formalnie zdefiniować liczby naturalne. Ponieważ jedynymi obiektami, których istnienie gwarantuje teoria mnogości, są zbiory, więc liczby naturalne także definiuje się jako szczególne rodzaje zbiorów.

Dla dowolnego zbioru Z jego *następnikiem* nazwa się zbiór

$$Succ(Z) =_{def} Z \cup \{Z\}$$

Zachodzi więc $Z \subseteq Succ(Z)$ oraz $Z \in Succ(Z)$.

Punktem wyjścia w konstrukcji zbioru liczb naturalnych jest przyjęcie istnienia zbioru pustego. Zbiór liczb naturalnych definiuje się jako najmniejszy zbiór *Nat*, definiowany rekursywnie w sposób następujący:

1. $\varnothing \in Nat$
2. jeżeli $Z \in Nat$, to $Succ(Z) \in Nat$

Elementy zbioru *Nat* nazywa się liczbami naturalnymi i są nimi:

$\varnothing$,

$\varnothing \cup \{\varnothing\} = \{\varnothing\}$,

$\{\varnothing\} \cup \{\{\varnothing\}\} = \{\varnothing, \{\varnothing\}\}$,

$\{\varnothing, \{\varnothing\}\} \cup \{\{\varnothing, \{\varnothing\}\}\} = \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}$ itd.

Zbiór liczb naturalnych jest więc rodziną zbiorów. W celu uproszczenia notacji elementów tej rodziny wprowadza się powszechnie znane oznaczenia:

$0 =_{def} \varnothing$,

$1 =_{def} \{\varnothing\} = \{0\}$

$2 =_{def} \{\varnothing, \{\varnothing\}\} = \{0, 1\}$

$3 =_{def} \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\} = \{0, 1, 2\}$

.....

$n =_{def} \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}, ..., \{\varnothing, \{\varnothing\}, ..., \{...\{\varnothing\}...\} ...\} = \{0, 1, 2, ..., n-1\}$

które są znacznie wygodniejsze w użyciu.

Uwaga

Zbiór liczb naturalnych jest przykładem tak zwanej induktywnej rodziny zbiorów. Rodzina zbiorów $\boldsymbol{A}$ jest *induktywna*, gdy spełnia warunki:

1. $\varnothing \in \boldsymbol{A}$,
2. dla każdego zbioru $X \in \boldsymbol{A}$, również zbiór $X \cup \{X\} \in \boldsymbol{A}$.

Operacja, która zbiorowi X przyporządkowuje $X \cup \{X\}$, nazywa się operacją *następnika*. Istnienie zbiorów induktywnych jest postulowane aksjomatem nieskończoności. Zbiór liczb naturalnych *Nat* jest najmniejszym zbiorem induktywnym, to znaczy dla każdego zbioru induktywnego $\boldsymbol{A}$ zachodzi $Nat \subseteq \boldsymbol{A}$.

Możliwe są także inne mnogościowe sposoby definiowania liczb naturalnych, na przykład:

$0 =_{def} \varnothing$,

$1 =_{def} \{\varnothing\} = \{0\}$

$2 =_{def} \{\{\varnothing\}\} = \{1\}$

$3 =_{def} \{\{\{\varnothing\}\}\} = \{2\}$

.....

$n =_{def} \{\{...\{\varnothing\}...\}\} = \{n-1\}$

Traktując liczby naturalne jako induktywnie zdefiniowane zbiory, można pokazać następujące twierdzenie.

Twierdzenie 4.1

Dla dowolnych liczb $m, n \in Nat$ zachodzą związki:

1. Jeśli $m \in n$, to $m \subseteq n$.
2. $n \notin n$.
3. Jeśli $Succ(m) = Succ(n)$, to $m = n$.
4. Jeśli $m \subseteq n$ oraz $m \neq n$, to $m \in n$.
5. Zachodzi $m \subseteq n$ lub $n \subseteq m$.
6. Zachodzi dokładnie jedna z możliwości: $m \in n$, $n = m$, $m \in n$.

Dowód twierdzenia można znaleźć na przykład w książce [Tiuryn 2003].

Przy wprowadzonych oznaczeniach operację tworzenia nowego zbioru *Succ* można traktować też jako funkcję dodawania jedynki do danej liczby naturalnej. Jest to funkcja całkowicie określona o sygnaturze $Succ : Nat \rightarrow Nat$. Nazywa się ją *operacją następnika* i można pisać:

$Succ(0) = 1$

$Succ(Succ(0)) = Succ(1) = 2$

$Succ(Succ(Succ(0))) = Succ(Succ(1)) = Succ(2) = 3$

Operacja następnika pozwala na zdefiniowanie innych operacji (działań). Na przykład *dodawanie* oraz mnożenie są funkcjami o sygnaturze:

$\_+\_ : Nat \times Nat \rightarrow Nat$

$\_*\_ : Nat \times Nat \rightarrow Nat$

Dodawanie można zdefiniować rekursywnie:

$m + 0 = m$	dla dowolnego $m \in Nat$
$m + Succ(n) = Succ(m + n)$	dla dowolnych $m, n \in Nat$

Dysponując dodawaniem również rekursywnie można określić *mnożenie*:

$m * 0 = 0$	dla dowolnego $m \in Nat$
$Succ(m) * n = m * n + n$	dla dowolnych $m, n \in Nat$

Mając liczby naturalne, można zdefiniować inne rodzaje liczb: liczby całkowite, wymierne, rzeczywiste i zespolone.

Definicję liczb całkowitych poprzedza się pewnym wyjaśnieniem intuicyjnym. Każdej liczbie całkowitej przypisuje się parę liczb naturalnych $<m, n>$ takich, że różnica $m - n$ jest równa tej liczbie całkowitej. Na przykład liczbie całkowitej –2 może być przyporządkowana para $<4, 6>$, liczbie 0 – para $<10, 10>$, a liczbie 3 – para $<4, 1>$. Dwie pary $<m_1, n_1>$ oraz $<m_2, n_2>$, które spełniają warunek

$m_1 - n_1 = m_2 - n_2$

reprezentują tę samą liczbę całkowitą. Ponieważ różnica dwóch liczb naturalnych nie zawsze jest liczbą naturalną, dlatego zamiast takiego warunku można sformułować inny warunek równoważny, w którym nie odwołuje się do różnicy. Jest to warunek postaci

$$m_1 + n_2 = m_2 + n_1$$

Przyjmuje się teraz następującą definicję relacji binarnej $R \subseteq Nat^2 \times Nat^2$, określonej na parach liczb naturalnych w sposób następujący:

$$R =_{def} \{<<m_1, n_2>, <m_1, n_2>> \mid m_1 + n_2 = m_2 + n_1\}$$

Łatwo sprawdzić, że R jest relacją równoważności na Nat^2. Zbiór liczb całkowitych jest określony jako zbiór ilorazowy Nat^2/R, czyli

$$Całkowite = Nat^2/R$$

Konstrukcja liczb wymiernych opiera się na założeniu, że każdej liczbie wymiernej można przyporządkować parę $<l, m>$, gdzie $l \in Całkowite$ oraz $m \in Nat\backslash\{0\}$. Dalsza część konstrukcji jest podobna do konstrukcji zbioru liczb całkowitych. Definiuje się mianowicie relację $Q \subseteq (Całkowite \times Nat)^2$ w sposób następujący:

$$Q =_{def} \{<<l_1, m_2>, <l_1, m_2>> \mid l_1 * m_2 = l_2 * m_1\}$$

Q jest relacją równoważności na $Całkowite \times Nat$. Zbiór liczb wymiernych jest określony jako zbiór ilorazowy $(Całkowite \times Nat)/Q$, czyli

$$Wymierne = (Całkowite \times Nat)/Q$$

Definicja zbioru liczb rzeczywistych jest bardziej złożona i dlatego jest tu pomijana.

4.3. Wielozbiory

Uogólnieniem pojęcia zbioru jest pojęcie wielozbioru. Zbiór jest określony jako kolekcja dobrze wyróżnionych obiektów – elementów zbioru. Czasem nie ma potrzeby jednoznacznego rozróżniania pomiędzy elementami zbioru. Tak jest wtedy, gdy elementami zbioru jest wiele kopii tego samego rodzaju obiektów. Jeżeli na przykład rozważa się zbiór, którego elementami są różne owoce – jabłka, gruszki, śliwki, to może nas interesować tylko liczba poszczególnych rodzajów owoców, bez rozróżniania konkretnych owoców.

Definicja 4.1

Jeżeli A jest dowolnym zbiorem, to *wielozbiorem* (albo *multizbiorem*) W nad zbiorem A jest para

$$W =_{def} <A, f>$$

gdzie f jest *funkcją liczności* wielozbioru. Funkcja f jest dowolną, całkowicie określoną na A, o wartościach w zbiorze liczb naturalnych, czyli jest funkcją o sygnaturze $f: A \rightarrow Nat$ oraz dziedzinie $dom(f) = A$.

Jeżeli $a \in A$, to $f(a)$ jest liczbą elementów a w danym wielozbiorze W. Wielozbiór W jest pusty, gdy $f(a) = 0$ dla każdego $a \in A$.

Niech będą dane dwa wielozbiory nad zbiorem A:

$$W_1 = <A, f> \text{ oraz } W_2 = <A, g>$$

W_1 jest podwielozbiorem W_2, co oznacza się $W_1 \subseteq W_2$, jeżeli $f(a) \leq g(a)$, dla każdego $a \in A$.

Wielozbiory W_1 oraz W_2 są identyczne, co pisze się $W_1 = W_2$, wtedy i tylko wtedy, gdy $W_1 \subseteq W_2$ oraz $W_2 \subseteq W_1$.

Na wielozbiorach definiuje się operacje mnogościowe sumy, przekroju i różnicy.

Suma wielozbiorów jest zdefiniowana następująco:

$$W_1 \cup W_2 = <A, h>$$

gdzie h jest funkcją liczności, spełniającą warunek

$h(a) = f(a) + g(a)$ dla dowolnego $a \in A$

Przekrój wielozbiorów jest zdefiniowany następująco:

$$W_1 \cap W_2 = <A, h>$$

gdzie h jest funkcją liczności, spełniającą warunek

$h(a) = min(f(a), g(a))$ dla dowolnego $a \in A$

Różnica wielozbiorów jest zdefiniowana następująco:

$$W_1 \backslash W_2 = <A, h>$$

gdzie h jest funkcją liczności, spełniającą warunek

$h(a) = max(f(a) - g(a), 0)$ dla dowolnego $a \in A$

min oraz *max* są funkcjami, które wyliczają odpowiednio większą oraz mniejszą liczbę spośród dwóch liczb, które są jej argumentami.

Przykład 4.1

Niech $W_1 = <A, f_1>$ oraz $W_2 = <A, f_2>$, gdzie $A = \{a, b, c, d, e\}$ oraz funkcje liczności są zdefiniowane następująco:

$f_1 = \{<a, 4>, <b, 3>, <c, 2>, <d, 1>, <e, 0>\}$
$f_2 = \{<a, 0>, <b, 1>, <c, 2>, <d, 3>, <e, 4>\}$

wówczas

$W_1 \cup W_2 = <A, \{<a, 4>, <b, 4>, <c, 4>, <d, 4>, <e, 4>\}>$

$W_1 \cap W_2 = <A, \{<a, 0>, <b, 1>, <c, 2>, <d, 1>, <e, 0>\}>$

$W_1 \setminus W_2 = <A, \{<a, 4>, <b, 2>, <c, 0>, <d, 0>, <e, 0>\}>$

Uwaga

Często, gdy rozważa się rodzinę wielozbiorów $W_i = <A, f_i>$, dla $i \in I$, nad ustalonym zbiorem A, przyjmuje się uproszczoną notację – wielozbiór W_i utożsamia się z funkcją liczności f_i. Wtedy, zamiast pisać na przykład $W_1 \cup W_2$, pisze się $f_1 \cup f_2$.

4.4. Zbiory rozmyte

Zbiory rozmyte są uogólnieniem zbiorów, którym można się posługiwać w sytuacjach określonych nieprecyzyjnie lub niejednoznacznie. Takie sytuacje występują na przykład wtedy, gdy mówi się *wysoki mężczyzna*, *duże miasto* lub *drogi samochód*. Gdy mówi się o kimś, że jest wysokim mężczyzną, wyraża się przekonanie o stopniu przynależności danego mężczyzny do zbioru wysokich mężczyzn.

Definicja 4.2

Jeżeli A jest dowolnym zbiorem, to *zbiorem rozmytym* Z nad zbiorem A jest para

$Z =_{def} <A, \mu>$

gdzie μ jest *funkcją przynależności* do zbioru rozmytego. Funkcja μ jest dowolną funkcją całkowicie określoną na A, o wartościach w zbiorze liczb rzeczywistych z przedziału [0, 1], czyli funkcją o sygnaturze $\mu : A \to [0, 1]$ oraz $dom(f) = A$.

Jeżeli $a \in A$, to $\mu(a)$ określa stopień przynależności elementu a do danego zbioru rozmytego. Dla $a \in A$ wartość funkcji $\mu(a) = 0$ oznacza brak przynależności, $\mu(a) = 1$ oznacza pełną przynależność, $0 < \mu(a) < 1$ oznacza zaś częściową przynależność elementu a do zbioru rozmytego.

Zbiór rozmyty Z jest pusty, gdy $\mu(a) = 0$ dla każdego $a \in A$.

Przykład 4.2

Przyjmując, że za wysokich mężczyzn można uważać tych, którzy mają co najmniej 170 cm wzrostu, rozmyty zbiór wysokich mężczyzn *WysocyMężczyźni* można zdefiniować następująco:

$WysocyMężczyźni = <Wzrost, \mu_{wzrost}>$

gdzie

$Wzrost = \{x \in Rzeczywiste \mid x \geq 100\}$

$$\mu_{wysoki}(x) = \begin{cases} 0 & \text{dla } x < 170 \\ \frac{1}{15} * (x - 170) & \text{dla } 170 \leq x \leq 185 \\ 1 & \text{dla } x > 185 \end{cases}$$

Niech będą dane dwa zbiory rozmyte $Z_i = <A, \mu_i>$ dla $i = 1, 2$.

Z_1 jest podzbiorem Z_2, co oznacza się $Z_1 \subseteq Z_2$, jeżeli $\mu_1(a) \leq \mu_2(a)$, dla każdego $a \in A$.

Zbiory rozmyte Z_1 oraz Z_2 są identyczne, co pisze się $Z_1 = Z_2$, wtedy i tylko wtedy, gdy $W_1 \subseteq W_2$ oraz $W_2 \subseteq W_1$.

Operacje mnogościowe na zbiorach rozmytych są definiowane następująco:

Suma zbiorów rozmytych

$Z_1 \cup Z_2 = <A, \mu>$

gdzie μ jest funkcją przynależności, spełniającą warunek

$\mu(a) = max(\mu_1(a), \mu_2(a))$ dla dowolnego $a \in A$

Przekrój zbiorów rozmytych

$Z_1 \cap Z_2 = <A, \mu>$

gdzie μ jest funkcją przynależności, spełniającą warunek:

$\mu(a) = min(\mu_1(a), \mu_2(a))$ dla dowolnego $a \in A$

Różnica zbiorów rozmytych

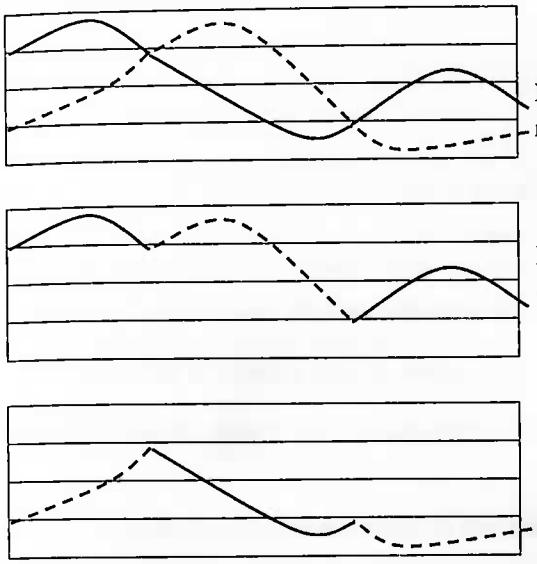
$Z_1 \setminus Z_2 = <A, \mu>$

gdzie μ jest funkcją przynależności, spełniającą warunek:

$\mu(a) = max(\mu_1(a) - \mu_2(a), 0)$ dla dowolnego $a \in A$

Przykład 4.3

Zbiory rozmyte można przedstawić graficznie za pomocą odpowiadających im wykresów funkcji przynależności. Na pierwszym z rysunków przedstawiono wykresy funkcji przynależności dwóch zbiorów rozmytych nad zbiorem liczb rzeczywistych – linia ciągła (zbiór pierwszy) i przerywana (zbiór drugi), a na następnych – wykresy funkcji przynależności ich sumy i przekroju.



Rys. 4.1. Graficzna ilustracja operacji na zbiorach rozmytych

Uwaga

Badania nad zbiorami rozmytymi zainicjował swoimi pracami Lofti Zadeh w połowie lat sześćdziesiątych ubiegłego wieku. Podane wyżej definicje zawierania i równości zbiorów rozmytych oraz operacje mnogościowe są tymi, które spotyka się najczęściej. W literaturze istnieje rozmaitość innych definicji tych pojęć: [Kacprzyk 1986], [Rutkowska, Piliński, Rutkowski 1997].

Z porównania podanych określeń zbiorów rozmytych z wcześniejszymi określeniami dla wielozbiorów wynika, że z obliczeniowego punktu widzenia różnice są mało istotne. Istotne są natomiast różnice interpretacyjne, gdyż funkcja liczności dla danego elementu $a \in A$ określa liczbę kopii samego elementu w wielozbiorze, podczas gdy funkcja przynależności określa stopień przynależności tego elementu do zbioru rozmytego.

4.5. Zbiory przybliżone

Pojęcie zbiorów przybliżonych wywodzi się z zagadnienia klasyfikacji.

Niech U będzie dowolnym zbiorem uniwersum oraz niech $R \subseteq U^2$ będzie pewną relacją równoważności określoną na U. Relacja równoważności wyznacza podział zbioru U na klasy abstrakcji. Przypomnijmy: dwie klasy abstrakcji są identyczne albo rozłączne, a suma mnogościowa wszystkich klas abstrakcji jest równa zbiorowi U. Zbiór wszystkich klas abstrakcji, oznaczany U/R, jest nazywany zbiorem ilorazowym zbioru U.

Niech będzie dany pewien podzbiór $X \subseteq U$. Podzbiór X jest oczywiście określony przez swoje elementy, ale można go też scharakteryzować tylko poprzez elementy zbioru ilorazowego U/R. Charakteryzacja polega na wprowadzeniu dwóch podzbiorów stanowiących dolne i górne przybliżenie zbioru X.

Dolnym przybliżeniem zbioru X względem relacji R, oznaczanym $\underline{R}X$, jest zbiór zdefiniowany następująco:

$$\underline{R}X = \bigcup \{Y \in U/R \mid Y \subseteq X\}$$

Górnym przybliżeniem zbioru X względem relacji R, oznaczanym $\overline{R}X$, jest zbiór zdefiniowany następująco:

$$\overline{R}X = \bigcup \{Y \in U/R \mid Y \cap X \neq \emptyset\}$$

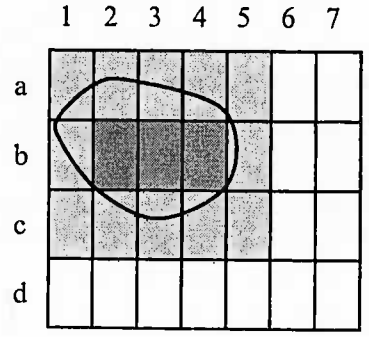
Z definicji wynika, że $\underline{R}X \subseteq X \subseteq \overline{R}X$.

Definicja 4.3

Zbiór X nazywa się *zbiorem przybliżonym* względem relacji R, gdy $\underline{R}X \neq \overline{R}X$. W przeciwnym przypadku, gdy $\underline{R}X = \overline{R}X$, zbiór X nazywa się zbiorem dokładnym względem R.

Przykład 4.4

Ilustracją wprowadzonych pojęć jest rysunek 4.2.



Rys. 4.2. Graficzna ilustracja zbioru przybliżonego

Zbiorem U jest prostokątny obszar na płaszczyźnie X–Y, podzielony na mniejsze prostokąty – kratki są jednoznacznie identyfikowane przez numery wierszy i kolumn. Kratki te są elementami pewnego zbioru ilorazowego dzielącego zbiór U. Zbiór X jest zaznaczony pogrubioną linią. Jego dolnym przybliżeniem $\underline{R}X$ jest ob-

szar zaznaczony trzema mocniej zacieniowanymi kratkami, a górnym jego przybliżeniem $\overline{R}X$ jest obszar zaznaczony wszystkimi zacieniowanymi kratkami.

$\underline{R}X = \{<b, 2>, <b, 3>, <b, 4>\}$

$\overline{R}X = \{<a, 1>, <a, 2>, <a, 3>, <a, 4>, <a, 5>,$
$<b, 1>, <b, 2>, <b, 3>, <b, 4> <b, 5>,$
$<c, 1>, <c, 2>, <c, 3>, <c, 4> <c, 5>\}$

W praktycznych zastosowaniach liczności elementów zbiorów $\underline{R}X$ oraz $\overline{R}X$ dają podstawę do liczbowej oceny dokładności przybliżenia zbioru X. Jeżeli zbiory $\underline{R}X$ oraz $\overline{R}X$ są skończone, to tak zwana miara dokładności przybliżenia zbioru X jest określana jako

$$\alpha_R(X) = card(\underline{R}X) / card(\overline{R}X)$$

Oczywiście $0 \leq \alpha_R(X) \leq 1$.

Uwaga

Zbiory przybliżone zostały wprowadzone przez Zdzisława Pawlaka (1926–2006) w połowie lat osiemdziesiątych ubiegłego wieku. Znalazły one powszechne zastosowanie w informatyce, między innymi w analizie danych, przybliżonej klasyfikacji i przetwarzaniu obrazów [Pawlak 1991].

Ćwiczenia

1. Niech X, Y, Z będą wielozbiorami. Pokazać, że jeżeli $X \subseteq Y$ oraz $Y \subseteq Z$, to $X \subseteq Z$.
2. Niech $W = <U, F>$ będzie wielozbiorem nad zbiorem U, gdzie funkcja liczności F jest zdefiniowana następująco: $F(x) = n$ dla każdego $x \in U$. Jeżeli A jest podwielozbiorem wielozbioru W, to przez A' oznaczamy operację dopełnienia wielozbioru A, którą definiujemy jako: $A' =_{def} W \setminus A$. Sprawdzić, czy zachodzą prawa de Morgana:

   $(A \cap B)' = A' \cup B'$

   $(A \cup B)' = A' \cap B'$
3. Niech X, Y, Z będą zbiorami rozmytymi. Pokazać, że jeżeli $X \subseteq Y$ oraz $Y \subseteq Z$, to $X \subseteq Z$.
4. Niech $Z = <U, \mu>$ będzie zbiorem rozmytym, gdzie funkcja przynależności μ jest zdefiniowana następująco: $\mu(x) = 1$ dla każdego $x \in U$. Jeżeli A jest podzbiorem rozmytym zbioru Z, to przez A' oznaczamy operację dopełnienia zbioru A, którą definiujemy jako: $A' =_{def} Z \setminus A$. Sprawdzić, czy zachodzą prawa de Morgana:

   $(A \cap B)' = A' \cup B'$

   $(A \cup B)' = A' \cap B'$
5. Niech dany będzie zbiór *Osoba* = *Nazwisko* × *Wiek* × *Zarobek*, gdzie *Nazwisko* jest zbiorem nazwisk, *Wiek* = *Nat*, *Zarobek* = *Nat*, oraz niech będą dane dwie relacje równoważności: relacja W na zbiorze *Wiek* i relacja Z na zbiorze *Zarobek*. Uzasadnić, że rodzina zbiorów $\{b_{w,z} \subseteq Osoba \mid w \in Wiek, z \in Zarobek\}$, zdefiniowanych następująco: $b_{w,z} =_{def} \{<o_1, w_1, z_1> \in Osoba \mid w_1 \in [w]_{Wiek} \wedge z_1 \in [z]_{Zarobek}\}$, dla $w \in Wiek$ oraz $z \in Zarobek$, stanowi partycję na zbiorze *Osoba*, czyli wyznacza pewną relację równoważności $R_{W,Z}$ na tym zbiorze. Podać przykłady relacji $W \subseteq Wiek$ i $Z \subseteq Zarobek$. Podać dolne i górne ograniczenia dla przykładowego podzbioru $B \subseteq Osoba$ względem relacji $R_{W,Z}$.

5. Równoliczność zbiorów, liczby kardynalne

5.1. Zbiory przeliczalne

Pojęcie funkcji pozwala na porównywanie liczności zbiorów.

Definicja 5.1

Dwa zbiory A i B są *równoliczne*, co notujemy w postaci $A \sim B$, wtedy i tylko wtedy, gdy istnieje wzajemnie jednoznaczna funkcja (bijekcja)

$$f: A \to B$$

O zbiorach równolicznych mówi się też, że są zbiorami o tej samej *mocy*.

Łatwo zauważyć, że związek równoliczności ma oczywiste własności:

- $A \sim A$,
- jeżeli $A \sim B$, to $B \sim A$,
- jeżeli $A \sim B$ i $B \sim C$, to $A \sim C$.

Uwaga

Związek równoliczności ma własności relacji równoważności. Nie mówi się jednak o relacji równoliczności, gdyż wymagałoby to określenia zbioru, na którym relacja ta jest określona. W tym przypadku dziedziną tą powinien być zbiór, którego elementami są wszystkie możliwe zbiory. Określenie tego, czym jest zbiór wszystkich zbiorów, stwarza jednak problemy interpretacyjne. Wprowadzenie pojęcia zbioru wszystkich zbiorów nasuwa pytanie: skoro zbiór wszystkich zbiorów jest zbiorem, to czy nie powinien być swoim elementem? Próba odpowiedzi na takie pytanie prowadzi do paradoksu Russella. Pojęcie zbioru wszystkich zbiorów jest zatem wewnętrznie sprzeczne.

Dla zbiorów nieskończonych zachodzi charakterystyczna własność, polegająca na tym, że cały zbiór jest równoliczny z pewnym swoim podzbiorem właściwym. Ta własność jest podstawą formalnej definicji zbiorów nieskończonych. Zbiór jest *nieskończony* wtedy i tylko wtedy, gdy ma podzbiór właściwy, który jest z nim równoliczny.

Skończoność zbioru A oznacza, że istnieje $n \in Nat$ takie, że $|A| \sim n$.

Zapis $A \sim n$ wymaga komentarza. Należy przypomnieć, że liczby naturalne zdefiniowano w podrozdziale 4.2. jako najmniejszy zbiór induktywny. Symbol n jest zatem nazwą zbioru reprezentującego liczbę naturalną, co wyjaśnia sensowność zapisu $A \sim n$. W twierdzeniu 5.1 symbol liczby naturalnej n występuje w roli zbioru.

Twierdzenie 5.1

1. Dla dowolnej liczby $n \in Nat$ nie istnieje funkcja różnowartościowa z $n \cup \{n\}$ w n.
2. Dla $n, m \in Nat$, jeśli $m \sim n$, to $m = n$.
3. Żadna liczba naturalna nie jest równoliczna z Nat, a zatem Nat jest nieskończony.

Dowód twierdzenia można znaleźć na przykład w książce [Tiuryn 2003].

W tym przypadku mówimy, że A ma n elementów i oznaczamy to, pisząc $card(A)$. Takie oznaczenie było wprowadzone już wcześniej w rozdziale 2.

Przykład 5.1

a) Zbiór liczb parzystych *Parzyste* jest równoliczny ze zbiorem liczb naturalnych *Nat*. Wystarczy zauważyć, że funkcja $f: Nat \to Parzyste$, która wzajemnie jednoznacznie odwzorowuje zbiór *Nat* w zbiór *Parzyste*, jest zdefiniowana wzorem

$$f(n) = 2*n$$

w którym: $n \in Nat$, a $2*n \in Parzyste$.

b) Podobnie równoliczne są zbiory *Parzyste* i *Nieparzyste*.

c) Zbiór liczb rzeczywistych odcinka $[a, b]$, gdzie $a < b$, jest równoliczny ze zbiorem liczb rzeczywistych odcinka $[0, 1]$. Odpowiednią bijekcją jest tu $f: [a, b] \to [0, 1]$, zdefiniowana wzorem

$$f(x) = (x - a)/(b - a)$$

d) Zbiór liczb rzeczywistych jest równoliczny z podzbiorem liczb rzeczywistych odcinka otwartego $(-\pi/2, \pi/2)$. Wzajemnie jednoznaczne odwzorowanie reprezentuje tu funkcja tangens.

Definicja 5.2

Każdy zbiór skończony lub równoliczny ze zbiorem liczb naturalnych nazywa się *zbiorem przeliczalnym*. Zbiór nieskończony, który nie jest równoliczny ze zbiorem liczb naturalnych, nazywa się *zbiorem nieprzeliczalnym*.

5.2. Zbiory nieprzeliczalne

Twierdzenie 5.2

Zbiór potęgowy zbioru liczb naturalnych nie jest równoliczny ze zbiorem liczb naturalnych, czyli jest zbiorem nieprzeliczalnym. Oznacza to, że nie istnieje wzajemnie jednoznaczna funkcja $f: Nat \rightarrow 2^{Nat}$.

Dowód

Dowód pochodzi od Cantora i jest oparty na tzw. *metodzie przekątniowej*. Jeżeli zbiór potęgowy byłby równoliczny zbiorowi liczb naturalnych, to wszystkie podzbiory zbioru liczb naturalnych dałoby się ustawić w ciągi Z_1, Z_2, Z_3, ... w jeden ciąg. Każdy z tych zbiorów zawiera pewne liczby naturalne. Można to przedstawić w postaci tablicy, której przykładowa postać jest podana poniżej. Zbiór Z_1 w tej tablicy nie zawiera liczb 0, 1, 2, 3 itd.; zbiór Z_2 zawiera 0, 1, nie zawiera 2, 3 itd.

	0	1	2	3	
Z_1	nie	nie	nie	nie	...
Z_2	tak	tak	nie	nie	...
Z_3	nie	tak	nie	nie	...
Z_4	tak	tak	nie	tak	...
Z_5	...	...	...	...	...

Definiuje się teraz nowy zbiór Z' w taki sposób, aby był on różny od każdego ze zbiorów Z_1, Z_2, ... Poruszając się wzdłuż przekątnej tabeli od górnego lewego pola, definiuje się w sposób następujący przynależność kolejnej liczby naturalnej do zbioru Z': każde słowo „nie" zastępuje się słowem „tak", a każde słowo „tak" zastępuje się słowem „nie". Jeżeli po takim zastąpieniu na przekątnej w kolumnie k znajduje się „nie", oznacza to, że $k \notin Z'$, a jeżeli znajduje się „tak", oznacza to, że $k \in Z'$.

W rozpatrywanym przykładzie otrzymuje się mianowicie:

	0	1	2	3	
Z_1	tak	nie	nie	nie	...
Z_2	tak	nie	nie	nie	...
Z_3	nie	tak	tak	nie	...
Z_4	tak	tak	nie	nie	...
Z_5	...	...	...	...	...

Zbiór Z', zdefiniowany przez tak określoną przekątną, zawiera 0, nie zawiera 1, zawiera 2 itd.

Tak zdefiniowany zbiór Z' jest różny od każdego ze zbiorów Z_1, Z_2, ..., zatem Z' jest zbiorem, który nie daje się zestawić w ciąg wszystkich podzbiorów zbioru liczb naturalnych. Ponieważ rodzina podzbiorów 2^{Nat} jest nieskończona i nie jest *równoliczna* ze zbiorem liczb naturalnych, jest więc zbiorem nieprzeliczalnym. ■

Twierdzenie pokazuje, że istnieją co najmniej dwa rodzaje nieskończoności. Pierwszy reprezentuje nieskończoność liczb naturalnych i nazywa się nieskończonością przeliczalną. Pozostałe rodzaje nieskończoności nazywa się nieskończonościami nieprzeliczalnymi. Drugi rozpatrywany tu rodzaj nieskończoności, reprezentujący wszystkie podzbiory liczb naturalnych, reprezentuje rodzaj nieprzeliczalności zwany *continuum*.

Na konstrukcji podobnej do prezentowanej w poprzednim dowodzie opiera się dowód twierdzenia 5.3.

Twierdzenie 5.3

Przeliczalna suma mnogościowa zbiorów przeliczalnych jest zbiorem przeliczalnym.

Dowód

Niech Z_1, Z_2, ... będzie ciągiem zbiorów przeliczalnych. Niech $Z_i =_{def} \{z_{i1}, z_{i2}, ...\}$ dla $i = 1, 2, ...$ Elementy tych zbiorów można ustawić w tabelę:

	1	2	3	4	...
Z_1	z_{11}	z_{12}	z_{13}	z_{14}	...
Z_2	z_{21}	z_{22}	z_{23}	z_{24}	...
Z_3	z_{31}	z_{32}	z_{33}	z_{34}	...
Z_4	z_{41}	z_{42}	z_{43}	z_{44}	...
Z_5	...	...	...	...	...

Wszystkie te elementy, nie pomijając żadnego, można ustawić w jeden wspólny ciąg w sposób, który określają strzałki. Ciąg ten zawiera wszystkie elementy wszystkich ciągów Z_1, Z_2, ... i jest oczywiście równoliczny ze zbiorem liczb naturalnych, co dowodzi tezy twierdzenia. ■

Twierdzenie 5.4

Podzbiór zbioru przeliczalnego jest zbiorem przeliczalnym.

Dowód

Niech A będzie zbiorem przeliczalnym oraz $B \subseteq A$. Jeżeli $A = B$ lub B jest zbiorem skończonym, to oczywiście B jest zbiorem przeliczalnym. Niech B będzie zbiorem nieskończonym, różnym od A. Ponieważ A jest zbiorem przeliczalnym, istnieje wzajemnie jednoznaczna funkcja $f: Nat \rightarrow A$. Należy pokazać, że istnieje wzajemnie jednoznaczna funkcja $g: Nat \rightarrow B$. Funkcję tę można zdefiniować rekursywnie, na podstawie funkcji f, w sposób następujący:

- $g(0) = f(k_1)$, gdzie k_1 jest najmniejszą liczbą naturalną, taką że $f(k_1) \in B$,
- $g(n) = f(k_n)$, gdzie k_n jest najmniejszą liczbą naturalną, taką że $k_{n-1} < k_n$ oraz $f(k_n) \in B$.

Funkcja g jest różnowartościowa i przekształca Nat na B. Zbiór B jest zatem równoliczny ze zbiorem liczb naturalnych, a więc jest zbiorem przeliczalnym. ■

Przykładowym, ważnym zbiorem nieprzeliczalnym jest zbiór liczb rzeczywistych. Wynika to z następującego twierdzenia:

Twierdzenie 5.5

Zbiór liczb rzeczywistych z odcinka [0, 1] jest zbiorem nieprzeliczalnym.

Dowód

Wystarczy pokazać, że nie istnieje funkcja $f: Nat \rightarrow [0, 1]$, która jest bijekcją, to znaczy $dom(f) = Nat$ oraz $ran(f) = [0, 1]$. Funkcję f można przedstawić w postaci ciągu – zbioru par:

$$\{<0, f(0)>, <1, f(1)>, <0, f(2)>, \ldots\}$$

Dalej ciąg ten będzie zapisywany w uproszczonej postaci:

$$f(0), f(1), f(2), \ldots, f(n), \ldots$$

gdzie wyrazy ciągu $f(n)$ są liczbami z przedziału [0, 1].

Pokażemy, że dla dowolnie wybranego ciągu f istnieje liczba $c \in [0, 1]$, która nie należy do tego ciągu.

Oznaczmy przez $[a_n, b_n] \subseteq [0, 1]$, dla $n \in Nat$, ciąg przedziałów z odcinka [0, 1]. Ciąg tych przedziałów jest zdefiniowany następująco:

Niech $[a_0, b_0] = [0, 1]$. Podzielmy ten odcinek na trzy podprzedziały: [0, 1/3], [1/3, 2/3], [2/3, 1] i wybierzmy z nich ten podprzedział, do którego nie należy wyraz $f(0)$. Wybrany podprzedział oznaczmy przez $[a_1, b_1]$. Podobnie postępując z przedziałem $[a_1, b_1]$, wyznaczymy przedział $[a_2, b_2]$, do którego nie należy wyraz $f(1)$ itd.

Na mocy konstrukcji wyznaczony ciąg podprzedziałów $[a_n, b_n]$, dla $n \in Nat$, ma następujące własności:

$$f(n-1) \notin [a_n, b_n],$$

$$b_n - a_n = 1/3^n$$

$$0 \leq a_n \leq a_{n+1} < b_{n+1} \leq b_n \leq 1.$$

Ciągi liczbowe $\{a_n\}_{n \in Nat}$ oraz $\{b_n\}_{n \in Nat}$ są monotoniczne i ograniczone. Są one zbieżne do tej samej granicy $c \in [0, 1]$, gdyż $\lim_{n \to \infty}(b_n - a_n) = 0$. Liczba c należy do każdego z przedziałów $[a_n, b_n]$, a więc jest różna od każdego wyrazu $f(n)$, czyli nie należy do ciągu wyznaczonego przez funkcję f, z czego wynika teza.

Zauważmy, że podział danego odcinka na dwa podprzedziały domknięte, np. [0, ½], [1/2, 1] itd., nie pozwoliłby na zastosowanie opisanego postępowania dla przypadków, gdy ciąg $f(0)$, $f(1)$, $f(2)$, ..., $f(n)$, ... byłby zbieżny do $1/2^k$, dla dowolnego $k \in Nat$. Możliwe natomiast byłoby przeprowadzenie tego postępowania przy założeniu podziału danego odcinka na dowolną, większą niż 3, liczbę podprzedziałów. ■

Twierdzenie 5.6

Zbiór liczb rzeczywistych z odcinka otwartego (0, 1) jest równoliczny zbiorowi punktów wnętrza kwadratu o boku długości 1.

Szkic dowodu

Każdy punkt kwadratu można określić parą liczb $<x, y>$, stanowiących współrzędne punktu. Przy założeniu, że bok kwadratu ma długość jeden, liczby te można zapisać w postaci nieskończonych ułamków dziesiętnych w postaci:

$$x = 0, \alpha_1 \alpha_2 \ldots \alpha_n \ldots$$

$$y = 0, \beta_1 \beta_2 \ldots \beta_n \ldots$$

gdzie α_n, β_n (dla $n = 1, 2, \ldots$) są cyframi 0, 1, ..., 9.

Parze liczb $<x, y>$ przyporządkowujemy liczbę z na odcinku (0, 1), również reprezentowaną nieskończonym ułamkiem dziesiętnym o postaci

$$z = 0, \alpha_1 \beta_1 \alpha_2 \beta_2 \ldots \alpha_n \beta_n \ldots$$

Jest oczywiste, że przy takim przyporządkowaniu różnym punktom kwadratu odpowiadają różne punkty odcinka. Wynika stąd, że zbiór punktów wnętrza kwadratu o boku jednostkowym jest równoliczny z pewnym podzbiorem odcinka o długości jeden. Moc zbioru punktów kwadratu jest zatem nie większa od mocy zbioru punktów odcinka.

Z drugiej strony zbiór punktów odcinka jest podzbiorem punktów kwadratu. Moc zbioru punktów odcinka jest zatem nie większa od mocy zbioru punktów kwadratu, z czego wynika, że obie moce są równe. ■

5.3. Liczby kardynalne

Pojęcie równoliczności zbiorów jest bardzo ważne. Jest ono między innymi punktem wyjścia do współczesnej definicji liczby. Równoliczność wprowadza pewną klasyfikację zbiorów. Zbiory tego samego rodzaju są równoliczne. Rodzaje te nazywa się *liczbami kardynalnymi*. Takim rodzajem jest na przykład rodzina zbiorów czteroelementowych. Liczby naturalne są odpowiednikiem liczb kardynalnych dla zbiorów skończonych. Liczbą kardynalną zbioru wszystkich liczb naturalnych jest $\aleph_0$ (*alef zero*), a liczbą kardynalną rodziny wszystkich podzbiorów zbioru liczb naturalnych jest $\mathbf{c}$ (*continuum*). Inaczej: liczba kardynalna jest pewną cechą zbioru.

Liczbę kardynalną zbioru A, czyli rodzaj zbiorów, do którego należy zbiór A, będziemy oznaczać przez $|A|$. Liczby kardynalnej nie należy utożsamiać z pojęciem liczby. W szczególności, w przypadku zbioru skończonego A, jego liczba kardynalna $|A|$ jest czym innym niż liczba elementów tego zbioru $card(A)$.

Liczby kardynalne można ze sobą porównywać. Niech $|A| = \alpha$ oraz $|B| = \beta$.

Przyjmujemy, że liczba kardynalna α *jest nie większa* niż liczba kardynalna β, co piszemy $\alpha \leq \beta$, jeżeli zbiór A jest równoliczny z podzbiorem zbioru B.

Jeżeli $\alpha \leq \beta$ oraz $\alpha \neq \beta$, to mówimy, że liczba kardynalna α *jest mniejsza* niż liczba kardynalna β, co piszemy $\alpha < \beta$.

Z wcześniejszych rozważań wynika więc, że $\aleph_0 < \mathbf{c}$.

Porównywanie liczb kardynalnych ma własność zwrotności, to znaczy

$$\alpha \leq \alpha$$

i przechodniości, to znaczy

jeżeli $\alpha \leq \beta$ oraz $\beta \leq \gamma$, to $\alpha \leq \gamma$.

Ważny związek pomiędzy liczbami kardynalnymi wyraża podane niżej twierdzenie Cantora–Bernsteina.

Twierdzenie 5.7

Dla dowolnych liczb kardynalnych α, β:

jeżeli $\alpha \leq \beta$ oraz $\beta \leq \alpha$, to $\alpha = \beta$.

Dowód twierdzenia można znaleźć na przykład w pracy [Rasiowa 1998].

Uogólnieniem twierdzenia 5.3 jest twierdzenie Cantora.

Twierdzenie 5.8

Dla dowolnego zbioru A: $|A| < |2^A|$.

Dowód

Twierdzenie oznacza, że nie istnieje wzajemnie jednoznaczna funkcja odwzorowująca zbiór A w zbiór potęgowy 2^A. Załóżmy, że $f: A \to 2^A$ jest funkcją na 2^A. Niech

$$A_0 = \{a \in A \mid a \notin f(a)\}$$

Ponieważ f jest funkcją na 2^A, to istnieje $a_0 \in A$ taki, że $f(a_0) = A_0$, tak więc $a_0 \in A_0$ wtedy i tylko wtedy, gdy $a_0 \in f(a_0)$.

Innymi słowy: $a_0 \in A_0$ wtedy i tylko wtedy, gdy $a_0 \notin A_0$.

Otrzymana sprzeczność dowodzi, że nie istnieje wzajemnie jednoznaczna funkcja odwzorowująca zbiór A w zbiór potęgowy 2^A. ■

Ćwiczenia

1. Pokazać równoliczność zbioru liczb naturalnych i zbioru liczb pierwszych.
2. Pokazać równoliczność zbiorów:
   a) odcinek otwarty $(0, 1) \subset$ *Rzeczywiste*,
   b) odcinek półotwarty $[0, 1) \subset$ *Rzeczywiste*,
   c) okręg na płaszczyźnie o środku $(0, 0)$ i promieniu 1.
3. Pokazać, że zbiór potęgowy zbioru A jest równoliczny ze zbiorem funkcji określonych na A i o wartościach w zbiorze $\{0, 1\}$, czyli ze zbiorem $\{f \mid f: A \to \{0, 1\}\}$.
4. Ile jest rosnących ciągów liczb wymiernych zbieżnych do 1?
5. Ile jest relacji równoważności na zbiorze liczb naturalnych takich, że wszystkie ich klasy abstrakcji są skończone?
6. Udowodnić, że każdy zbiór rozłącznych odcinków na prostej jest przeliczalny. Pokazać, że istnieje nieprzeliczalny zbiór rozłącznych odcinków na płaszczyźnie.
7. Udowodnić, że jeżeli A nie jest zbiorem przeliczalnym i B jest zbiorem przeliczalnym, to A/B nie jest zbiorem przeliczalnym.
8. Udowodnić, że produkt kartezjański dwóch zbiorów przeliczalnych jest zbiorem przeliczalnym.
9. Udowodnić, że zbiór liczb wymiernych jest przeliczalny.
10. Udowodnić, że zbiór liczb niewymiernych jest nieprzeliczalny.
11. Udowodnić, że każdy zbiór nieskończony zawiera pewien podzbiór przeliczalny.
12. Udowodnić, że rodzina wszystkich skończonych podzbiorów zbioru przeliczalnego jest przeliczalna.

13. Udowodnić, że dla dowolnych liczb kardynalnych α, β, γ, zachodzą związki:
    a) $\alpha \leq \alpha$,
    b) jeżeli $\alpha \leq \beta$ oraz $\beta \leq \gamma$, to $\alpha \leq \gamma$.

14. Czy istnieje relacja równoważności $R \subseteq Rzeczywiste^2$, której każda klasa abstrakcji jest mocy $\aleph_0$ oraz:
    a) zbiór ilorazowy *Rzeczywiste/R* jest mocy $\aleph_0$?
    b) zbiór ilorazowy *Rzeczywiste/R* jest mocy $\mathfrak{c}$?

15. Które z poniższych zdań jest prawdziwe?
    a) Jeśli $f: A \to B$ jest różnowartościowa oraz nie jest na B, to $|A| < |B|$.
    b) Jeśli $|A| < |B|$ oraz $C \neq \emptyset$, to $|A \times C| < |B \times C|$.

6. Zbiory i funkcje obliczalne

6.1. Zbiory obliczalne i rekurencyjne

Z wykonywaniem obliczeń za pomocą komputerów wiąże się pojęcie *algorytmu*. Obliczeniami na komputerach rządzą ścisłe reguły, niekiedy mówi się nawet o regułach mechanicznych, mając na myśli to, że obliczenie można widzieć jako skończonej długości ciąg czynności, z których każda jest jednoznacznie zdefiniowana i – dodatkowo – jest realizowalna za pomocą ściśle zdefiniowanych środków. Do rozwiązania niektórych problemów, na przykład zadania obliczenia największego wspólnego podzielnika dwóch liczb, można wyobrazić sobie istnienie pewnego algorytmu. Trudno natomiast wyobrazić sobie istnienie algorytmu do przepowiadania wyniku rzutu monetą.

Określone tak intuicyjnie pojęcie algorytmu jest nieformalne, co utrudnia lub nawet uniemożliwia jego wykorzystanie w ścisłych rozważaniach. Próby ścisłego zdefiniowania pojęcia algorytmu[9] podjęto w pierwszej połowie ubiegłego stulecia. Ich rezultatem było powstanie kilku – jak się później okazało – równoważnych definicji algorytmu.

Niektóre z tych podejść wiązały się z ograniczeniem czynności wykonywanych w ramach algorytmu do manipulacji na symbolach. Oznacza to, że wykonywanie czynności polega na tworzeniu pewnych ciągów symboli (napisów) na podstawie innych ciągów symboli (napisów). Przykładem są tu algorytmy normalne Markowa[10].

Inne podejścia były oparte na wprowadzeniu pewnych „maszyn", które byłyby zdolne do samodzielnego wykonywania przetwarzania napisów. Przykładem znanych modeli algorytmu są maszyny opracowane przez Turinga[11] oraz Posta[12]. Oba te modele, opracowane niezależnie od siebie, są podobne i wiąże się z nimi hipoteza, nazywana *hipotezą Turinga–Posta* lub – częściej – *hipotezą Turinga*. Z tego względu dalej przedsta-

[9] Termin *algorytm* pochodzi od nazwiska arabskiego matematyka Abu Ja'far Muhammad ibn Musa Al-Kwarizmi (około 780–850).
[10] Andriej A. Markow (ur. 1903), syn innego matematyka Andrieja A. Markowa (1856–1922).
[11] Alan M. Turing (1921–1954).
[12] Emil Post (1897–1954).

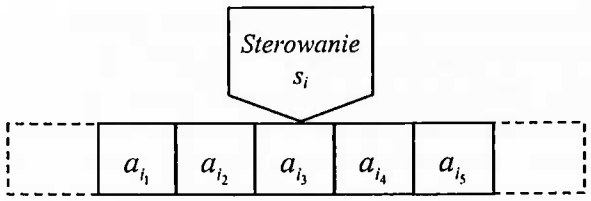
wiono tylko *maszynę Turinga*, a związana z nią, sformułowana w 1936 roku, hipoteza stwierdza [Arbib 1968]:

> *Nieformalne, intuicyjne pojęcie algorytmu na ciągach symboli jest tożsame ze ścisłym pojęciem procedury, którą może wykonać maszyna Turinga.*

Dla hipotezy tego rodzaju nie można nigdy podać formalnego dowodu, ponieważ taki dowód wymaga zdefiniowania pojęć, które zawiera. Można ją tylko obalić przez podanie przykładu intuicyjnie rozwiązywalnego problemu, dla którego nie daje się skonstruować odpowiedniej maszyny Turinga. Jak dotychczas, ilekroć było intuicyjnie oczywiste, że algorytm istnieje, tylekroć okazywało się możliwe skonstruowanie maszyny Turinga wykonującej ściśle ten algorytm i nie ma przesłanek, które wskazywałyby na możliwość zmiany tego stanu rzeczy.

Rozwiązanie pewnego problemu przez wyszukanie odpowiedniego algorytmu sprowadza się zatem do zbudowania odpowiedniej maszyny Turinga. Maszyna Turinga jest tylko konstrukcją teoretyczną i nie służy do rozwiązywania zagadnień praktycznych. Do celów praktycznych wystarczy przyjąć, że maszynę Turinga można utożsamiać z dowolnym komputerem, który dysponuje nieograniczenie pojemną pamięcią. Dokładny jej opis przedstawia się następująco:

Maszyna Turinga jest określona jako urządzenie składające się z nieskończenie długiej taśmy, podzielonej na kratki, zwanej *pamięcią maszyny*, oraz urządzenia sterującego, zwanego *sterowaniem maszyny*. W każdej kratce taśmy może być zapisany jeden z symboli ustalonego, skończonego zbioru symboli A, nazywanego *alfabetem maszyny*. Zakłada się, że alfabet zawiera symbol „pusty": *nic*. Urządzenie sterujące – krócej: maszyna – może się znajdować w jednym ze stanów skończonego zbioru S. Wyróżnia się pewien podzbiór stanów $F \subseteq S$, nazywanych *stanami końcowymi*. W każdym stanie urządzenie sterujące może „obserwować" (przez głowicę czytająco-piszącą) jedną kratkę taśmy T. Gdyby kratki taśmy T można ponumerować liczbami całkowitymi, wtedy znajdujący się w obserwowanej kratce o numerze $i \in$ *Całkowite* symbol $a_i \in A$ nazywałby się *symbolem obserwowanym*.



Rys. 6.1. Maszyna Turinga

W danym momencie czasu maszynę charakteryzuje jej *konfiguracja*, na którą składa się stan pamięci, czyli aktualna zawartość taśmy, stan urządzenia sterującego oraz położenie głowicy czytająco-piszącej przy jednym z pól taśmy.

Maszyna rozpoczyna pracę w *konfiguracji początkowej*, to jest takiej, w której taśma T ma ustaloną początkową zawartość. Aktualnym stanem urządzenia sterującego jest wyróżniony *stan początkowy*, a głowica czytająco-pisząca znajduje się przy kratce taśmy T wskazanej jako kratka początkowa. W danej konfiguracji takiej, w której stan urządzenia sterującego jest stanem s, maszyna „czyta" obserwowany symbol a, po czym:

- zmienia swój stan s na nowy stan $s' \in S$,
- zmienia obserwowany symbol a na $a' \in A$, w szczególności może to być ten sam symbol a lub symbol pusty,
- przesuwa swoją głowicę czytająco-piszącą o jedną kratkę w lewo albo w prawo, albo pozostawia ją w miejscu.

W ten sposób maszyna przechodzi do nowej konfiguracji i powtarza swoje działanie według opisanego schematu aż do momentu, gdy osiągnie *konfigurację końcową*, to jest taką, w której stan aktualny urządzenia sterującego określa się jako stan końcowy. Po osiągnięciu konfiguracji końcowej maszyna zatrzymuje się i nie wykonuje dalszych czynności.

Formalnie maszynę Turinga MT definiuje się jako siódemkę:

$$MT = <S, A, \delta, \lambda, \rho, s_0, F>$$

gdzie:

S jest zbiorem stanów,
A jest alfabetem,
$\delta : S \times A \to S$ jest funkcją przejść pomiędzy stanami,
$\lambda : S \times A \to A$ jest funkcją wyjść,
$\rho : S \times A \to \{-1, 0, 1\}$ jest funkcją przesunięcia głowicy,
s_0 jest stanem początkowym,
$F \subseteq S$ jest podzbiorem stanów końcowych.

Obliczenie maszyny Turinga MT dla danej taśmy T można opisać w postaci ciągu trójek:

$$< s_0, a_{i_0}, i_0 >, < s_1, a_{i_1}, i_1 >, \ldots, < s_n, a_{i_n}, i_n >, \ldots$$

gdzie:

s_k jest stanem urządzenia sterującego,
a_{i_k} jest zawartością kratki obserwowanej przez głowicę czytająco-piszącą,
i_k oznacza kratkę taśmy T, przy której znajduje się głowica czytająco-pisząca.

Pierwsza trójka $< s_0, a_{i_0}, i_0 >$ jest elementem konfiguracji początkowej maszyny, a następne trójki są określone następująco:

$$s_{k+1} = \delta(s_k, a_{i_k})$$
$$a'_{i_k} = \lambda(s_i, a_{i_k})$$
$$i_{k+1} = i_k + \rho(s_i, a_{i_k})$$

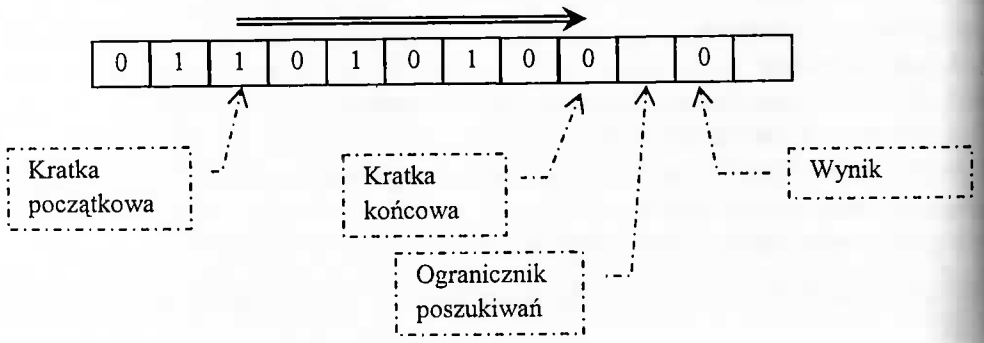
dla $k \in Nat$. Pierwsza z funkcji określa nowy stan, druga – nową zawartość przeczytanej kratki, a trzecia – wskazuje na numer kolejnej kratki, do której przesuwa się głowica.

Ciąg ten zawsze rozpoczyna się w konfiguracji początkowej i może być skończony lub nieskończony. Jeżeli jest on ciągiem skończonym, to ostatnia trójka jest elementem konfiguracji końcowej, co oznacza, że $s_n \in F$.

Obliczenie maszyny rozpoczyna się i kończy pewnym ciągiem symboli zapisanych na taśmie. Symbole na taśmie przed rozpoczęciem obliczeń interpretuje się jako dane algorytmu, a symbole po wykonaniu obliczeń maszyny interpretuje się jako wynik obliczeń algorytmu.

Przykład 6.1

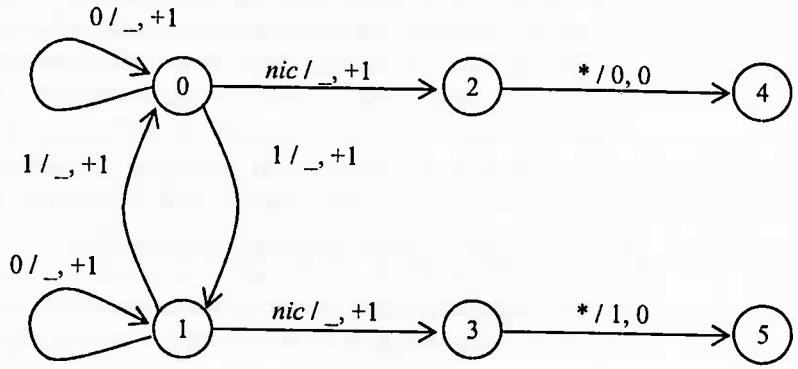
Alfabet maszyny Turinga $A = \{0, 1, nic\}$. Zadaniem maszyny Turinga jest stwierdzenie, czy liczba symboli 1 zapisana na taśmie, poczynając od wskazanej niepustej kratki aż do pierwszej kratki po prawej stronie, która zawiera symbol *nic*, jest parzysta czy nieparzysta. Wynik obliczeń maszyny ma być zapisany na taśmie, w kratce sąsiadującej po prawej stronie z pierwszą kratką zawierającą znaleziony symbol *nic* (kratka pusta).



Rys. 6.2. Przykładowa zawartość taśmy

Maszynę Turinga można w zwarty sposób przedstawić za pomocą diagramu przejść pomiędzy stanami. Diagram ten jest grafem, którego wierzchołkami są stany maszyny. Łuki reprezentujące przejścia pomiędzy stanami są etykietowane napisami postaci a / b, c. Pierwszy element a jest symbolem alfabetu maszyny, symbol * oznacza, że może to być dowolny element. Drugi element b jest symbolem, który maszyna wypisuje na taśmie, symbol _ oznacza, że napis w danej kratce nie ulega zmianie. Trzeci element c może przyjąć wartości –1, 0, +1, co oznacza przesunięcie głowicy maszyny odpowiednio w lewo, brak przesunięcia lub przesunięcie w prawo. Ukośnik rozdziela symbol wejściowy od symboli, które reprezentują reakcję maszyny w danej konfiguracji. Łącznie etykieta a / b, c na przejściu ze stanu

i do stanu j oznacza, że: jeżeli w stanie i maszyna przeczyta w obserwowanej kratce symbol a, to zapisuje w tej kratce symbol b i przesuwa się do następnej kratki o c, po czym przechodzi do stanu j.



Rys. 6.3. Diagram przejść pomiędzy stanami

Przedstawiony diagram wymaga uzupełnienia o wskazanie stanu początkowego – stan 0 – oraz stanów końcowych – stany 4, 5.

Maszyna Posta, a także inne maszyny, na przykład Rabina i Scotta, maszyny wielotaśmowe są równoważne maszynie Turinga w tym sensie, że jeżeli dany problem daje się rozwiązać przez zbudowanie jakiejkolwiek z tych maszyn, to daje się również rozwiązać za pomocą pozostałych maszyn.

Przyjmując maszynę Turinga za pojęcie algorytmu, można sformułować ważne pojęcia. Niech dany będzie pewien zbiór przeliczalny A.

Zbiór A jest *rekurencyjny*, jeżeli istnieje algorytm rozstrzygania, czy dany element jest czy nie jest elementem A.

Zbiór A jest *rekurencyjnie przeliczalny*, jeżeli istnieje algorytm, który wylicza wszystkie jego elementy.

Z definicji bezpośrednio wynika, że jeżeli podzbiór liczb naturalnych jest rekurencyjny, to jest rekurencyjnie przeliczalny, ale nie odwrotnie, gdyż okazuje się, że zachodzi twierdzenie:

Twierdzenie 6.1

Istnieje rekurencyjnie przeliczalny zbiór liczb naturalnych, który nie jest rekurencyjny.

Dowód twierdzenia można znaleźć na przykład w książce [Arbib 1968].

6.2. Funkcje obliczalne i rekurencyjne

Pojęcie funkcji obliczalnych wiąże się z funkcjami określonymi na zbiorze liczb naturalnych i o wartościach w zbiorze liczb naturalnych. Intuicyjnie, funkcje obliczalne to takie funkcje, których wartości dla dowolnych argumentów można obliczyć na komputerze w skończonej liczbie kroków. Formalnie funkcja jest obliczalna, gdy istnieje algorytm jej obliczenia, inaczej – istnieje dla niej odpowiednia maszyna Turinga. Próby scharakteryzowania funkcji obliczalnych doprowadziły do wyłonienia klasy funkcji rekurencyjnych. Okazało się, że funkcje obliczalne są funkcjami rekurencyjnymi. Dokładniej wyraża to powszechnie akceptowana hipoteza Churcha, która stwierdza, że:

Każda funkcja obliczalna w nieformalnym sensie jest rekurencyjna.

Definicja klasy *funkcji rekurencyjnych* opiera się na zbiorze pewnych funkcji elementarnych i zbiorze operacji, które pozwalają na konstruowanie z funkcji elementarnych nowych funkcji.

Zbiór funkcji elementarnych zawiera:

- funkcję następnika zdefiniowaną wzorem $Succ(x) = x + 1$,
- funkcję tożsamościową $I(x) = x$,
- funkcje rzutowania $p_i(x_1, ..., x_n) = x_i$, dla $i = 1, ..., n$,
- funkcję zeroargumentową – stałą 0.

Zbiór operacji na funkcjach składa się z trzech operacji. Pierwszą jest – omówiona wcześniej – operacja składania funkcji, dwie pozostałe operacje – nazywane operacją rekursji prostej i operacją minimum – wymagają zdefiniowania.

Operacja rekursji prostej polega na tym, że mając dwie funkcje:

$f : Nat^{n-1} \to Nat$ oraz $g : Nat^{n+1} \to Nat$ dla $n \in Nat \setminus \{0\}$

nową funkcję

$h : Nat^n \to Nat$

definiuje się za pomocą dwóch następujących równości:

$h(x_1, ..., x_{n-1}, 0) = f(x_1, ..., x_{n-1})$

$h(x_1, ..., x_{n-1}, Succ(x_n)) = g(x_1, ..., x_n, h(x_1, ..., x_n))$

Termin rekursja prosta, wprowadzony przez Hilberta[13] i Bernaysa[14] w 1934 roku, nie jest szczęśliwy, gdyż schemat generacji wartości funkcji bardziej się wiąże z iteracją, z jaką mamy do czynienia w językach programowania, niż z rekursją. Rekursja prosta wyraża pewien indukcyjny sposób definiowania wartości.

[13] David Hilbert (1862–1943).

[14] Paul I. Bernays (1888–1977).

Funkcje, które daje się zdefiniować za pomocą operacji składania i rekursji prostej, nazywa się funkcjami *pierwotnie rekurencyjnymi*.

Przykład 6.2

Funkcję dodawania liczb naturalnych, reprezentowaną symbolem + w notacji przedrostkowej, definiuje się za pomocą operacji rekursji prostej w sposób następujący:

$+(x, 0) = I(x)$

$+(Succ(y), x) = Succ(p_3(x, y, +(x, y)))$

W tym przypadku rolę definiowanej funkcji h pełni +, funkcja f jest funkcją tożsamościową I, a funkcja g jest złożeniem funkcji następnika $Succ$ z funkcją rzutowania p_3. Tę samą definicję, w sposób równoważny, można zapisać prościej:

$+(x, 0) = x$

$+(Succ(y), x) = Succ(+(x, y))$

Po zastosowaniu notacji wrostkowej dla funkcji dwuargumentowych definicja przyjmie jeszcze bardziej czytelną postać:

$x + 0 = x$

$Succ(y) + x = Succ(x + y)$

Korzystając z funkcji dodawania, podobnie można zdefiniować funkcję mnożenia:

$x * 0 = 0$

$Succ(y) * x = (x * y) + x$

Przykład 6.3

Odejmowanie w zbiorze liczb naturalnych jest funkcją zdefiniowaną częściowo. Definiowana poniżej funkcja *dif*, określona dla wszystkich liczb naturalnych, jest tylko pewnym odpowiednikiem odejmowania w zbiorze liczb całkowitych. Jej definicja wymaga wprowadzenia funkcji pomocniczej *Pred*, nazywanej funkcją *poprzednika*. Rekursywna definicja jednoargumentowej funkcji poprzednika ma postać:

$Pred(0) = 0$

$Pred(Succ(x)) = x$

Odejmowanie w dziedzinie liczb całkowitych nieujemnych, oznaczone *dif* w celu odróżnienia od symbolu odejmowania „–” w zbiorze liczb całkowitych, jest zdefiniowane metodą rekursji prostej:

$dif(x, 0) = 0$

$dif(x, Succ(y)) = Pred(dif(x, y))$

Metodą rekursji prostej można definiować różne funkcje, między innymi funkcje określone wariantowo.

Przykład 6.4

Niech dana będzie funkcja

$$h(x)=\begin{cases}2x & \text{dla } x\le 3\\ 2x-2 & \text{dla } x>3\end{cases}$$

Jej definicja wymaga uprzedniego zdefiniowania funkcji pomocniczych: jednoargumentowej funkcji znaku *sg*, definiowanej rekursją prostą (przypadek zdegenerowany):

$sg(0) = 0$

$sg(Succ(x)) = 1$

oraz funkcji porównań definiowanych przez wyrażenia funkcyjne:

$gt(x, y) = sg(dif(x, y))$

$ge(x, y) = gt(Succ(x), y)$

Definicja funkcji h przybierze zatem postać wyrażenia funkcyjnego

$((2*x) * ge(3, x)) + (dif((2*x), 2) * gt(x, 3))$

Niestety, za pomocą operacji rekursji prostej nie można definiować dowolnych funkcji. Przykładem funkcji, która nie daje się zdefiniować w ten sposób, jest przedstawiana już poprzednio, w rozdziale 3, funkcja Ackermanna[15]

$$Ack(x,y)=\begin{cases}y+1 & \text{gdy } x=0\\ Ack(x-1,1) & \text{gdy } y=0\\ Ack(x-1,Ack(x,y-1)) & \end{cases}$$

W 1936 roku Kleene[16] uzupełnił listę schematów kompozycji funkcji o operację minimum, albo inaczej – operację μ-rekursji.

Niech dana będzie funkcja

$f: Nat^{n+1} \to Nat$

taka, że dla każdych $x_1, ..., x_n \in Nat$ istnieje $y \in Nat$ takie, że $f(x_1, ..., x_n, y) = 0$.
Operacja minimum dla funkcji $f: Nat^{n+1} \to Nat$ polega na zdefiniowaniu nowej funkcji

$h : Nat^n \to Nat$

[15] Wilhelm Ackermann (1896–1962).

[16] Stephen Kleene (1909–1994).

która spełnia warunek

$f(x_1, ..., x_n, h(x_1, ..., x_n)) = 0$

oraz dodatkowo – aby zapewnić jednoznaczność definicji funkcji h – warunek

$h(x_1, ..., x_n)$ jest równe najmniejszej wartości $y \in Nat$ takiej, że $f(x_1, ..., x_n, y) = 0$.

Ostatni warunek zapisuje się też w postaci

$h(x_1, ..., x_n) = \mu y[f(x_1, ..., x_n, y) = 0]$

Symbol μy oznacza najmniejszą wartość y, dla której, przy danych wartościach $x_1, ..., x_n$, jest spełniony warunek $f(x_1, ..., x_n, y) = 0$.

Operacja minimum wyznacza więc funkcję, która przyjmuje wartość $h(x_1, ..., x_n) = y$ wtedy i tylko wtedy, gdy:

- $f(x_1, ..., x_n, y) = 0$
- dla dowolnego $y' < y$ $f(x_1, ..., x_n, y') \neq 0$.

Jeżeli dla danego zestawu wartości $x_1, ..., x_n$ funkcja f nie spełnia podanych warunków, to funkcja h dla tych wartości nie jest zdefiniowana.

Przykład 6.5

Operację minimum wykorzystano do definicji funkcji

$h_1(x, y) = (\mu z)[isg(eq(y * z, x)) = 0]$

Funkcja h_1 definiuje najmniejszą wartość z taką, że $y * z = x$. Gdy x nie jest wielokrotnością y, funkcja ta nie jest określona.

$h_2(x, y) = (\mu z)[y * gt(x, y * Succ(z)) = 0]$

Funkcja h_2 wyznacza część całkowitą z dzielenia x przez y.

$h_3(x, y) = (\mu z)[dif(x, z) = 0]$

Funkcja h_3 równa się x dla dowolnych x, y; jest więc równoważna funkcji projekcji p_1.

Funkcje, które można zdefiniować za pomocą operacji składania, rekursji pierwotnej i minimum nazywa się też funkcjami *ogólnie rekurencyjnymi*.

Uwaga

Chociaż funkcje rekurencyjne są definiowane jako funkcje przekształcające liczby naturalne w liczby naturalne, to ich własności można przenieść na dowolne funkcje, których zbiory argumentów i wartości dają się opisać skończonymi ciągami symboli. Niech $\{a_1, a_2, ..., a_n, ...\}$ będzie zbiorem symboli wykorzystywanych do

tworzenia takich opisów. Każdemu opisowi argumentu lub wartości funkcji, który jest reprezentowany przez skończony ciąg postaci

$$a_{i_1}, a_{i_2}, \ldots, a_{i_k}$$

można przyporządkować w jednoznaczny sposób liczbę naturalną. Uniwersalny sposób kodowania, zwany numeracją Gödla, polega na przyporządkowaniu temu ciągowi liczby

$$2^{i_1}3^{i_2}\ldots p_k^{i_k}$$

gdzie 2, 3, ..., p_k są kolejnymi liczbami pierwszymi. Dzięki jednoznaczności rozkładu liczb naturalnych na czynniki pierwsze można sprawdzić, czy dana liczba jest przyporządkowana jakiemuś opisowi, a jeśli tak – to jakiemu.

Ćwiczenia

1. Maszynę Turinga, przedstawioną w przykładzie 6.1, rozbudować w taki sposób, aby stwierdzała parzystą bądź nieparzystą liczbę symboli 1 pomiędzy pierwszą kratką po lewej i pierwszą kratką po prawej stronie początkowego położenia głowicy, które zawierają symbol *nic*.
2. Zdefiniować maszynę Turinga, która jako daną wejściową przyjmuje liczbę naturalną w zapisie binarnym i produkuje jako wynik tę samą liczbę zwiększoną o jeden, również w zapisie binarnym.
3. Zdefiniować maszynę Turinga, która jako dane wejściowe przyjmuje n-elementowy ciąg znaków, $n \in Nat\backslash\{0\}$ oraz liczbę $k \in \{1, \ldots, n\}$ i produkuje jako wynik k-ty element wejściowego ciągu.
4. Wykorzystując operację rekursji prostej, zdefiniować funkcję:
   a) potęgowania,
   b) minimum i maksimum dwóch liczb,
   c) dzielenia całkowitego i reszty z dzielenia całkowitego.
5. Zdefiniować jako funkcję rekurencyjną funkcję:
   a) najmniejszej wspólnej wielokrotności dwóch liczb naturalnych,
   b) największego wspólnego podzielnika dwóch liczb.
6. Zakładając, że znane są operacje dodawania i mnożenia na liczbach naturalnych, definiować rekursywnie operacje dodawania i mnożenia na liczbach całkowitych.

7. Języki formalne i gramatyki

7.1. Ciągi i słowa

Zbiory są nieuporządkowaną kolekcją pewnych elementów. Często potrzebne jest wprowadzenie uporządkowania wśród rozważanej kolekcji obiektów, a jednym ze sposobów uporządkowania jest zdefiniowanie ciągu.

Niech A będzie dowolnym zbiorem. Niepustym *ciągiem* o długości $n \in Nat\backslash\{0\}$ nad zbiorem A będzie się nazywać dowolną całkowicie określoną funkcję o sygnaturze

$$s : \{1, \ldots, n\} \to A$$

Ciąg o długości zero jest ciągiem *pustym* i będzie oznaczany symbolem ε.

Przez $CiągiSkończone_n(A)$ będzie oznaczany zbiór wszystkich ciągów skończonych długości $n \in Nat$ nad zbiorem A. Z definicji:

$$CiągiSkończone_0(A) =_{def} \{\varepsilon\}$$
$$CiągiSkończone_n(A) =_{def} \{s \mid s : \{1, \ldots, n\} \to A \wedge dom(s) = \{1, \ldots, n\}\}$$

dla $n \in Nat\backslash\{0\}$.

Zbiorem wszystkich ciągów skończonych będzie zatem

$$CiągiSkończone(A) =_{def} \bigcup_{n \in Nat} CiągiSkończone_n(A)$$

Zbiór wszystkich ciągów nieskończonych nad A jest zdefiniowany jako

$$CiągiNieskończone(A) =_{def} \{s \mid s : Nat\backslash\{0\} \to A \wedge dom(s) = Nat\backslash\{0\}\}$$

Zbiorem wszystkich ciągów nad A jest zatem zbiór

$$Ciągi(A) =_{def} CiągiSkończone(A) \cup CiągiNieskończone(A)$$

Dalej będą używane następujące oznaczenia: Niech s będzie niepustym ciągiem nad A. Wartość funkcji s dla argumentu i, czyli $s(i)$, oznacza i-ty element ciągu. Skończony ciąg s o długości $n \in Nat\backslash\{0\}$ nad zbiorem A jest zbiorem par:

$$\{<1, s(1)>, \ldots, <n, s(n)>\}$$

a nieskończony ciąg jest zbiorem par:

$$\{<1, s(1)>, \ldots, <n, s(n)>, \ldots\}$$

Zwykle używa się uproszczonego zapisu ciągu, odpowiednio w postaci:

$s(1)\ s(2)\ ...\ s(n)$ lub $s(1)\ s(2)\ ...\ s(n)\ ...$

albo

$a_1\ a_2\ ...\ a_n$ lub $a_1\ a_2\ ...\ a_n\ ...$

gdzie $a_i = s(i)$ dla $i = 1, ..., n, ...$ dla $i \in Nat\backslash\{0\}$.

Ciągi zapisywane w uproszczonej postaci będą oznaczane literami greckimi α, β, γ itd. Będzie się pisać na przykład $\alpha =_{def} a_1\ a_2\ ...\ a_n$, a przez $len(\alpha)$ będzie się oznaczać długość ciągu α. Oczywiście $len(\alpha) = 1$.

Równość ciągów oznacza równość reprezentujących je funkcji. Zapis $\alpha = \beta$ oznacza, że ciąg α jest identyczny z ciągiem β.

Przykład 7.1

Ciągami nad $A = \{0, 1, 2, 4, 5\}$ będą napisy:

0
001
12345

Ciągami nad *Nat* będą napisy:

11
1 1
11 1 0 54

Należy zwrócić uwagę na to, że pierwszy i drugi ciąg są ciągami różnymi. Pierwszy składa się z jednego, a drugi – z dwóch elementów. Aby unikać wątpliwości przy identyfikacji elementów ciągu, można stosować elementy rozdzielające – separatory, na przykład odstępy, jak wyżej, lub inne symbole, różne od elementów ciągu.

Uproszczony zapis ciągów pozwala na stwierdzenie, że zbiór ciągów długości $n \in Nat\backslash\{0\}$ nad zbiorem A można utożsamiać ze zbiorem wszystkich n-krotek nad zbiorem A, czyli z n-krotnym produktem kartezjańskim A^n nad zbiorem A. Oznacza to, że istnieje wzajemnie jednoznaczne odwzorowanie pomiędzy zbiorem ciągów o długości n a produktem kartezjańskim A^n, zatem

zbiorowi $CiągiSkończone_0(A)$ odpowiada zbiór $A^0 =_{def} \{<>\}$
zbiorowi $CiągiSkończone_n(A)$ odpowiada zbiór A^n dla $n \in Nat\backslash\{0\}$
zbiorowi $CiągiSkończone(A)$ odpowiada zbiór $\bigcup_{n \in Nat} A^n$

Ciągi zapisywane w uproszczonej postaci nazywa się *słowami*, a zbiór A nazywa się *alfabetem*. W dalszej części rozdziału będą używane właśnie te terminy.

Zbiór A^* jest zatem zbiorem wszystkich skończonych słów nad alfabetem A. Zbiór wszystkich niepustych skończonych słów nad alfabetem A będzie oznaczany przez A^+

$A^+ =_{def} A^*\backslash\{\varepsilon\}$

7.2. Operacje na słowach

Niech dany będzie dowolny, co najwyżej przeliczalny, alfabet A oraz niech A^* będzie zbiorem wszystkich skończonych słów nad A. Na słowach można definiować różne operacje. Podstawową jest operacja konkatenacji słów.

Niech $\alpha, \beta \in A^*$ będą dowolnymi słowami nad alfabetem A.

Konkatenacja słów α, β, co zapisuje się $\alpha \wedge \beta$, jest słowem, które powstaje przez dopisanie na końcu słowa α słowa β. Jeżeli

$\alpha = a_1 ... a_n$ oraz $\beta = b_1 ... b_m$

to

$\alpha \wedge \beta = a_1 ... a_n \wedge b_1 ... b_m =_{def} a_1 ... a_n\ b_1 ... b_m$

Niech $\alpha, \beta, \gamma \in A^*$. Konkatenacja słów ma następujące oczywiste własności:

$\varepsilon \wedge \varepsilon = \varepsilon$
$\varepsilon \wedge \alpha = \alpha \wedge \varepsilon = \alpha$
$(\alpha \wedge \beta) \wedge \gamma = \alpha \wedge (\beta \wedge \gamma)$

Ze względu na te własności nawiasy będą dalej opuszczane.

Słowo $\beta \in A^+$ jest *podsłowem* słowa $\alpha \in A^+$, gdy istnieją słowa $\gamma, \delta \in A^*$ takie, że

$\alpha = \gamma \wedge \beta \wedge \delta$

Jeżeli $\gamma \wedge \delta \neq \varepsilon$, to β jest *podsłowem właściwym* słowa α, jeżeli $\gamma = \varepsilon$, to β jest *podsłowem początkowym* słowa α, a jeżeli $\delta = \varepsilon$, to β jest *podsłowem końcowym* słowa α.

Konkatenacja jest podstawą do definiowania innych operacji na słowie:

- n-krotnej *iteracji* słowa,
- *czoła* słowa,
- *ogona* słowa,
- *inwersji* słowa.

Niech $\alpha = a_1 ... a_n$. Operacja n-krotnej *iteracji* słowa, dla $n \in Nat$, jest zdefiniowana rekursywnie następująco:

$\alpha^0 =_{def} \varepsilon$
$\alpha^{n+1} =_{def} \alpha^n \wedge \alpha$ dla $n \in Nat$

Operacje czoła *head* i ogona *tail*, określone następująco:

$head(\alpha) =_{def} a_1$ oraz $tail(\alpha) =_{def} a_2 \ldots a_n$

oznaczają odpowiednio pierwszy element słowa $\alpha = a_1 \ldots a_n$ oraz nowe słowo, które powstaje z α przez usunięcie jego pierwszego elementu. Oczywiście, dla dowolnego niepustego słowa zachodzi własność

$\alpha = head(\alpha) \wedge tail(\alpha)$

Operacja inwersji słowa $\alpha = a_1 \ldots a_n$, zapisywana w postaci α^{-1}, określona następująco:

$\alpha^{-1} =_{def} a_n a_{n-1} \ldots a_1$

oznacza lustrzane odbicie tego słowa.

Przykład 7.2

Niech $A = \{a, b, c\}$, wówczas słowami nad A są na przykład:

a aab $cabca$

Konkatenacją dwóch ostatnich słów jest słowo

$aab \wedge cabca = aabcabca$

Iteracjami słów są na przykład

$a^3 = aaa$
$(aab)^2 = aabaab$
$(cabca)^1 = cabca$

Operacje czoła i ogona dla dwóch pierwszych słów wyznaczają słowa:

$head(a) = a$ $tail(a) = \varepsilon$
$head(aab) = a$ $tail(aab) = ab$

Inwersjami słów są:

$a^{-1} = a$
$(aab)^{-1} = baa$
$(cabca)^{-1} = acbac$

Dla uproszczenia notacji, gdy nie będzie to wprowadzać niejednoznaczności, zamiast $\alpha \wedge \beta \wedge \gamma$ będzie się pisać $\alpha\beta\gamma$.

Dalej definiuje się jeszcze dwie operacje na słowach.

Najpierw wprowadza się pojęcie produkcji. Para słów $\beta, \gamma \in A^*$ zapisywana w postaci

$\beta ::= \gamma$

będzie nazywana *produkcją* lub *regułą przepisywania*.

Produkcję $\beta ::= \gamma$ można traktować tak samo jak uporządkowaną parę $<\beta, \gamma>$.

Symbol ::=, czytany: *jest zastępowany przez*, pełni rolę separatora oddzielającego dwa elementy. Słowo β po lewej stronie produkcji jest nazywane *poprzednikiem*, a słowo γ po prawej stronie produkcji jest nazywane *następnikiem* produkcji.

Niech $\alpha \in A^*$ oraz niech $\beta ::= \gamma$ będzie pewną produkcją.

Słowo δ jest wyprowadzeniem ze słowa α na podstawie produkcji $\beta ::= \gamma$, co zapisuje się w postaci

$$\alpha \xrightarrow{\beta ::= \gamma} \delta$$

gdy są spełnione warunki:

$\alpha = \alpha_1 \beta \alpha_2$
$\delta = \alpha_1 \gamma \alpha_2$

Przykład 7.3

Niech $A = \{a, b, c\}$. Ze słowa *aabcaa*, stosując produkcję $aa ::= cba$, można wyprowadzić słowa:

cbabcaa oraz *aabccba*

czyli

$$aabcaa \xrightarrow{aa::=cba} cbabcaa \quad \text{oraz} \quad aabcaa \xrightarrow{aa::=cba} aabccba$$

Jak pokazuje przykład, operacja wyprowadzenia nowego słowa δ ze słowa α na podstawie produkcji $\beta ::= \gamma$ nie musi być jednoznaczna. Liczba możliwych wyprowadzeń zależy od liczby wystąpień podsłowa β w słowie α. W szczególnym przypadku, gdy poprzednik reguły nie jest podsłowem w α, nie można wyprowadzić nowego słowa.

Niech $a ::= \beta$ będzie produkcją, w której poprzednik jest tylko pojedynczym symbolem $a \in A$ (słowem długości jeden), a następnik – jak poprzednio – jest dowolnym słowem $\beta \in A^*$ nad alfabetem A. Produkcja takiej postaci będzie nazywana *podstawieniem*.

Niech $\alpha \in A^*$ oraz niech $a ::= \beta$ będzie pewnym podstawieniem.

Słowo γ powstaje ze słowa α *przez podstawienie* $a ::= \beta$, co zapisuje się w postaci

$\alpha[a ::= \beta]$

gdy każde wystąpienie symbolu a w słowie α jest zastąpione słowem β.

Przykład 7.4

Niech $A = \{a, b, c\}$. W wyniku operacji określonej przez podstawienie $a ::= cba$ słowo $abcab$ zostanie przekształcone w słowo

$cbabccbab$

czyli

$abcab[a ::= cba] = cbabccbab$

Podobnie:

$abbacc[a ::= cbc] = cbcbbcbccc$
$abbacc[b ::= cbc] = acbccbcacc$
$abbacc[c ::= cbc] = abbacbccbc$

7.3. Języki formalne

Językiem formalnym L nad alfabetem A nazywa się dowolny podzbiór zbioru A^*, czyli $L \subseteq A^*$.

Język formalny jest tylko pewnym przybliżeniem języka naturalnego lub sztucznego, gdyż wyraża on tylko składniowy aspekt języka. W myśl wprowadzonej definicji alfabetem dla języka naturalnego jest zbiór słów w danym języku, a odpowiadający mu język formalny może być zbiorem wszystkich zdań w tym języku. W przypadku języka programowania alfabetem jest zbiór symboli leksykalnych, a odpowiadający mu język formalny definiuje zbiór wszystkich poprawnie tekstowo zbudowanych programów. Język formalny nie określa znaczenia i tym samym nie gwarantuje sensowności zdania czy programu, wyraża wyłącznie poprawność tekstową (składniową) zdania lub programu.

Przykład 7.5

Niech $A = \{a, b, c\}$, wówczas językami formalnymi nad A są na przykład skończone zbiory słów:

$\{a\}$, $\{aab, c\}$, $\{a, b, c, ab, cba\}$

Wykorzystując operację k-iteracji słów, dla $k \in Nat \setminus \{0\}$, można zdefiniować również pewne nieskończone języki formalne nad A, na przykład:

$\{s \in A^* \mid s = a^k \wedge b^l \wedge c^m \wedge k < l < m \wedge k, l, m \in Nat\}$
$\{a, b, c, ab, cba\} \cup \{s \in A^* \mid s = a^k \wedge b^{k+1} \wedge k \in Nat\}$

Niech A oraz B będą dwoma alfabetami. Funkcję $h : A^* \to B^*$ nazywa się *morfizmem* wtedy i tylko wtedy, gdy dla dowolnych $\alpha, \beta \in A^*$

$h(\alpha \wedge \beta) = h(\alpha) \wedge h(\beta)$

Morfizm nazywa się *izomorfizmem*, gdy h jest funkcją różnowartościową.

Przykład 7.6

Dla alfabetów $A = \{0, 1, 2, ..., 9\}$ oraz $B = \{0, 1\}$ izomorfizmem jest funkcja

$h(0) = 0000$, $h(1) = 0001$, ..., $h(9) = 1001$

wyrażająca kodowanie binarne liczb dziesiętnych.

Warto zwrócić uwagę, że alfabet przeliczalny A nie ma większej siły ekspresji niż dowolny alfabet skończony B. Oznacza to, że dla dowolnego języka formalnego $L_A \subseteq A^*$ istnieje taki język $L_B \subseteq B^*$, że istnieje wzajemnie jednoznaczne odwzorowanie pomiędzy obu językami $f : L_A \to L_B$.

Istotnie, niech będzie dany przeliczalny alfabet A o symbolach $a_1, a_2, a_3, ...$ oraz alfabet B zawierający tylko dwa symbole, na przykład 0, 1. Istnieje wzajemne odwzorowanie elementów alfabetu A w pewien podzbiór ciągów zero-jedynkowych nad alfabetem B. Na przykład ciągi binarne 1, 11, 111, ... itd. mogą być kodami indeksów kolejnych symboli $a_1, a_2, a_3, ...$ Dowolne słowo nad alfabetem A można przestawiać jako konkatenację odpowiednich ciągów kodujących nad alfabetem B. Na przykład słowo $a_3\ a_2\ a_2$ w alfabecie A będzie jednoznacznie reprezentowane przez słowo 111011011 w alfabecie B – symbol 0 pełni tu rolę separatora między kodami kolejnych symboli alfabetu A. Oznacza to, że dla dowolnego języka formalnego L_A nad A istnieje funkcja, która wzajemnie jednoznacznie odwzorowuje ten język w pewien język L_B nad B. Ciągi binarne mogą pełnić tę samą rolę, jaką pełnią symbole alfabetu A, co wyjaśnia powszechność stosowania kodowania binarnego.

Ponieważ języki formalne są zbiorami, można więc na nich wykonywać dowolne operacje mnogościowe. Jeżeli L_1, L_2 są językami nad alfabetem A, to także $L_1 \cup L_2$, $L_1 \cap L_2$ oraz L_1/L_2 są językami nad A.

Na językach można zdefiniować operację konkatenacji, która jest uogólnieniem konkatenacji zdefiniowanej na słowach. *Konkatenacja* języków $L_1 \subseteq A^*$, $L_2 \subseteq B^*$, oznaczana $L_1 \wedge L_2$, jest określona następująco:

$$L_1 \wedge L_2 =_{def} \{\alpha\beta \mid \alpha \in A^* \wedge \beta \in B^*\}$$

Korzystając z konkatenacji języków, wprowadza się operację *iteracji języków*, określoną dla dowolnego języka i liczby naturalnej n w sposób następujący:

$$L^0 =_{def} \{\varepsilon\}$$
$$L^{n+1} =_{def} L^n \wedge L \qquad \text{dla } n \in Nat$$

oraz operację *domknięcia* języka (nazywaną także gwiazdką Kleenego) określoną jako

$$L^* =_{def} \bigcup_{n \in Nat} L^n$$

Podobnie można uogólnić operacje *czoła*, *ogona* i *inwersji* dla języka $L \subseteq A^*$:

$$HEAD(L) =_{def} \{\alpha \in A^* \mid \exists \beta \in A^* \bullet \alpha\beta \in L\}$$
$$TAIL(L) =_{def} \{\beta \in A^* \mid \exists \alpha \in A^* \bullet \alpha\beta \in L\}$$
$$L^{-1} =_{def} \{\alpha \mid \alpha^{-1} \in L\}$$

7.4. Gramatyki bezkontekstowe

Nietrywialne języki formalne składają się z nieskończenie wielu słów. Nie można ich definiować enumeracyjnie, czyli przez jawne wyliczenie słów. Nieskończone języki formalne definiuje się rekursywnie, przy czym wykorzystuje się specyficzny mechanizm oparty na pojęciu *gramatyki* języka formalnego.

Gramatyka bezkontekstowa G jest czwórką

$$G =_{def} <T, N, P, S>$$

gdzie:

- T jest skończonym zbiorem, nazywanym *alfabetem symboli terminalnych*,
- N jest skończonym zbiorem, nazywanym *alfabetem symboli nieterminalnych*,
- P jest skończonym zbiorem *produkcji*,
- S jest wyróżnionym symbolem nieterminalnym, nazywanym *symbolem początkowym*.

Zakłada się, że zbiory symboli terminalnych i nieterminalnych są rozłączne, to jest

$$N \cap T = \varnothing$$

O pojedynczej produkcji $p \in P$ zakłada się, że jest postaci

$$v ::= \alpha$$

gdzie jej poprzednik v może być dowolnym symbolem nieterminalnym, czyli $v \in N$, a jej następnik α może być dowolnym niepustym słowem nad sumą mnogościową zbiorów symboli terminalnych i nieterminalnych, czyli $\alpha \in (T \cup N)^+$.

Gramatyka G generuje pewien język formalny $L(G) \subseteq T^*$. Nieformalnie jest to zbiór wszystkich słów nad alfabetem T, które można wyprowadzić z symbolu początkowego gramatyki S, za pomocą przekształceń, określonych przez zbiór P produkcji gramatyki.

Niech dane będą dwa słowa $\alpha, \beta \in (T \cup N)^+$.

Słowo β jest w gramatyce G *bezpośrednio wyprowadzane* ze słowa α, gdy istnieje taka produkcja $p \in P$, że

$$\alpha \xrightarrow{p} \beta$$

Fakt bezpośredniego wyprowadzenia słowa β ze słowa α w gramatyce G zapisuje się:

$$\alpha \xrightarrow[G]{} \beta$$

lub

$$\alpha \longrightarrow \beta$$

gdy z kontekstu wynika, o jaką gramatykę chodzi.

Słowo β jest w gramatyce G *wyprowadzone* ze słowa α, gdy istnieje skończony ciąg słów $\beta_1, \beta_2, ..., \beta_n \in (T \cup N)^+$ taki, że

$$\alpha = \beta_1 \qquad \beta_n = \beta$$

oraz

$$\beta_i \xrightarrow[G]{} \beta_{i+1} \qquad \text{dla } i \in \{1, 2, ..., n-1\}$$

Dalej symbol gramatyki G będzie pomijany.

Fakt, że słowo β jest *wyprowadzane* ze słowa α, zapisuje się w postaci

$$\alpha \xrightarrow{*} \beta$$

Językiem formalnym $L(G)$ generowanym przez gramatykę G jest zbiór

$$L(G) =_{def} \{\alpha \in T^* \mid S \xrightarrow{*} \alpha\}$$

Słowo $\alpha \in L(G)$ nazywa się też słowem *wywodliwym* w gramatyce G. Generowany przez gramatykę G język $L(G)$ jest zatem zbiorem wszystkich słów wywodliwych w G.

Poniżej rozpatrujemy przykłady gramatyk i wyprowadzenia słów, przy czym zapis produkcji jest oparty na powszechnie stosowanej tzw. *notacji BNF* (*Backus*[17] *Normal Form* lub *Backus-Naur Form*). Notacja ta wprowadza bardziej zwarty zapis produkcji, które mają tę samą lewą stronę. Zestaw produkcji, na przykład:

$$v ::= \alpha_1$$
$$.....$$
$$v ::= \alpha_m$$

zapisuje się w postaci

$$v ::= \alpha_1 \mid ... \mid \alpha_m$$

gdzie, jak poprzednio, $v \in N$ oraz $\alpha_1, ..., \alpha_m \in (T \cup N)^+$. Symbol | czyta się *lub*.

[17] John Backus (ur. 1924).

Przykład 7.7

Zbiór identyfikatorów tworzy pewien język formalny. Zbiór ten poprzednio był definiowany następująco:

Ident $=_{def}$ {*s* | *s jest niepustym ciągiem składającym się z liter lub cyfr, którego pierwszym elementem jest litera*}

Generująca zbiór identyfikatorów *Ident* gramatyka G_{ID} jest zdefiniowana następująco:

$G_{ID} =_{def} <T_{ID}, N_{ID}, P_{ID}, S_{ID}>$

gdzie:

$T_{ID} =_{def} \{a, b, ..., z\} \cup \{0, 1, ..., 9\}$
$N_{ID} =_{def}$ {*identyfikator, znak, litera, cyfra*}
$P_{ID} =_{def}$ {*identyfikator* ::= *litera* | *identyfikator znak*
znak ::= *litera*| *cyfra*
litera ::= *a* | *b* | ... | *z*
cyfra ::= 0 | 1 | ... | 9}
S_{ID} = *identyfikator*

Poszczególne produkcje w zbiorze są pisane w oddzielnych wierszach, bez oddzielania przecinkiem.

Rozpatruje się dwa przykłady wyprowadzenia konkretnych identyfikatorów. Pierwsze wyprowadzenie:

identyfikator $\xrightarrow{identyfikator::=litera}$ *litera*

litera $\xrightarrow{litera::=a}$ *a*

z symbolu początkowego *identyfikator* wyprowadza jednoelementowe słowo *a*.

Drugie z tego samego symbolu początkowego wyprowadza słowo *b*8:

identyfikator $\xrightarrow{identyfikator::=identyfikator\ znak}$ *identyfikator znak*

identyfikator znak $\xrightarrow{znak::=cyfra}$ *identyfikator cyfra*

identyfikator cyfra $\xrightarrow{identyfikator::=litera}$ *litera cyfra*

litera cyfra $\xrightarrow{litera::=b}$ *b cyfra*

b cyfra $\xrightarrow{cyfra::=8}$ *b*8

Pokazano zatem dwa wyprowadzenia:

identyfikator $\xrightarrow{*}$ *a*

identyfikator $\xrightarrow{*}$ *b*8

Oznacza to, że $a \in L(G_{ID})$ oraz $b8 \in L(G_{ID})$.

Przykład 7.8

Przykład pokazuje zbiór napisów reprezentujących liczby wymierne w zapisie dziesiętnym. Gramatyka G_{DEC} jest zdefiniowana następująco:

$G_{DEC} =_{def} <T_{DEC}, N_{DEC}, P_{DEC}, S_{DEC}>$

gdzie:

$T_{DEC} =_{def} \{0, 1, ..., 9\} \cup \{.\}$
$N_{DEC} =_{def}$ {*liczba, liczba_całkowita, kropka, cyfra*}
$P_{DEC} =_{def}$ {*liczba* ::= *liczba_całkowita* |
liczba_całkowita kropka liczba_całkowita
liczba_całkowita ::= *cyfra* | *liczba_całkowita cyfra*
kropka ::= .
cyfra ::= 0 | 1 | ... | 9}
S_{DEC} = *liczba*

Łatwo sprawdzić, że na przykład słowa 10.9 oraz 213 są wyprowadzalne w gramatyce G_{DEC}, natomiast słowo .01 nie jest wyprowadzalne w G_{DEC}.

Stosowane języki formalne, oprócz trywialnych przypadków, są zbiorami nieskończonymi i dlatego nie ma algorytmów generujących wszystkie słowa języka. Praktycznie rozwiązuje się dwa zadania.

Pierwsze jest zadaniem analizy – polega na zbadaniu, czy dane słowo jest elementem danego języka $L(G)$. Z tym zadaniem spotyka się podczas kompilacji programu. Celem pracy kompilatora jest w pierwszej kolejności stwierdzenie, czy program jest poprawny składniowo.

Drugie jest zadaniem generacji – polega na wygenerowaniu pewnego podzbioru słów języka, na przykład wszystkich słów o ustalonej długości.

7.5. Klasyfikacja gramatyk

Rozpatrzona gramatyka bezkontekstowa jest szczególnym przypadkiem szerszej klasy gramatyk, zwanych gramatykami struktur frazowych. *Gramatyka struktur frazowych* jest taką samą czwórką jak gramatyka bezkontekstowa, czyli

$G = <T, N, P, S>$

a różnica dotyczy tylko ogólniejszej postaci produkcji. Niech $V = T \cup N$. *Produkcja* albo *reguła przepisywania* $p \in P$ jest tu dowolną parą słów $\alpha \in V^+$ oraz $\beta \in V^*$ zapisywaną, jak poprzednio, w postaci $\alpha ::= \beta$. Generowanie języka formalnego przez gramatykę struktur frazowych jest definiowane, podobnie jak poprzednio, dla gramatyki bezkontekstowej.

Zgodnie z klasyfikacją wprowadzoną przez Chomsky'ego wyróżnia się cztery typy gramatyk struktur frazowych różniące się postacią dopuszczalnych produkcji.

Gramatyki klasy 0, zwane *gramatykami bez ograniczeń*, mają następującą postać produkcji

$\alpha ::= \beta$ dla $\alpha \in V^+, \beta \in V^*$

Gramatyki klasy 1., zwane *gramatykami kontekstowymi*, wymagają, by produkcje były postaci

$\alpha_1 \, v \, \alpha_2 ::= \alpha_1 \, \beta \, \alpha_2$ dla $\alpha_1, \alpha_2 \in V^*, v \in N, \beta \in V^+$

Gramatyki klasy 2., zwane *gramatykami bezkontekstowymi*, wymagają, by produkcje były postaci

$v ::= \beta$ dla $v \in N, \beta \in V^+$

Gramatyki klasy 3., zwane *gramatykami regularnymi*, wymagają, by produkcje były postaci (gramatyki *prawostronnie* regularne)

$v ::= \beta u$ dla $v \in N, u \in N \cup \{\varepsilon\}, \beta \in V^+$

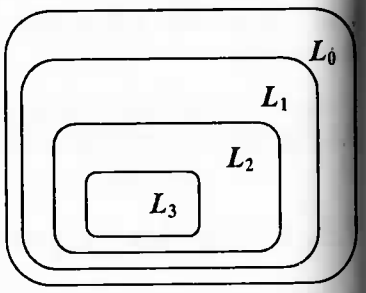
albo postaci (gramatyki *lewostronnie* regularne)

$v ::= u \beta$ dla $v \in N, u \in N \cup \{\varepsilon\}, \beta \in V^+$

Łatwo się przekonać, że każda produkcja gramatyki i jest jednocześnie produkcją gramatyki j, dla $0 \le j \le i \le 3$. Każdy zatem język formalny wygenerowany przez pewną gramatykę klasy i jest również generowany przez pewną gramatykę klasy j. Oznaczając symbolami L_0, L_1, L_2, L_3 zbiory języków formalnych generowanych przez gramatyki poszczególnych klas, można stwierdzić, że zachodzą inkluzje właściwe

$$L_3 \subset L_2 \subset L_1 \subset L_0$$

co oznacza, że wśród języków generowanych przez gramatyki klasy i istnieje co najmniej jeden język, który nie jest generowany przez gramatyki klasy j, dla $0 \le i \le j$ (rys. 7.1).



Rys. 7.1. Hierarchia Chomsky'ego języków formalnych

Gramatyki klas 1., 2. i 3. są *gramatykami nieskracającymi*, co oznacza, że długość nowego słowa nie jest mniejsza od długości starego słowa, do którego zastosowano produkcję gramatyki.

Nieskracalność gramatyki umożliwia efektywne badanie, czy dane słowo jest wywodliwe w gramatyce. Oznacza to, że można zbudować algorytm, który dla dowolnego słowa, po skończonej liczbie kroków, rozstrzyga, czy to słowo jest wyprowadzalne w danej gramatyce.

Schemat takiego algorytmu jest oczywisty. Niech α będzie badanym słowem. Najpierw generuje się zbiór Z_1 wszystkich słów bezpośrednio wyprowadzalnych z symbolu początkowego gramatyki, których długość nie przekracza długości badanego słowa α. Następnie generuje się zbiór Z_2 wszystkich słów, które są bezpośrednio wyprowadzalne ze słów zbioru Z_1, których długość nie przekracza długości badanego słowa α. Dalej generuje się zbiór słów Z_3, który jest bezpośrednio wyprowadzalny ze zbioru Z_2, itd. Każdy z wygenerowanych zbiorów jest oczywiście skończony. Postępowanie takie prowadzi się do momentu, gdy w zbiorze generowanych słów napotka się na słowo α albo gdy długość wszystkich słów należących do ostatniego zbioru będzie przekraczać długość słowa α. Pierwszy przypadek oznacza, że α należy do języka generowanego przez daną gramatykę, a drugi, że α nie należy do tego języka.

7.6. Drzewa rozbioru i diagramy składniowe

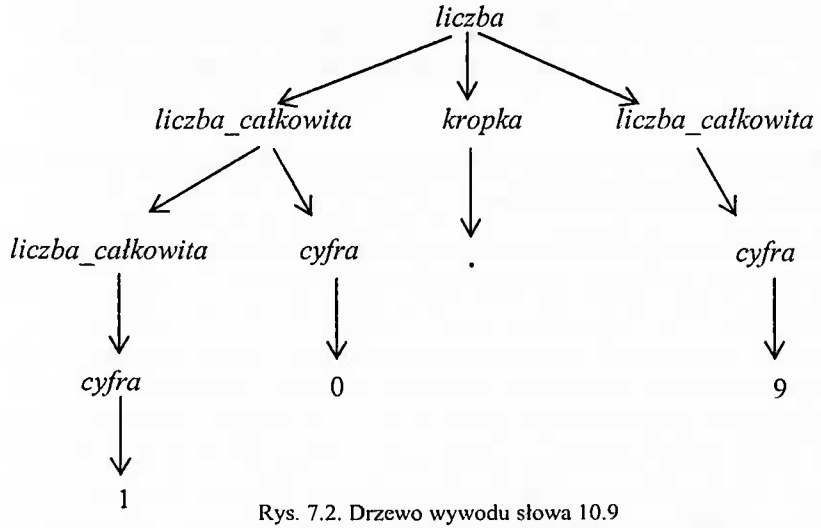
Dysponując pojęciem grafu, można zdefiniować drzewo wywodu – graf ilustrujący wyprowadzenia słowa w gramatyce, zwłaszcza w gramatyce bezkontekstowej.

Drzewem wywodu dla gramatyki $G =_{def} <T, N, P, S>$ jest graf-drzewo $T = <V, A>$, gdzie $A \subseteq V^2$, którego wierzchołki V są etykietowane symbolami ze zbioru $T \cup N$ w taki sposób, że:

- korzeń drzewa jest etykietowany symbolem początkowym S,
- każdy liść drzewa jest etykietowany symbolem terminalnym ze zbioru T,
- jeżeli węzeł v ma etykietę e i węzły $v_1, v_2, \ldots, v_n$ są jego następnikami, to znaczy $<v, v_1>, <v, v_2>, \ldots, <v, v_n> \in A$, o etykietach $e_1, e_2, \ldots, e_n$, to $e ::= e_1 \, e_2 \ldots e_n$ musi być produkcją gramatyki.

Przykład 7.9

Dla przedstawionej wcześniej gramatyki $G_{DEC} =_{def} <T_{DEC}, N_{DEC}, P_{DEC}, S_{DEC}>$ drzewo wywodu dla słowa 10.9 ma postać jak na rysunku 7.1.



Rys. 7.2. Drzewo wywodu słowa 10.9

Jeżeli dla pewnego słowa istnieją dwa różne drzewa wywodu, to gramatykę nazywa się *składniowo wieloznaczną*. Gramatyka G_{DEC} jest składniowo jednoznaczna, natomiast nie jest nią poniżej zdefiniowana gramatyka G_W.

Przykład 7.10

Niech

$G_W =_{def} <T_W, N_W, P_W, S_W>$

gdzie:

T_W = {*wyrażenie*, *składnik*, *czynnik*}

N_W = {*a*, *b*, *c*, +, –, *, /, (,)}

P_W = {*wyrażenie* ::= *składnik* | *składnik* + *wyrażenie* | *składnik* – *wyrażenie*
składnik ::= *czynnik* | *czynnik* * *czynnik* | *czynnik* | *czynnik*
czynnik ::= *a* | *b* | *c* | (*wyrażenie*)}

S_W = *wyrażenie*

W celu przekonania się o niejednoznaczności gramatyki G_W wystarczy rozpatrzyć możliwe wywody, na przykład słowa *a* + *b* – *c*.

Pojęcie drzewa rozbioru stanowi między innymi podstawę do określenia równoważności gramatyk.

Dwie gramatyki G_1 oraz G_2 są *słabo równoważne*, jeżeli generują te same języki, to znaczy $L(G_1) = L(G_2)$, oraz są *silnie równoważne*, jeżeli generują te same zbiory drzew rozbiorów.

Niech $T_1 = <V_1, A_1>$, $T_2 = <V_2, A_2>$ będą drzewami rozbioru oraz niech $h : V_1 \rightarrow V_2$.

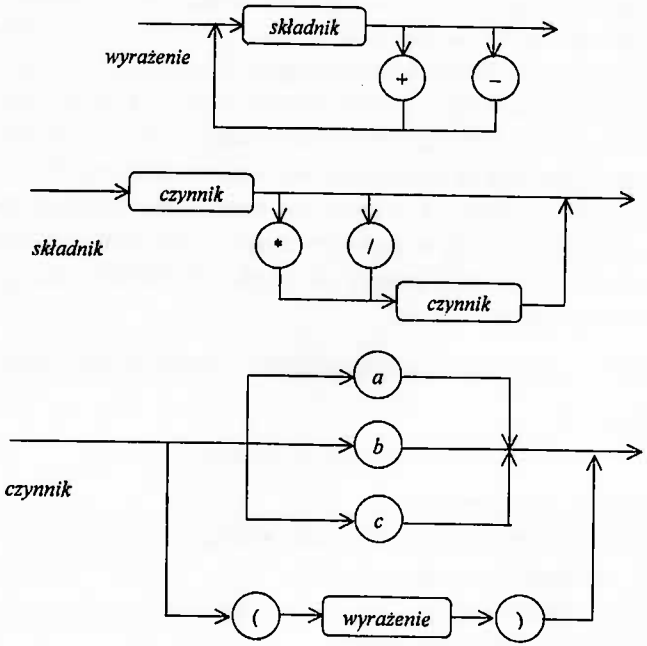
Funkcja *h zachowuje relację* wtedy i tylko wtedy, gdy dla dowolnych v, $v' \in V_1$, jeżeli $<v, v'> \in A_1^*$, to $<h(v), h(v')> \in A_2^*$, gdzie A_1^* i A_2^* są zwrotnymi, przechodnimi domknięciami relacji A_1 i A_2.

Jeżeli funkcja *h* zachowuje relację A_1 oraz dodatkowo jest bijekcją, to jest nazywana *izomorfizmem* drzewa T_1 w drzewo T_2.

Grafy są także wykorzystywane do prezentacji produkcji gramatyki w postaci diagramów składniowych. Wierzchołki tego grafu są etykietowane symbolami ze zbioru $T \cup N$. Każdej produkcji odpowiada pojedynczy graf z etykietowanymi wierzchołkami, który ma dokładnie jeden wierzchołek niemający poprzedników, zwany początkowym, i dokładnie jeden wierzchołek niemający następników, zwany końcowym. Wierzchołki te nie są etykietowane. Każdej ścieżce, która w grafie prowadzi od wierzchołka początkowego do wierzchołka końcowego, odpowiada pewien ciąg etykiet wierzchołków ze zbioru $T \cup N$. Ciąg etykiet stanowi ciąg symboli, które można wygenerować na podstawie danej produkcji.

Przykład 7.11

Produkcjom wcześniej zdefiniowanej gramatyki G_W odpowiadają następujące diagramy składniowe pokazane na rysunku 7.2.



Rys. 7.3. Diagramy składniowe gramatyki G_W

Pierwszy z diagramów, opisujący produkcję *wyrażenie*, jest grafem zawierającym cykl. Powodem pojawienia się cyklu jest to, że symbol *wyrażenie* występuje zarówno po lewej, jak i po prawej stronie produkcji, przy czym po prawej stronie występuje na końcu ciągu symboli.

Występowanie takiego samego symbolu po lewej i po prawej stronie produkcji, a tym samym istnienie cyklu na diagramie składniowym, można interpretować jako rekursywną definicję zbioru słów wyprowadzanych na podstawie danej produkcji.

7.7. Automaty i gramatyki

Gramatyki są mechanizmem generowania języków formalnych. Z gramatykami ściśle się wiąże pojęcie automatów skończonych jako mechanizmu służącego do rozpoznawania, czy dane słowa należą do danego języka formalnego. Przedstawiona dalej definicja automatu skończonego prezentuje tylko pewną klasę automatów skończonych,

służących do rozpoznawania słów należących do języków klas L_3 oraz L_2. Są to automaty Rabina–Scotta (*RS*) oraz automaty ze stosem.

Automat skończony jest modelem urządzenia, którego zadaniem jest rozpoznawanie czy słowa podawane na jego wejście są słowami danego języka formalnego. Automat działa krokowo; kolejne kroki są związane z analizą kolejnej litery słowa podawanego na jego wejście. Automat w każdym kroku swego działania jest w określonym stanie. Działanie rozpoczyna w ustalonym stanie początkowym, a kończy, gdy zostanie przeczytane całe badane słowo. Podczas kolejnych kroków automat przechodzi pomiędzy stanami, co następuje na skutek odczytania na wejściu kolejnej litery analizowanego słowa. Po zakończenie działania, to znaczy po odczytaniu ostatniej litery analizowanego słowa, automat znajdzie się w pewnym stanie. Jeżeli jest to jeden z jego stanów końcowych, oznacza to, że słowo należy do języka formalnego akceptowanego (rozpoznawanego) przez automat.

Formalnie skończony *automat akceptujący RS* jest określony jako piątka

$$A = <S, X, \delta, s_0, F>$$

gdzie:

S jest skończonym zbiorem stanów,
X jest alfabetem – zbiorem symboli wejściowych,
$\delta: S \times X \rightarrow S$ jest funkcją zmiany stanów,
s_0 jest stanem początkowym,
$F \subseteq S$ jest zbiorem stanów końcowych.

Działanie automatu przy analizie słowa wejściowego $x_1, x_2, ..., x_n$, dla $n > 0$, polega na wykonywaniu ciągu kroków, które określają ciąg stanów:

$$s_0, s_1, ..., s_n$$

taki, że

$$s_{k+1} = \delta(s_k, x_k) \qquad \text{dla } k = 1, ..., n$$

Słowo $x_1, x_2, ..., x_n$ jest akceptowane przez automat, gdy $s_k \in F$, i nie jest akceptowane w przypadku przeciwnym. Zbiór wszystkich akceptowanych słów, oznaczany $L(RS)$, stanowi język formalny akceptowany przez automat *RS*.

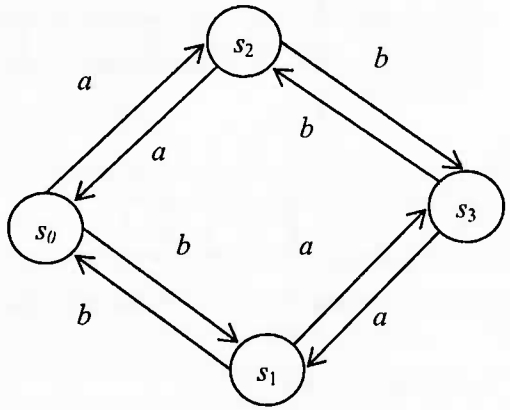
Przykład 7.12

Automat *RS* postaci $<S, X, \delta, s_0, F>$, gdzie:

$S = \{s_0, s_1, s_2, s_3\}$
$X = \{a, b\}$
$F = \{s_0\}$
$\delta(s_0, a) = s_2$, $\delta(s_0, b) = s_1$
$\delta(s_1, a) = s_3$, $\delta(s_1, b) = s_0$

$\delta(s_2, a) = s_0$, $\delta(s_2, b) = s_3$
$\delta(s_3, a) = s_1$, $\delta(s_3, b) = s_2$

można przedstawić graficznie w postaci pokazanej na rysunku 7.4.



Rys. 7.4. Graf przejść stanów automatu

Można też łatwo sprawdzić, że językiem formalnym akceptowanym przez automat jest $L \subseteq \{a, b\}^*$ taki, że

$$L = \{\alpha \mid len(\alpha|_a) \text{ jest parzysta oraz } len(\alpha|_b) \text{ jest parzysta}\},$$

gdzie: $len(\alpha)$ oznacza długość ciągu α, a $\alpha|_a$ oraz $\alpha|_b$ oznaczają podciągi ciągu α złożone wyłącznie z symboli a oraz b.

Pomiędzy skończonymi automatami *RS* a gramatykami zachodzą następujące związki:

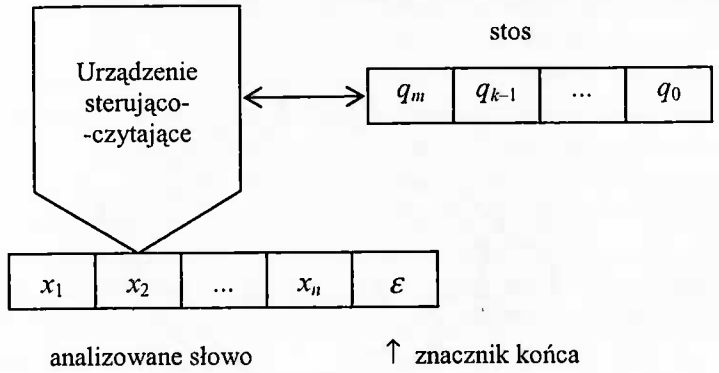
- Dla każdego języka formalnego klasy L_3 istnieje skończony automat *RS* akceptujący ten język, to znaczy każde skończone obliczenie automatu kończy się osiągnięciem stanu końcowego.
- Każdy język akceptowany przez skończony automat *RS* jest językiem klasy L_3.

Języki klasy L_3 nazywa się językami *regularnymi*.

Rozpoznawanie języków klasy L_2, czyli języków *bezkontekstowych*, może być również realizowane przez automaty, z tym że są to bardziej złożone automaty, nazywane automatami ze stosem (rys. 7.5).

Stos jest pewnego rodzaju pamięcią, w której przechowuje się ciągi symboli z danego repertuaru symboli. Na stosie można wykonywać dwie zasadnicze operacje: dopisywania symbolu do stosu lub odczytywania i zdjęcia elementu ze stosu. Jeżeli zawartością stosu jest ciąg symboli $q_1\ q_2\ ...\ q_k$, to dopisanie symbolu q' polega na dołączeniu go na początku ciągu, czyli zmieni jego zawartość na $q'q_1\ q_2\ ...\ q_k$, zdjęcie natomiast

elementu ze stosu polega na usunięciu elementu na początku ciągu, czyli zmieni jego zawartość na $q_2 \ldots q_k$.



Rys. 7.5. Automat ze stosem

Automat ze stosem AS jest definiowany jako siódemka

$$AS = <S, X, Q, \delta, s_0, q_0, F>$$

gdzie:

- S jest skończonym zbiorem stanów,
- X jest alfabetem – zbiorem symboli wejściowych,
- Q jest skończonym zbiorem symboli składowanych na stosie,
- $\delta: S \times (X \cup \{\varepsilon\}) \times Q \rightarrow \wp_{fin}(S \times Q^*)$ jest funkcją zmiany stanu i zawartości stosu ($\wp_{fin}(Z)$ oznacza rodzinę wszystkich skończonych podzbiorów zbioru Z),
- s_0 jest stanem początkowym,
- q_0 jest symbolem początkowym znajdującym się na stosie,
- $F \subseteq S$ jest zbiorem stanów końcowych.

Jeżeli zbiór $\delta(s, x, q)$ dla każdego $s \in S$, $x \in X$, $q \in Q$ zawiera co najwyżej jeden element, to automat AS jest automatem deterministycznym, w przypadku przeciwnym – jest automatem niedeterministycznym.

Działanie automatu AS przebiega krokowo, a w każdym kroku następuje zmiana konfiguracji automatu. Konfiguracjami automatu są trójki:

$$<s, \omega, \alpha> \in S \times X^* \times Q^*$$

gdzie:

- s jest aktualnym stanem urządzenia sterująco-czytającego,
- ω jest częścią analizowanego słowa, nieprzeczytaną jeszcze przez głowicę czytającą; pierwszy symbol ciągu ω znajduje się pod głowicą czytającą; jeżeli symbolem tym jest ε (słowo puste) oznacza to, że całe słowo zostało już przeczytane,
- α jest aktualną zawartością stosu.

Krok działania automatu AS polega na przejściu z konfiguracji do konfiguracji w wyniku przeczytania pojedynczego symbolu analizowanego słowa. Przejście takie będzie opisywane relacją $\longrightarrow$ i zapisywane w postaci

$$<s, x\omega, q\alpha> \longrightarrow <s', \omega, \beta\alpha>$$

jeżeli zbiór $\delta(s, x, q)$ zawiera parę $<s', \beta>$, gdzie $s, s' \in S$, $x \in X \cup \{\varepsilon\}$, $\omega \in X^*$, $q \in Q$ oraz $\alpha, \beta \in Q^*$.

Jeżeli $x \neq \varepsilon$, to automat AS jest w stanie s, x jest symbolem znajdującym się pod głowicą czytającą, q jest symbolem będącym się na szczycie stosu. Automat przechodzi do nowego stanu s', przesuwa głowicę czytającą o jedną pozycję w prawo i zamienia symbol na szczycie stosu ciągiem β złożonym z elementów zbioru symboli Q. Jeżeli $\beta = \varepsilon$, to usuwa się element ze szczytu stosu, skracając tym samym jego zawartość.

Jeżeli $x = \varepsilon$, to oznacza, że całe słowo zostało przeczytane. W kroku tym, nazywanym ε-krokiem, automat AS nie przesuwa głowicy czytającej, jednak stan automatu i zawartość stosu mogą się zmieniać.

Początkową konfiguracją automatu AS jest $<s_0, \omega, q_0>$, co oznacza, że automat znajduje się w stanie początkowym $s_0 \in S$, pod głowicą czytającą znajduje się pierwszy symbol analizowanego słowa $\omega \in X^*$, a zawartością stosu jest tylko jeden początkowy symbol $q_0 \in Q$. Konfiguracją końcową jest konfiguracja postaci $<s, \varepsilon, \varepsilon>$, gdzie $s \in F$.

Słowo jest akceptowane przez automat AS, jeżeli

$$<s_0, \omega, q_0> \xrightarrow{*} <s, \varepsilon, \varepsilon>,$$

gdzie $\xrightarrow{*}$ jest zwrotnym i przechodnim domknięciem relacji $\longrightarrow$.

Zbiór wszystkich akceptowanych słów, oznaczany $L(AS)$, stanowi język formalny akceptowany przez automat AS.

Przykład 7.13

Automat AS postaci $<S, X, Q, \delta, s_0, q_0, F>$, gdzie:

$S = \{s_0, s_1, s_2\}$
$X = \{0, 1\}$
$Q = \{q_0, 0\}$
$F = \{s_0\}$
$\delta(s_0, 0, q_0) = \{<s_1, 0q_0>\}$

$\delta(s_1, 0, 0) = \{<s_1, 00>\}$
$\delta(s_1, 1, 0) = \{<s_2, \varepsilon>\}$
$\delta(s_2, 1, 0) = \{<s_2, \varepsilon>\}$
$\delta(s_2, \varepsilon, q_0) = \{<s_0, \varepsilon>\}$

Zilustrujmy działanie automatu podczas analizy konkretnego słowa 0011:

$$<s_0, 0011, q_0> \longrightarrow <s_1, 011, 0q_0>$$
$$\longrightarrow <s_1, 11, 00q_0>$$
$$\longrightarrow <s_2, 1, 0q_0>$$
$$\longrightarrow <s_2, \varepsilon, q_0>$$
$$\longrightarrow <s_0, \varepsilon, \varepsilon>$$

Ogólnie, można pokazać, że:

$$<s_0, 0, q_0> \longrightarrow <s_1, \varepsilon, 0q_0>$$
$$<s_1, 0^i, 0q_0> \xrightarrow{i} <s_1, \varepsilon, 0^{i+1}q_0>$$
$$<s_1, 1, 0^{i+1}q_0> \longrightarrow <s_2, \varepsilon, 0^iq_0>$$
$$<s_2, 1^i, 0^iq_0> \xrightarrow{i} <s_2, \varepsilon, q_0>$$
$$<s_2, \varepsilon, q_0> \longrightarrow <s_0, \varepsilon, \varepsilon>$$

Na tej podstawie można pokazać, że dla $n \geq 1$ zachodzi

$$<s_0, 0^n1^n, q_0> \xrightarrow{2n+1} <s_0, \varepsilon, \varepsilon>$$

oraz

$$<s_0, \varepsilon, q_0> \xrightarrow{0} <s_0, \varepsilon, q_0>$$

co oznacza, że zbiór akceptowanych słów zawiera się w języku $L = \{0^n1^n \mid n \geq 0\}$. Pomijamy tu pokazanie faktu, że automat akceptuje słowa tylko z tego języka.

Rozpoznawanie języków klas $\boldsymbol{L}_1$ oraz $\boldsymbol{L}_0$ jest jeszcze bardziej złożone. Pomijając szczegóły, ograniczymy się tu do stwierdzenia, że do rozpoznawania języka dowolnej klasy, a więc również klasy $\boldsymbol{L}_0$, można zbudować odpowiednią maszynę Turinga.

Uwaga

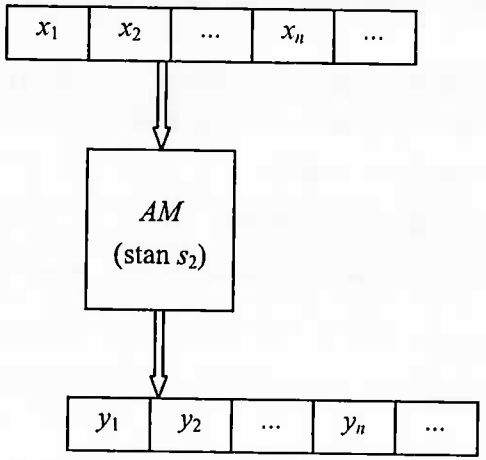
Bardziej szczegółowe informacje o językach formalnych, gramatykach i automatach skończonych można znaleźć na przykład w książce [Hopcroft, Ullman 2003].

Pojęcie automatu skończonego jest często używane w różnych obszarach informatyki, zwłaszcza w projektowaniu układów cyfrowych z pamięcią (w odróżnieniu od układów cyfrowych bez pamięci – bramek logicznych, zob. rozdz. 9.). Najczęściej spotykanymi tam pojęciami są skończone automaty Moore'a i automaty Mealy'ego. Oba automaty są modelami „czarnej skrzynki" z jednym wejściem i jednym wyjściem (rys. 7.6). Na podstawie ciągu symboli czytanych na wejściu automat podaje na swoje wyjście inny ciąg symboli; ciąg wyjściowy jest wynikiem przetworzenia ciągu wejściowego. Ponieważ automaty te można stosować zamiennie, poniżej podaje się tylko definicję automatu Moore'a *AM*:

$AM = <X, Y, S, \delta, \lambda, s_0>$

gdzie:

X jest skończonym zbiorem symboli wejściowych,
Y jest skończonym zbiorem symboli wejściowych,
S jest skończonym zbiorem stanów,
$\delta: X \times S \rightarrow S$ jest funkcją przejść,
$\lambda: S \rightarrow Y$ jest funkcją wyjść,
s_0 jest stanem początkowym.



Rys. 7.6. Schemat automatu Moore'a

Automat *AM* przetwarza ciągi $x_1, x_2, ..., x_n, ...$ symboli alfabetu wejściowego X w ciągi $y_1, y_2, ..., y_n, ...$ symboli alfabetu Y. Przetworzenie to jest określone następująco: Niech

$s_0, s_1, s_2, ..., s_n, ...$

będzie ciągiem stanów takim, że

$s_{k+1} = \delta(x_k, s_k)$, dla $k = 1, 2, ..., n, ...$

wówczas $y_k = \lambda(s_k)$.

Ćwiczenia

1. Niech $A =_{def} \{+, =\}$. Które z poniższych zdań są prawdziwe?
   a) Można utworzyć co najwyżej skończoną liczbę języków formalnych nad alfabetem A.
   b) Można utworzyć dokładnie 4 słowa nad alfabetem A.
   c) Zbiór { ++, +++, =, += } jest pewnym językiem formalnym nad A.
   d) Nad alfabetem A można utworzyć dokładnie 2^4 języków formalnych.
   e) Zbiór wszystkich słów nad alfabetem A definiuje pewien nowy alfabet A'.
2. Niech $A =_{def} \{0, 1\}$, $B =_{def} \{0, 1, 2, ..., n\}$ oraz $C = Nat$. Które pary spośród zbiorów A^*, B^*, C^* są zbiorami równolicznymi?
3. Czy zbiór liczb naturalnych Nat jest równoliczny ze zbiorem Nat^* – zbiorem wszystkich skończonych ciągów nad Nat?
4. Czy zbiór wszystkich języków formalnych nad przeliczalnym alfabetem A jest przeliczalny?
5. Niech L będzie językiem formalnym nad alfabetem $\{0, 1\}$. Dwa słowa $u, v \in \{0, 1\}^*$ są równoważne względem języka L, co oznacza się $u \sim_L v$, jeżeli

   $$\forall x \in \{0, 1\}^* \bullet (u \wedge x \in L \Leftrightarrow v \wedge x \in L)$$

   Pokazać, że $\sim_L$ jest relacją równoważności.
6. Przedstawić klasy abstrakcji relacji równoważności słów $\sim_L$ względem następujących języków:
   a) $L_1 = \{1^n \mid 1 \le n \le 6\}$
   b) $L_2 = \{0^n 1^n \mid n \in Nat\}$
   c) $L_3 = (0011)$
7. Pokazać, że jeśli dla pewnych słów $u \in L$ oraz $v \in \{0, 1\}^*$ zachodzi $u \wedge v \in [u]_{\sim L}$, gdzie $\sim_L$ jest relacją równoważności słów względem języka L, to $u \wedge v^n \in L$ dla dowolnego $n \in Nat$.
8. Dana jest gramatyka $G = <T, N, P, S>$, gdzie:

   $T =_{def} \{A, B, C\}$
   $N =_{def} \{a, b, c\}$
   $P =_{def} \{a ::= A \mid aA \mid bC$
   $\quad b ::= BcC$
   $\quad c ::= abC \mid ABc \mid AbC \}$
   $S = a$

   Czy słowa $AAAA$, $ABCA$ należą do języka generowanego przez G? Podać zbiór wszystkich słów długości 1, 2 i 3 należących do języka generowanego przez G. Scharakteryzować zbiór wszystkich słów generowanych przez gramatykę G. Czy można zdefiniować „prostszą" gramatykę, która generuje taki sam język formalny jak gramatyka G?
9. Przedstawić drzewa rozbioru składniowego dla wszystkich słów długości 4 generowanych przez gramatykę z zadania 8.
10. Czy jednoznaczne są gramatyki:
    a) gramatyka z zadania 8.,
    b) $G = <\{a, +, *\}, \{S\}, \{S ::= S + S \mid S * S \mid a\}, S>$.
11. Zdefiniować gramatykę generującą język $L = \{a^n b^m c^m \mid m, n \ge 1\}$. Zbadać, czy gramatyka jest jednoznaczna.
12. Zdefiniować gramatykę generującą następujące zbiory:
    a) zbiór wszystkich palindromów (słów, które można odczytywać zarówno w przód, jak i wspak) nad alfabetem $\{a, b\}$,
    b) zbiór wszystkich słów nad alfabetem $\{a, b\}$ zawierających dokładnie dwa razy więcej symboli a niż symboli b,
    c) $L = \{a^i b^j c^k \mid i \ne j \text{ lub } j \ne k\}$.
13. Dana jest gramatyka $G = <\{a, b\}, \{S\}, \{S ::= aS \mid aSbS \mid \varepsilon\}, S>$. Udowodnić, że $L(G) = \{x \mid$ każdy przedrostek x ma co najmniej tyle symboli a, co symboli $b\}$
14. Dla znanego języka programowania, na przykład Pascal, C, C++, zdefiniować gramatykę określającą wybrany podzbiór wyrażeń arytmetycznych z tego języka.
15. Przedstawić w postaci diagramów składniowych produkcje gramatyk zdefiniowanych w zadaniach 5., 7. i 8.
16. Symbol nieterminalny $A \in N$ gramatyki jest zbędny, gdy jest nieaktywny lub nieosiągalny. Symbol $A \in N$ jest nieaktywny, gdy język generowany przez gramatykę $G_A = <T, N, P, A>$ jest pusty. $A \in N$ jest nieosiągalny, gdy nie występuje w żadnym słowie wyprowadzanym w gramatyce G. Wskazać zbędne symbole nieterminalne w gramatyce $G = <T, N, P, S>$, gdzie:

    $T =_{def} \{A, B, C\}$
    $N =_{def} \{a, b, c, d\}$
    $P =_{def} \{a ::= A \mid aA \mid bC \mid AcA$
    $\quad b ::= BcC \mid cAc$
    $\quad c ::= abC \mid ABc \mid AbC$
    $\quad d ::= aA \mid dbC\}$
    $S = a$

17. Gramatykę $G = <T, N, P, S>$, gdzie:

$T =_{def} \{A, B\}$
$N =_{def} \{a, b, c\}$
$P =_{def} \{c ::= a \mid b$
$a ::= Ab \mid Bc \mid B$
$b ::= ab \mid bA \mid ac \mid B\}$
$S = c$

przekształcić do słabo równoważnej gramatyki bezkontekstowej, niezawierającej zbędnych symboli.

18. Zdefiniować automat z pamięcią stosową, rozpoznający słowa języka

$L = \{\alpha\alpha^{-1} \mid \alpha \in \{a, b\}^+\}$

8. Algebry abstrakcyjne

8.1. Algebry jednorodzajowe

Jednorodzajową algebrą abstrakcyjną, albo krótko – algebrą, nazywa się parę

$ALG =_{def} <A, \{c_1, ..., c_m\} \cup \{f_1, ..., f_n\}>$ dla $m \in Nat, n \in Nat \setminus \{0\}$

w której:

A jest dowolnym zbiorem, zwanym *nośnikiem* algebry,

c_i są *stałymi* algebry, to znaczy $c_i \in A$, dla $i = 1, ..., m$,

f_j są *operacjami* albo *działaniami* algebry, to znaczy k-argumentowymi funkcjami o sygnaturze

$f_j: A^k \rightarrow A$

gdzie: $k \in Nat \setminus \{0\}, j = 1, ..., n$.

Stałą algebry c_i można także rozumieć jako funkcję zeroargumentową, to znaczy jako funkcję o sygnaturze

$c_i : \rightarrow A$.

Uwaga

Algebry będą też definiowane jako pary:

$ALG =_{def} <A, \{c_1, ..., c_m, f_1, ..., f_n\}>$

lub

$ALG =_{def} <A, F>$

gdzie drugim elementem pary jest zbiór stanowiący sumę mnogościową zbioru stałych i operacji. Wynika to z faktu, że stałe można traktować jako funkcje zeroargumentowe.

Przykład 8.1

Przykładem prostej algebry jest zbiór liczb naturalnych *Nat* z operacją dodawania

$ALG_{Nat} =_{def} <Nat, \{0, 1\} \cup \{\_+\_\}>$

gdzie:

$0, 1 : \rightarrow Nat$

są operacjami zeroargumentowymi, a

$\_+\_ : Nat^2 \rightarrow Nat$

jest dodawaniem.

Należy przypomnieć, że podkreślenia obok symbolu funkcji wskazują położenie argumentów.

Przykład 8.2

Bardziej złożona jest algebra określona na zbiorze liczb całkowitych *Całkowite*, z operacjami dodawania, odejmowania i mnożenia

$ALG_{Całkowite} =_{def} <Całkowite, \{0, 1\} \cup \{ \_+\_, \_-\_, \_*\_ \}>$

gdzie: 0 oraz 1 są stałymi, czyli mają sygnatury:

$0, 1 : \rightarrow Całkowite$

a +, –, * są symbolami operacji dwuargumentowych o sygnaturach:

$\_+\_, \_-\_, * : Całkowite^2 \rightarrow Całkowite$

Należy zauważyć, że odejmowanie może być również traktowane jako zmiana znaku liczby. W tym przypadku symbol byłby symbolem przeciążonym, a odpowiadająca mu sygnatura miałaby postać

$-\_ : Całkowite \rightarrow Całkowite$

Algebra z tak dołączoną operacją miałaby natomiast postać

$ALG_{Całkowite} =_{def} < Całkowite, \{0, 1\} \cup \{-\_, \_+\_, \_-\_, \_*\_ \}>$

W dalszych przykładach dla symboli funkcyjnych dwuargumentowych będzie stosowana notacja wrostkowa, a podkreślenia w napisach określających sygnaturę będą pomijane.

Przykład 8.3

Algebra słów nad pewnym alfabetem A jest zdefiniowana

$ALG_{A^*} =_{def} <A^*, \{\varepsilon\} \cup \{\hat{}\} >$

gdzie:
A^* jest nośnikiem algebry,
$\varepsilon : \rightarrow A^*$ jest słowem pustym, czyli stałą algebry,
$\hat{} : A^* \times A^* \rightarrow A^*$ jest konkatenacją, czyli dwuargumentowym działaniem.

Przykład 8.4

W programowaniu przez typ danych rozumie się pewien zbiór wartości i zestaw związanych z nim operacji. Powszechnie używany typ logiczny jest określony przez zbiór

$Boolean =_{def} \{false, true\}$

oraz przez zestaw operacji o następujących sygnaturach:

$not : Boolean \rightarrow Boolean$

$and, or : Boolean^2 \rightarrow Boolean$

gdzie: *not*, *and* oraz *or* są operacjami *negacji*, *koniunkcji* i *dysjunkcji*. Definicję tych funkcji przedstawia tablica:

a	*b*	*not(a)*	*a and b*	*a or b*
false	*false*	*true*	*false*	*false*
false	*true*	*true*	*false*	*true*
true	*false*	*false*	*false*	*true*
true	*true*	*false*	*true*	*true*

Zdefiniowane funkcje warto porównać z definicją spójników logicznych określonych w podrozdziale 1.2. Różnica między tymi definicjami sprowadza się do różnicy symboli.

Algebra, która jest modelem typu logicznego, ma zatem postać

$ALG_{Boolean} =_{def} <Boolean, \{not, and, or\}>$

Zbiór stałych jest tutaj pusty.

Przykład 8.5

W językach programowania odpowiednikiem wcześniej przedstawianej algebry, określonej na zbiorze liczb całkowitych *Całkowite*, jest algebra określona na zbiorze *Integer* z odpowiednikami operacji dodawania, odejmowania i mnożenia:

$ALG_{Integer} =_{def} < Integer, \{0, 1\} \cup \{\oplus,\ominus,\otimes\}>$

gdzie:

$Integer =_{def} \{-N, ..., N\}$ jest tylko skończonym podzbiorem zbioru *Całkowite*, zakłada się przy tym, że $N \in Nat \setminus \{0, 1\}$,

0 oraz 1 są stałymi o sygnaturach:

$0, 1 : \rightarrow Integer$

a $\oplus,\ominus,\otimes$ są – odpowiednio – symbolami dodawania, odejmowania i mnożenia o sygnaturach:

$\oplus,\ominus,\otimes: Integer^2 \rightarrow Integer$

Istotna różnica między algebrą $ALG_{Integer}$ a $ALG_{Całkowite}$ wypływa z definicji operacji w algebrze $ALG_{Integer}$. Ze względu mianowicie na ograniczoność zbioru *Integer* operacje dodawania, odejmowania i mnożenia nie są funkcjami całkowicie określonymi. Aby odróżnić je od operacji określonych w zbiorze liczb *Całkowite*, są one zapisywane inaczej. Definicje operacji algebry $ALG_{Integer}$ przedstawiają się następująco:

$$a \oplus b =_{def} \begin{cases} a+b \ \text{ gdy } |a+b| \le N \\ \text{nieokreślona w przypadku przeciwnym} \end{cases}$$

$$a \ominus b =_{def} \begin{cases} a-b \ \text{ gdy } |a-b| \le N \\ \text{nieokreślona w przypadku przeciwnym} \end{cases}$$

$$a \otimes b =_{def} \begin{cases} a*b \ \text{ gdy } |a*b| \le N \\ \text{nieokreślona w przypadku przeciwnym} \end{cases}$$

Symbole występujące po prawej stronie definicji, czyli funkcje dodawania, odejmowania, mnożenia i wartości bezwzględnej, są określone na zbiorze liczb całkowitych i należą do języka arytmetyki. Bez znajomości tych funkcji nie można zrozumieć definicji nowych operacji.

Częściowa określoność operacji algebry $ALG_{Integer}$ ma interpretację praktyczną. Brak określoności operacji oznacza możliwość powstania nadmiaru podczas wykonywania programu. Powstanie nadmiaru podczas obliczenia programu prowadzi do wygenerowania wyjątku lub do zerwania obliczeń z sygnalizacją błędu.

Algebry, które mają operacje określone częściowo, mogą być uciążliwe w zastosowaniach, dlatego – zwłaszcza w programowaniu – dokonuje się pewnej modyfikacji takich algebr tak, aby uzyskać całkowitą określoność ich operacji. Sposób tej modyfikacji wyjaśnia przykład algebry $ALG_{Integer}$.

Definiowaną w przykładzie algebrę $ALG_{Integer\cup\{nadmiar\}}$ można traktować jako algebraiczny model całkowitoliczbowego typu danych występującego w językach programowania.

Przykład 8.6

Niech dana będzie algebra

$$ALG_{Integer\cup\{nadmiar\}} =_{def} <Integer \cup \{nadmiar\},\ \{0, 1\} \cup \{\oplus,\ominus,\otimes\}>$$

Nośnikiem algebry jest zbiór *Integer* z dołączonym elementem *nadmiar*. Operacjami algebry są operacje dodawania, odejmowania i mnożenia, oznaczane – jak

poprzednio – symbolami: $\oplus,\ominus,\otimes$. Operacje te mają inne sygnatury:

$\oplus,\ominus,\otimes : (Integer \cup \{nadmiar\})^2 \rightarrow Integer \cup \{nadmiar\}$

Wszystkie operacje arytmetyczne mają wspólną własność:

Jeżeli wartością któregokolwiek argumentu operacji jest *nadmiar*, to wynikiem operacji jest również *nadmiar*, na przykład *nadmiar* $\oplus 1$ = *nadmiar*.

W pozostałych przypadkach, gdy oba argumenty operacji są różne od *nadmiar*, definicje operacji są następujące:

$$a \oplus b =_{def} \begin{cases} a+b \ \text{ gdy } |a+b| \le N \\ nadmiar \text{ w przypadku przeciwnym} \end{cases}$$

$$a \ominus b =_{def} \begin{cases} a-b \ \text{ gdy } |a-b| \le N \\ nadmiar \text{ w przypadku przeciwnym} \end{cases}$$

$$a \otimes b =_{def} \begin{cases} a*b \ \text{ gdy } |a*b| \le N \\ nadmiar \text{ w przypadku przeciwnym} \end{cases}$$

Uwaga

Symbol *nadmiar* w powyższym przykładzie jest odpowiednikiem symbolu $\perp$, wprowadzonego w podrozdziale 3.7 na oznaczenie nieokreśloności funkcji. Symbol *nadmiar* odnosi się również do sytuacji, gdy funkcja jest nieokreślona, a ponadto wskazuje na przyczynę nieokreśloności.

8.2. Algebry wielorodzajowe

Uogólnieniem algebr jednorodzajowych są *algebry wielorodzajowe* [Ehrig, Mahr 1985]. Uogólnienie polega na zastąpieniu pojedynczego nośnika skończoną rodziną nośników. Algebra wielorodzajowa jest zdefiniowana jako układ

$$ALG =_{def} <\{A_1, ..., A_k\},\ \{c_1, ..., c_m\} \cup \{f_1, ..., f_n\}> \qquad \text{dla } m \in Nat \text{ oraz } k, n \in Nat \setminus \{0\}$$

gdzie:

$A_1, ..., A_k$ są dowolnymi zbiorami, nazywanymi *nośnikami* algebry,

c_i jest *stałą* algebry, to znaczy $c_i \in A_{j_i}$, dla $i = 1, ..., m, j_i \in \{1, ..., k\}$,

f_j jest *operacją* algebry, dla $j = 1, ..., n$, to znaczy jest k_j-argumentową funkcją, $k_j \in Nat \setminus \{0\}$, o sygnaturze:

$$f_j : A_{j_1} \times ... \times A_{j_{k_j}} \rightarrow A_{j_{k_0}}$$

gdzie:

$j_1, ..., j_{k_j} \in \{1, ..., k\}$ są indeksami nośników, które są argumentami operacji,

j_{k_0} jest indeksem nośnika, który jest wynikiem operacji.

Uwaga

Algebra wielorodzajowa będzie też oznaczana krócej

$$ALG =_{def} <A, F>$$

gdzie: A jest zbiorem nośników, a F jest zbiorem operacji, w tym operacji zeroargumentowych, czyli stałych.

Przykład 8.7

Rozpatruje się dwurodzajową algebrę określoną na liczbach całkowitych, która – oprócz operacji arytmetycznych: dodawania, odejmowania, mnożenia, dzielenia całkowitoliczbowego – obejmuje również operacje porównywania liczb: równy, nie mniejszy. Argumentami zarówno działań arytmetycznych, jak i operatorów porównania są liczby, wynikami działań są także liczby, wynikami zaś porównań są wartości logiczne. Tym samym wprowadzenie operacji porównań wprowadza niejawnie dodatkowy nośnik zawierający wartości logiczne. Może nim być na przykład zbiór

$$Boolean =_{def} \{false, true\}$$

Algebrę można przedstawić jako algebrę dwurodzajową

$$ALG_{Całkowite} =_{def}$$

$$<\{Całkowite, Boolean\}, \{0,1\} \cup \{-, +, -, *, /, =, \geq\}>$$

gdzie:

0, 1 są stałymi liczbowymi, *zerem* i *jedynką*,

$-$: $Całkowite \rightarrow Całkowite$ jest jednoargumentową operacją *zmiany znaku* liczby,

$+, -, *, /$: $Całkowite^2 \rightarrow Całkowite$ są dwuargumentowymi operacjami dodawania, odejmowania, mnożenia i dzielenia,

$=, \geq$: $Całkowite^2 \rightarrow Boolean$ są dwuargumentowymi operacjami porównań równy i nie mniejszy.

Algebry wielorodzajowe mogą być modelem złożonych typów danych.

Algebra $ALG_{Całkowite}$ może być potraktowana jako pewna charakterystyka całkowitoliczbowego typu danych spotykanego w językach programowania. Charakterystyka ta nie uwzględnia ograniczoności zbioru wartości typu. Pełną charakterystyką typu jest algebra przedstawiona w przykładzie 8.8.

Przykład 8.8

Typ całkowitoliczbowy na zbiorze

$$Integer =_{def} \{-N, ..., 0, ..., N\}$$

ma określone operacje arytmetyczne: zmiany znaku –, dodawania ⊕, odejmowania ⊖, mnożenia ⊗, dzielenia całkowitoliczbowego ⊘, oraz ma operacje porównywania liczb: równy =, nie mniejszy ≥, których wartościami są elementy zbioru

$$Boolean =_{def} \{false, true\}$$

Pełny model typu całkowitoliczbowego można przedstawić jako algebrę dwurodzajową

$$ALG_{Integer \cup \{nadmiar\}} =_{def} <\{Integer \cup \{nadmiar\}, Boolean\}, \{0,1\} \cup$$

$$\{-, \oplus, \ominus, \otimes, \oslash, =, \geq\}>$$

Stałe i operacje algebry mają sygnatury:

$0, 1 : \rightarrow Integer$

$- : Integer \rightarrow Integer$

$\oplus, \ominus, \otimes : (Integer \cup \{nadmiar\})^2 \rightarrow Integer \cup \{nadmiar\}$

$\oslash : (Integer \cup \{nadmiar\})^2 \rightarrow Integer \cup \{nadmiar\}$

$=, \geq : Integer^2 \rightarrow Boolean$

Oprócz operacji dzielenia, pozostałe operacje definiuje się podobnie, jak w przykładzie 8.6. Definicja operacji dzielenia przedstawia się następująco:

$$a \oslash b =_{def} \begin{cases} a/b & \text{gdy } a \in Integer,\ b \in Integer \setminus \{0\} \text{ oraz } |a/b| \leq N \\ nadmiar & \text{gdy } a \in Integer,\ b \in Integer \setminus \{0\} \text{ oraz } |a/b| > N \\ nadmiar & \text{gdy } b = 0 \text{ lub } a = nadmiar \text{ lub } b = nadmiar \end{cases}$$

Operacje porównań są obcięciem funkcji porównań = oraz ≥, określonych na zbiorze liczb całkowitych *Całkowite*, do zbioru *Integer*.

8.3. Termy

Z każdą algebrą jest związany pewien zbiór napisów, które powstają ze złożenia symboli stałych, zmiennych i działań algebry. Zbiór ten nazywa się zbiorem termów. Poniżej przedstawia się definicję termów, najpierw dla algebr jednorodzajowych, a następnie dla wielorodzajowych [Ehrig, Mahr 1985]. Niech

$$ALG =_{def} <A, \{c_1, ..., c_m\} \cup \{f_1, ..., f_n\}>$$

będzie pewną algebrą jednorodzajową oraz niech V będzie zbiorem zmiennych, to znaczy symboli, którym można przyporządkowywać pewne wartości z dziedziny A.

Symbolem $Term_{ALG}(V)$ oznacza się *zbiór termów* algebry ALG nad zbiorem zmiennych V. Zbiór ten jest zdefiniowany rekursywnie w sposób następujący:

- $V \subseteq Term_{ALG}(V)$ oraz $\{c_1, ..., c_m\} \subseteq Term_{ALG}(V)$, to znaczy, że zmienne i stałe są termami,
- jeżeli $t_1, ..., t_k \in Term_{ALG}(V)$, czyli napisy $t_1, ..., t_k$ są termami oraz f_j jest działaniem k-argumentowym, to $f_j(t_1, ..., t_n) \in Term_{ALG}(V)$, czyli napis postaci $f_j(t_1, ..., t_n)$ jest termem.

Uwaga

Jeżeli $t_1, t_2 \in Term_{ALG}(V)$ oraz f_j jest działaniem dwuargumentowym zapisywanym w konwencji wrostkowej, to za term będzie przyjmowany napis $(t_1\ f_j\ t_2) \in Term_{ALG}(V)$.

Termy są słowami nad pewnym alfabetem wyznaczonym przez daną algebrę. W skład takiego alfabetu wchodzą symbole stałych, zmiennych, działań oraz nawiasów i przecinka. Jak wynika z definicji, termy są napisami złożonymi w tym sensie, że term może się składać z części składowych, które również są termami. Termy, które są częściami składowymi innych termów, nazywa się podtermami. Dokładniej: term t_1 jest *podtermem* termu t_2, gdy t_1 jest podsłowem słowa t_2.

Zbiór termów nad pustym zbiorem zmiennych, czyli $Term_{ALG}(\varnothing)$, nazywa się zbiorem termów stałych.

Termy są napisami, które wyrażają pewne znaczenie – reprezentują one wartości ze zbioru A. Inaczej: są one tekstową reprezentacją pewnych wartości, należących do nośnika A. Termy stałe $Term_{ALG}(\varnothing)$ wyrażają bezpośrednio pewne wartości. Termy, w których występują zmienne $Term_{ALG}(V)$, również reprezentują pewne wartości, ale wartości te zależą od wartościowania zmiennych, czyli od wartości, jakie są przyporządkowane zmiennym V. *Wartościowanie* zmiennych jest wyrażane przez funkcję v o sygnaturze

$v : V \to A$

Wartość funkcji $v(v)$ dla zmiennej v wyznacza pewien element ze zbioru A, który zmienna ta reprezentuje.

W dalszym ciągu będzie się pisać $Term(\varnothing)$ i $Term(V)$, gdy z kontekstu wiadomo, o jaką algebrę chodzi.

Przykład 8.9

Do zbioru termów stałych $Term(\varnothing)$, generowanych przez algebrę

$ALG_{Nat} =_{def} <Nat, \{0, 1\} \cup \{+\}>$

gdzie + jest – jak poprzednio – dodawaniem w zbiorze liczb naturalnych, zapisywanym w notacji wrostkowej, należą na przykład napisy:

0, 1, (0 + 0), (0 + 1), (1 + 0), (1 + 1), (0 + (0 + 0)), (0 + (0 + 1))

Przy zapisie w notacji przedrostkowej te same napisy przyjmą postać:

0, 1, +(0, 0), +(0, 1), +(1, 0), +(1, 1), +(0, +(0, 0)), +(0, +(0, 1))

Niektóre termy, na przykład 1, (0 + 1), (1 + 0), ((0 + 0) + 1), reprezentują tę samą wartość – liczbę naturalną 1. Zbiór termów reprezentujących tę samą wartość jest oczywiście nieskończony.

Niech $V = \{a, b, c\}$. Do zbioru termów $Term(V)$ generowanych przez algebrę ALG_{Nat} będą na przykład należeć:

$a, b, c, (a + 0), (0 + b), (a + c), (c + b), (a + (b + 0)), (1 + (b + 1))$

Jeżeli założyć funkcję wartościowania

$v = \{<a, 1>, <b, 0>, <c, 1>\}$

to wyżej wymienione termy będą kolejno reprezentować wartości:

1, 0, 1, 1, 0, 2, 1, 0, 2

Zbiór termów dla algebry wielorodzajowej

$ALG =_{def} <\{A_1, ..., A_k\}, \{c_1, ..., c_m\} \cup \{f_1, ..., f_n\}>$ dla $m \in Nat$ oraz $k, n \in Nat \setminus \{0\}$

jest bardziej złożony. Wynika to z podziału termów na rodzaje. Rodzaj termu wskazuje na jedną z dziedzin $A_1, ..., A_k$ algebry ALG, której wartości term reprezentuje.

Niech V_i będzie zbiorem zmiennych rodzaju A_i. Zbiorem wszystkich zmiennych jest $V = V_1 \cup ... \cup V_k$. Zmienna $v \in V_i$ jest rodzaju A_i $(i = 1, ..., k)$, co będzie zapisywane $v : A_i$. Zmiennej $v \in V_i$ można przyporządkowywać wartości tylko ze zbioru A_i.

Stałe także mają swój rodzaj. Dla $c \in \{c_1, ..., c_m\}$ zapis $c : A_i$ oznacza, że stała c jest rodzaju A_i, czyli jest elementem zbioru A_i $(i = 1, ..., k)$.

Dalej, zamiast pisać rodzaj A_i, będzie się pisać krótko rodzaj i.

Zbiór termów rodzaju i $(i = 1, ..., k)$ dla algebry wielorodzajowej ALG nad zbiorem zmiennych V, oznaczany $Term_i(V)$, jest zdefiniowany rekursywnie w sposób następujący:

- jeżeli $c_j : A_i$, to $c_j \in Term_i(V)$
  $V_i \subseteq Term_i(V)$
- jeżeli napisy $t_1, ..., t_k$ są termami rodzajów $i_1, ..., i_k$ oraz $f_j : A_{j_1} \times ... \times A_{j_k} \to A_i$ jest działaniem k-argumentowym, to napis postaci $f_j(t_1, ..., t_n)$ jest termem rodzaju i, czyli $f_j(t_1, ..., t_n) \in Term_i(V)$.

Zbiór wszystkich termów dla algebry wielorodzajowej ALG nad zbiorem zmiennych V, oznaczany $Term(V)$, jest określony jako mnogościowa suma

$$Term(V) = \bigcup_{i=1}^{k} Term_i(V).$$

Uwaga

W programowaniu przyporządkowanie rodzajów stałym, zmiennym oraz wyrażeniom nazywa się *typowaniem* lub *typizacją*. Typizacja przejawia się w sposobie deklarowania stałych i zmiennych, w rozróżnianiu na przykład wyrażeń arytmetycznych i logicznych, w sprawdzaniu poprawnego użycia zmiennych w wyrażeniach itd.

Przykład 8.10

W zdefiniowanej wcześniej algebrze

$$ALG_{Integer \cup \{nadmiar\}} =_{def}$$

$$< \{Integer \cup \{nadmiar\}, Boolean\}, \{0, 1\} \cup \{-, \oplus, \ominus, \otimes, \oslash, =, \geq\}>$$

wyróżnia się dwa rodzaje termów: $Integer \cup \{nadmiar\}$ oraz $Boolean$. Zakłada się, że:

$$Integer = \{-10, ..., -1, 0, ..., 10\}.$$

Niech ponadto $V_{Integer \cup \{nadmiar\}} =_{def} \{a, b\}$ oraz $V_{Boolean} =_{def} \varnothing$.

Termami rodzaju $Integer \cup \{nadmiar\}$ są na przykład:

$$0, 1, a, b, (a \ominus b), (a \ominus (a \otimes b)).$$

Termy te reprezentują pewne wartości ze zbioru $Integer \cup \{nadmiar\}$, przy czym zależą one od wartościowania ν zbioru zmiennych. Niech $\nu = \{<a, 3>, <b, 4>\}$.

Term 0 reprezentuje wówczas wartość 0, term a – wartość 3, term b – wartość 4, a term $(a \ominus b)$ – wartość 7. Wartością termu $(a \ominus (a \otimes b))$ jest natomiast *nadmiar*, gdyż wartością jego podtermu $(a \otimes b)$ jest *nadmiar*, ponieważ $a*b > 10$.

Termami rodzaju *Boolean* są na przykład:

$$((a \ominus b) = (1 \otimes a)), \qquad ((-a \oplus b) \geq (b \oslash a))$$

Wartościami obu termów, przy wartościowaniu ν, jest *false*.

Wartości termów przy danym wartościowaniu ν w algebrze

$$ALG =_{def} <\{A_1, ..., A_k\}, \{c_1, ..., c_m\} \cup \{f_1, ..., f_n\}> \quad \text{dla } m \in Nat \text{ oraz } k, n \in Nat \setminus \{0\}$$

można zdefiniować ogólnie.

Przez $WAR_\nu(t)$ oznacza się wartość termu t przy wartościowaniu ν. WAR_ν jest funkcją o sygnaturze

$$WAR_\nu : Term(V) \rightarrow A$$

gdzie:

$$Term(V) = \bigcup_{i=1}^{k} Term_i(V),$$

$$A = \bigcup_{i=1}^{k} A_i.$$

Funkcję obliczania wartości termów WAR_ν można zdefiniować rekursywnie w sposób następujący:

- jeżeli term jest postaci v, gdzie v jest zmienną, czyli $v \in V$, to $WAR_\nu(v) = \nu(v)$,
- jeżeli term jest postaci c, gdzie c jest stałą, czyli $c \in \{c_1, ..., c_m\}$, to $WAR_\nu(c) = c$,
- jeżeli term jest postaci $f(t_1, ..., t_k)$, gdzie f jest k-argumentowym działaniem, czyli $f \in \{f_1, ..., f_n\}$, $t_1, ..., t_k$ są termami, czyli $t_1, ..., t_k \in Term(V)$, to

  $$WAR_\nu(f(t_1, ..., t_k)) = f(WAR_\nu(t_1), ..., WAR_\nu(t_k)).$$

Niech $Term_{ALG}(\varnothing)$ będzie zbiorem termów stałych pewnej algebry ALG. Wartość termu stałego t nie zależy od wartościowania ν. Dla dowolnych dwóch wartościowań ν oraz ν' zachodzi zatem

$$WAR_\nu(t) = WAR_{\nu'}(t)$$

Niech $WAR(t)$ oznacza wartość termu stałego t. Definiuje się relację binarną $\approx$ na zbiorze termów stałych:

jeżeli $t_1, t_2 \in Term_{ALG}(\varnothing)$, to $t_1 \approx t_2$ wtedy i tylko wtedy, gdy $WAR(t_1) = WAR(t_2)$.

Relacja $\approx$ jest oczywiście relacją równoważności i wyznacza podział zbioru termów stałych na klasy abstrakcji. Do jednej klasy abstrakcji należą wszystkie termy tego samego rodzaju, które reprezentują tę samą wartość. Oznacza to, że relację $\approx$ można byłoby określić jako rodzinę relacji binarnych zdefiniowanych na podzbiorze termów stałych ustalonego rodzaju, czyli $\approx =_{def} \{\approx_1, ..., \approx_k\}$, gdzie $\approx_i$ dla $i = 1, ..., k$ jest zdefiniowane:

jeżeli $t_1, t_2 \in Term_i(\varnothing)$, to $t_1 \approx t_2$ wtedy i tylko wtedy, gdy $WAR(t_1) = WAR(t_2)$.

Przykład 8.11

Dla algebry $ALG_{Integer \cup \{nadmiar\}}$, zdefiniowanej we wcześniejszym przykładzie, relacja R wyznacza podział termów stałych rodzaju $Integer \cup \{nadmiar\}$ na klasy abstrakcji, z których każda reprezentuje termy przyjmujące wartości $-10, ..., -1, 0, ..., 10$ oraz *nadmiar*. Każda z klas zawiera nieskończenie wiele termów. Na przykład do jednej klasy termów o wartości 0 należą między innymi:

$$0 \qquad ((0 \oplus 1) \otimes 0) \qquad (((1 \otimes 2) \ominus 3) \oplus 1)$$

Do klasy termów o wartości *nadmiar* należą między innymi:

$(1 \oplus nadmiar)$ $((0 \ominus nadmiar) \otimes 1)$ $(((1 \oslash 0) \ominus 0) \ominus 0)$

Niech $Term_{ALG}(V)$ będzie zbiorem termów algebry ALG na zbiorem zmiennych V. Na zbiorze tym można również zdefiniować relację równoważności, która jest uogólnieniem relacji równoważności $\approx$, zdefiniowanej na zbiorze termów stałych. Będzie ona oznaczona tym samym symbolem $\approx$ i będzie zdefiniowana jako rodzina relacji

$$\approx =_{def} \{\approx_1, ..., \approx_k\},$$

gdzie $\approx_i$ dla $i = 1, ..., k$ jest zdefiniowane:

jeżeli $t_1, t_2 \in Term_i(V)$, to $t_1 \approx_i t_2$ wtedy i tylko wtedy, gdy $WAR_v(t_1) = WAR_v(t_2)$ dla każdego wartościowania v.

Jeżeli dwa termy należą do jednej klasy abstrakcji wyznaczonej przez relację $\approx$, to dla dowolnie wybranego wartościowania reprezentują one tę samą wartość.

Przykład 8.12

Dla poprzednio rozważanej algebry

$$ALG_{Całkowite} =_{def} <\{Całkowite, Boolean\}, \{0, 1\} \cup \{-, +, -, *, /, =, \geq\}>$$

i zbioru termów nad zbiorem zmiennych $V_{Całkowite} =_{def} \{a, b\}$ oraz $V_{Boolean} =_{def} \emptyset$, relacja $\approx_V$ wyznacza podział termów rodzaju $Integer \cup \{nadmiar\}$ na klasy abstrakcji, z których każda reprezentuje termy przyjmujące te wartości dla dowolnie ustalonego wartościowania. Na przykład do tej samej klasy termów należą między innymi:

1 $((a - a) + 1)$ $(((1/1 - b) - 0) + b)$

Do innej klasy należą między innymi termy:

a $((a - b) + b)$ $(((a*1) - b) + b)$

8.4. Algebry Boole'a

Wśród algebr, które w informatyce mają szerokie zastosowania, szczególną rolę odgrywają *algebry Boole'a*. Stanowią one pewną klasę algebr zdefiniowaną przez określenie pewnych własności wyrażanych w postaci równości. Taki sposób definiowania algebr nazywa się *definiowaniem równościowym*.

Algebrą Boole'a określa się każdą algebrę o strukturze

$$BOOL =_{def} <A, \{0, 1\} \cup \{-, +, *\}>$$

gdzie:

- A jest dowolnym zbiorem, nośnikiem algebry,
- 0, 1 są stałymi, nazywanymi *zerem* i *jednością boolowską*,
- $-$ jest działaniem jednoargumentowym, nazywanym dopełnieniem boolowskim,
- $+, *$ są działaniami dwuargumentowymi, nazywanymi umownie *dodawaniem* i *mnożeniem boolowskim* (nie należy tych działań utożsamiać z działaniami arytmetycznymi ani mnogościowymi),

która ponadto, dla dowolnych $a, b, c \in A$, spełnia następujące własności:

1. $(a + b) = (b + a)$	$(a * b) = (b * a)$
2. $((a + b) + c) = (a + (b + c))$	$((a * b) * c) = (a * (b * c))$
3. $((a + b) * c) = ((a * c) + (b * c))$	$((a * b) + c) = ((a + c) * (b + c))$
4. $(a + 0) = a$	$(a * 1) = a$
5. $(a + (-a)) = 1$	$(a * (-a)) = 0$

Własności te wyrażają:

1. *przemienność* dodawania i mnożenia,
2. *łączność* dodawania i mnożenia,
3. *rozdzielność* mnożenia względem dodawania oraz dodawania względem mnożenia,
4. *neutralność* zera względem dodawania oraz jedynki względem mnożenia,
5. *neutralność* elementu odwrotnego względem dodawania oraz względem mnożenia.

Własności są wyrażane poprzez równości. Równość jest napisem postaci $t_1 = t_2$, gdzie składowe t_1 i t_2 są termami ze zbioru $Term_{BOOL}(V)$ nad dowolnym zbiorem zmiennych V. Równość $t_1 = t_2$ jest spełniona, gdy dla dowolnego wartościowania v oba termy reprezentują tę samą wartość, czyli $WAR_v(t_1) = WAR_v(t_2)$.

Od algebry Boole'a wymaga się, aby były spełnione wszystkie wyżej wymienione równości.

Przykład 8.13

Łatwo sprawdzić, że wcześniej zdefiniowana algebra

$$ALG_{Boolean} =_{def} <Boolean, \{not, and, or\}>$$

ma wszystkie wymagane własności, a zatem jest algebrą Boole'a.

Przykład 8.14

Niech U będzie dowolnym zbiorem, a 2^U rodziną wszystkich jego podzbiorów. Podobnie można sprawdzić, że algebrą Boole'a jest struktura

$$BOOL_U =_{def} <2^U, \{\emptyset, U\} \cup \{\setminus, \cup, \cap\}>$$

w której: 2^U jest nośnikiem algebry, $\emptyset$ oraz U są zerem i jednością, a działania mnogościowe $\setminus, \cup, \cap$ są odpowiednio dopełnieniem, sumą i iloczynem algebry.

Ogólniej, zamiast pełnej rodziny podzbiorów 2^U wystarczy przyjąć dowolną jej podrodzinę, zamkniętą ze względu na działania dopełnienia, sumy i iloczynu. Zamknięta jest na przykład rodzina złożona ze zbioru pustego i zbioru pełnego, czyli $\{\emptyset, U\}$, to znaczy wynikiem każdej z operacji wykonanej na elementach tej rodziny jest element należący do tej rodziny.

8.5. Homomorfizm algebr

Definiowanie algebry abstrakcyjnej wygodnie jest rozpoczynać od opisania jej struktury. Najprostszą jej charakterystyką jest sygnatura.

Sygnaturą algebry nazywa się parę

$$Sig =_{def} <S, OP>$$

gdzie: S jest niepustym zbiorem identyfikatorów (nazw) nośników (rodzajów), OP jest zbiorem deklaracji operacji. *Deklaracja operacji* będzie zapisywana w postaci

$$op : s_1\ s_2\ \ldots\ s_n \rightarrow s$$

gdzie: op jest identyfikatorem (nazwą) operacji, $s_1\ s_2\ \ldots\ s_n$ jest listą, której elementy s_1, s_2, $\ldots\ s_n \in S$ są identyfikatorami rodzajów argumentów, a $s \in S$ jest identyfikatorem (nazwą) rodzaju wartości operacji. Deklaracja operacji o nazwie op wskazuje na nazwy zbiorów jej argumentów i nazwę zbioru jej wartości. Jeżeli op jest operacją zeroargumentową, czyli stałą, to jej deklaracja ma postać

$$op : \rightarrow s$$

Zakłada się, że każda deklaracja operacji ma różną nazwę operacji, dlatego dalej, zamiast pisać $(op : s_1\ s_2\ \ldots\ s_n \rightarrow s) \in OP$, będzie się pisać krótko $op \in OP$.

Uwaga

Zapis postaci

$$op : s_1\ s_2\ \ldots\ s_n \rightarrow s$$

nie jest tym samym co zapis

$$op : s_1 \times s_2 \times \ldots \times s_n \rightarrow s$$

Przykład 8.15

Przykłady dwóch sygnatur $Sig_i =_{def} <S_i, OP_i>$ dla $i = 1, 2$, gdzie:

$S_1 =_{def} \{A\}$ $\quad OP_1 =_{def} \{e : \rightarrow A,\ d : A\ A \rightarrow A\}$

$S_2 =_{def} \{A, B\}$ $\quad OP_2 =_{def} \{e : \rightarrow A,\ d : A\ A \rightarrow A,\ r : A\ A \rightarrow B\}$

Sygnatura jest tylko pewną charakteryzacją algebry, a nie określeniem algebry. Może być wiele algebr mających tę samą sygnaturę.

Algebrą nad sygnaturą *Sig*, krótko *Sig-algebrą*, nazywa się parę

$$ALG =_{def} <A, F>$$

gdzie:

$A =_{def} \{A_s \mid s \in S\}$ jest rodziną zbiorów zwanych *nośnikami* lub *dziedzinami* algebry,

$F =_{def} \{f_{op} \mid op \in OP\}$ jest rodziną funkcji zwanych *operacjami* algebry, przy czym każdej deklaracji operacji $op \in OP$:

$$op : s_1\ s_2\ \ldots\ s_n \rightarrow s$$

odpowiada funkcja

$$f_{op} : A_{s_1} \times \ldots \times A_{s_n} \rightarrow A_s$$

Dwie algebry o tej samej sygnaturze nazywa się algebrami *podobnymi*.

Przykład 8.16

Przykładami dwóch algebr podobnych o sygnaturze Sig_1 z poprzedniego przykładu są:

$$ALG_1 =_{def} <Nat, \{1, +\}>$$

gdzie nośnikiem algebry ALG_1 jest zbiór liczb naturalnych *Nat*, stała 1 jest liczbą naturalną jeden, zaś operacja + jest dodawaniem w zbiorze liczb naturalnych.

$$ALG_2 =_{def} <Nat, \{1, *\}>$$

Nośnikiem algebry ALG_2 jest zbiór liczb naturalnych *Nat*, stała 1 jest liczbą naturalną jeden, zaś operacja * jest mnożeniem w zbiorze liczb naturalnych.

Przykładami dwóch podobnych algebr dwurodzajowych o sygnaturze Sig_2 są:

$$ALG_3 =_{def} <\{Nat, Logiczne\}, \{1, +, =\}>$$

Nośnikami algebry ALG_3 są zbiór liczb naturalnych *Nat* oraz zbiór wartości logicznych $Logiczne =_{def} \{prawda, fałsz\}$, 1 oraz + są – jak poprzednio – stałą jeden i dodawaniem w zbiorze liczb naturalnych, natomiast operacja = jest równością w zbiorze liczb naturalnych.

$$ALG_4(X) =_{def} <\{2^X, Logiczne\}, \{\emptyset, \cup, =\}>$$

Algebra $ALG_4(X)$ jest algebrą parametryzowaną zbiorem X. Jej nośnikami są: rodzina podzbiorów pewnego zbioru X oraz zbiór *Logiczne*, a operacjami są: stała $\emptyset$, która jest zbiorem pustym, operacja $\cup$, która jest sumą mnogościową, oraz =, która jest równością zbiorów.

Niech będą dane dwie algebry: $ALG_1 =_{def} <A, F>$ oraz $ALG_2 =_{def} <B, G>$ o sygnaturze $Sig = <S, OP>$. Oznacza to, że

$A =_{def} \{A_s \mid s \in S\}$ oraz $B =_{def} \{B_s \mid s \in S\}$ są *nośnikami* algebr, a
$F =_{def} \{f_{op} \mid op \in OP\}$ oraz $G =_{def} \{g_{op} \mid op \in OP\}$ są rodzinami *operacji* tych algebr.

Homomorfizmem z algebry ALG_1 w algebrę ALG_2 nazywa się taką rodzinę funkcji H

$$H =_{def} \{h_s : A_s \to B_s \mid s \in S\}$$

że dla każdej funkcji $f_{op} : A_{s_1} \times \ldots \times A_{s_n} \to A_s$, dla $op \in OP$, zachodzi warunek

$$h_s(f_{op}(a_1, \ldots, a_n)) = g_{op}(h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$$

dla dowolnych $a_j \in A_{s_j}$, $j = 1, \ldots, n$. Fakt, że H jest homorfizmem zapisuje się

$$H : ALG_1 \to ALG_2$$

Przykład 8.17

Dane są dwie algebry jednorodzajowe:

$ALG_1 = \langle Nat, \{1, +\}\rangle$

$ALG_2 = \langle Nat_n, \{1, /\}\rangle$

gdzie: $Nat_n = \{0, 1, \ldots, n-1\}$, a / oznacza dodawanie modulo n.

Homomorfizmem jest funkcja $h : Nat \to Nat_n$, zdefiniowana wzorem

$h(m)$ = reszta z dzielenia m przez n, dla $m \in Nat$.

Jeżeli każda z funkcji h_s homomorfizmu $H = \{h_s : A_s \to B_s \mid s \in S\}$ jest funkcją wzajemnie jednoznaczną, to H nazywa się *izomorfizmem*.

8.6. Algebra ilorazowa termów

Dana algebra wielorodzajowa $ALG = \langle A, F\rangle$ o sygnaturze $Sig = \langle S, OP\rangle$,

gdzie:

$A = \{A_s \mid s \in S\}$ jest rodziną nośników algebry ALG,

$F = \{f_{op} \mid op \in OP\}$ jest zbiorem operacji algebry ALG,

generuje zbiór termów nad ustalonym zbiorem zmiennych V

$$Term(V) = \bigcup_{s \in S} Term_s(V)$$

Zbiór ten może być podstawą do utworzenia nowej algebry wielorodzajowej, zwanej *algebrą termów* [Ehrig, Mahr 1985], [Rasiowa 1998], która jest podobna do algebry ALG.

Algebrę termów

$$ALG_{Term} =_{def} \langle \boldsymbol{A}, \boldsymbol{F}\rangle$$

dla algebry ALG definiuje się następująco:

$A = \{Term_s(V) \mid s \in S\}$ jest rodziną nośników algebry termów,
$F = \{f_{op} \mid op \in OP\}$ jest zbiorem operacji algebry termów, przy czym operacja f_{op} ma sygnaturę:

$$f_{op} : Term_{s_1}(V) \times \ldots \times Term_{s_n}(V) \to Term_s(V)$$

gdy deklaracja operacji ma postać: $op : s_1\, s_2, \ldots, s_n \to s$ i jest zdefiniowana następująco:
jeżeli $t_j \in Term_{s_j}$ dla $j = 1, \ldots, n$, to $f_{op}(t_1, \ldots, t_n) =_{def} f_{op}(t_1, \ldots, t_n)$.

Każdemu nośnikowi A_s i każdej operacji f_{op} w algebrze ALG odpowiadają nośnik $Term_s(V)$ i operacja f_{op} w algebrze termów ALG_{Term}.

Niech $\approx$ będzie wcześniej określoną rodziną relacji binarnych $\approx_s$ na zbiorze termów rodzaju $s \in S$.

Relacje te są relacjami równoważności i mają następującą własność:

Jeżeli t_{s_j}, t'_{s_j} są termami rodzaju s_j oraz $t_{s_j} \approx_{s_j} t'_{s_j}$, dla $j = 1, \ldots, n$, to

$$f_{op}(t_{s_1}, \ldots, t_{s_n}) \approx_s f_{op}(t'_{s_1}, \ldots, t'_{s_n})$$

dla $op \in OP$.

Rodzinę relacji równoważności o tej własności nazywa się *kongruencją*.

Kongruencja jest podstawą do zdefiniowania algebry, nazywanej *ilorazową algebrą termów* $ALG_{\approx}$ dla algebry ALG. Dodatkowo algebry te są homomorficzne, to znaczy istnieje homomorfizm $H : ALG \to ALG_{\approx}$.

Definicja algebry $ALG_{\approx}$ jest następująca:

$$ALG_{\approx} =_{def} \langle \overline{\boldsymbol{A}}, \overline{\boldsymbol{F}}\rangle$$

gdzie:

$\overline{\boldsymbol{A}} = \{Term_s(V)/\approx_s \mid s \in S\}$ jest rodziną zbiorów ilorazowych termów rodzajów $s \in S$,
$\overline{\boldsymbol{F}} = \{\overline{f}_{op} \mid op \in OP\}$ jest zbiorem operacji ilorazowej algebry termów, przy czym operacja $\overline{f}_{op}$ ma sygnaturę

$$\overline{f}_{op} : Term_{s_1}(V)/\approx_{s_1} \times \ldots \times Term_{s_n}(V)/\approx_{s_n} \to Term_s(V)/\approx_s$$

gdy deklaracja operacji ma postać: $op : s_1\, s_2 \ldots s_n \to s$ i jest zdefiniowana następująco:
jeżeli $[t_j] \in Term_{s_j}/\approx_{s_j}$, dla $j = 1, \ldots, n$, to $\overline{f}_{op}([t_1], \ldots, [t_n]) =_{def} [f_{op}(t_1, \ldots, t_n)]$.

Należy przypomnieć, że $[t_j]$ jest oznaczeniem klasy abstrakcji generowanej przez term t_j.

Ćwiczenia

1. Niech będzie dana algebra $ALG_{Nat} =_{def} <Nat, \{0, 1\}, \{+, -, *\}>$. Przedstawić gramatykę generującą poprawne wyrażenia (termy) zbudowane ze zmiennych reprezentujących liczby oraz z operacji podanej algebry.
2. Niech będzie dana algebra $ALG_{Int} =_{def} <Int, \{+, -, *\}>$, gdzie $Int =_{def} \{-100, ..., 0, 1, ..., 100\}$. Przedstawić definicję działań algebry pozwalającą na określenie ich wyniku dla dowolnych argumentów.
3. Zdefiniować algebrę definiującą typ znakowy (**string**) w języku Pascal lub C.
4. Przedstawić wielorodzajową algebrę, która będzie wyrażać znaczenie typu wyliczeniowego, zdefiniowanego w języku Pascal w sposób następujący:

   type *DniTygodnia* = (*pon*, *wt*, *śr*, *czw*, *piąt*, *sob*, *niedz*).
5. Zdefiniować gramatykę, która będzie generować zbiór termów rodzaju *DniTygodnia*, określonych przez algebrę zdefiniowaną w zadaniu 4.
6. Dla algebry z zadania 4. zdefiniować algebrę termów i ilorazową algebrę termów.
7. Niech $A =_{def} \{1, 2, 3, 5, 6, 10, 15, 30\}>$. Pokazać, że algebra *ALG* zdefiniowana następująco:

   $ALG = <A, \{1, 30\} \cup \{-, +, *\}>$

   gdzie dla $a, b \in A$

   $-a$ oznacza liczbę stanowiącą wynik dzielenia 30 przez a,

   $a + b$ oznacza najmniejszą wspólną wielokrotność liczb a oraz b,

   $a * b$ oznacza największy wspólny podzielnik liczb a oraz b,

   jest algebrą Boole'a.
8. Przedstawić wszystkie homomorfizmy algebry słów $ALG_1 = <\{a\}^*, \{\varepsilon, ^\wedge\}>$ nad alfabetem $\{a\}$ w algebrę słów $ALG_2 = <\{a, b\}^*, \{\varepsilon, ^\wedge\}>$ nad alfabetem $\{a, b\}$, gdzie ε jest słowem pustym, a $^\wedge$ jest konkatenacją słów.
9. Niech $ALG = <\{a, b\}^*, \{^\wedge\}>$, gdzie $^\wedge$ jest konkatenacją słów, będzie algebrą słów nad alfabetem $\{a, b\}$. Jaka jest moc zbioru wszystkich homomorfizmów $h : \{a, b\}^* \rightarrow \{a, b\}^*$ algebry słów *ALG* w samą siebie?
10. Pokazać, że istnieje homomorfizm między dowolną algebrą a generowaną przez nią ilorazową algebrą termów.

9. Rachunek zdań

9.1. Składnia

Rachunek zdań jest podstawową częścią logiki klasycznej. Elementy rachunku były nieformalnie wprowadzone i używane w poprzednich rozdziałach. W tym rozdziale przedstawia się składnię i semantykę rachunku zdań.

Język rachunku zdań, tak jak każdy język formalny, definiuje się przez podanie alfabetu – zbioru symboli podstawowych, oraz przez podanie zasad tworzenia z nich napisów – słów nad alfabetem. Symbole alfabetu nazywa się jednostkami *leksykalnymi*. Takie wyróżnienie wynika stąd, że jednostki leksykalne mogą być słowami nad innym alfabetem.

Alfabet języka rachunku zdań składa się z następujących czterech kategorii jednostek *leksykalnych*:

- symboli *stałych logicznych* reprezentowanych przez napisy **true** oraz **false**;
- przeliczalnej liczby symboli *zmiennych zdaniowych*, reprezentowanych przez dowolnie ustalone identyfikatory, dalej najczęściej będą używane pojedyncze małe litery $p, q, r, ...$;
- symboli *spójników logicznych*:

negacji	$\neg$
koniunkcji	$\wedge$
dysjunkcji (lub *alternatywy*)	$\vee$
implikacji	$\Rightarrow$
równoważności	$\Leftrightarrow$

- dwóch symboli *pomocniczych*:

lewy nawias	(
prawy nawias	)

Alfabet rachunku zdań jest zbiorem nieskończonym, ale co najwyżej przeliczalnym, gdyż dopuszcza się używanie dowolnej liczby identyfikatorów do reprezentacji zmiennych zdaniowych. W praktycznych zastosowaniach dysponuje się oczywiście zawsze skończoną liczbą zmiennych zdaniowych. Ich postać – ustalana dowolnie – nie ma wpływu na znaczenie języka.

Z alfabetu tworzy się pewne napisy – formuły, które – z definicji – są napisami poprawnie zbudowanymi. Dalej pojedyncze formuły będą oznaczane małymi literami greckimi.

Zbiór *formuł* rachunku zdań *FORM*, nad określonym wyżej alfabetem, jest definiowany rekursywnie w następujący sposób:

- symbole zmiennych zdaniowych oraz symbole stałych logicznych są formułami, nazywa się je formułami *elementarnymi* albo *atomowymi*;
- jeżeli α oraz β są formułami, to formułami nazywanymi formułami *złożonymi* są napisy:

$$\neg\alpha, (\alpha \Rightarrow \beta), (\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \Leftrightarrow \beta) .$$

Zbiór formuł *FORM* jest językiem formalnym rachunku zdań. Formuły rachunku zdań też nazywa się *zdaniami*.

Jeżeli α jest formułą, to każde podsłowo słowa α, które jest formułą, nazywa się *podformułą* α.

Przykład 9.1

Jeżeli dana jest formuła $(\alpha \wedge \beta)$, to jej podformułami są α oraz β, a także wszystkie podformuły α oraz β. Dla formuły

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

gdzie: p, q, r są zmiennymi zdaniowymi, jej podformułami są formuły:

$\neg(p \vee q)$, $(p \vee q)$, p, q, $\neg r$, r.

Uwaga

W celu zredukowania liczby nawiasów w formułach dalej się przyjmuje (jak w rozdziale 1.), że spójniki logiczne mają ustaloną kolejność stosowania (wiązania) spójników (od najsilniejszego do najsłabszego):

$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$.

Pozwala to pisać na przykład:

$\neg\alpha \wedge \beta$ zamiast $(\neg \alpha \wedge \beta)$,

$\neg\alpha \wedge \beta \vee \gamma$ zamiast $((\neg \alpha \wedge \beta) \vee \gamma)$,

gdzie: α oraz β są dowolnymi formułami.

Gdy takie same spójniki występują obok siebie, zakłada się dodatkowo, że występujące obok siebie spójniki $\wedge$, $\vee$ łączą w lewo, a występujące obok siebie spójniki $\Rightarrow$, $\Leftrightarrow$ łączą w prawo. Na przykład:

$p \wedge q \wedge r$ znaczy $(p \wedge q) \wedge r$,

$p \Rightarrow q \Rightarrow r$ znaczy $p \Rightarrow (q \Rightarrow r)$.

Przedstawiony język formalny rachunku zdań abstrahuje od postaci zmiennych zdaniowych. Język ten można ukonkretnić, definiując odpowiednią gramatykę bezkontekstową.

Przyjmuje się konwencję powszechnie stosowaną w językach programowania, że identyfikatorem jest niepusty skończony ciąg znaków, którego pierwszym elementem jest dowolna litera alfabetu łacińskiego, a elementami pozostałymi są litery lub cyfry arabskie.

Gramatykę generującą język formalny rachunku zdań (*RZ*) można zdefiniować następująco:

$G_{RZ} =_{def} <T_{RZ}, N_{RZ}, P_{RZ}, S_{RZ}>$,

gdzie:

$T_{RZ} =_{def}$ {**true**, **false**} $\cup$ {$a, b, ..., z$} $\cup$ {0, 1, ..., 9} $\cup$ {$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$} $\cup$ {(,)}

$N_{RZ} =_{def}$ {*formuła, formuła-elementarna, stała-logiczna, zmienna-zdaniowa, identyfikator, litera, cyfra spójnik-logiczny*}

$S_{RZ} =_{def}$ *formuła*

a zbiór produkcji P_{RZ}, zapisany w konwencji BNF, ma postać

$P_{RZ} =_{def}$ {*formuła* ::= *formuła-elementarna* |¬*formuła*|
(*formuła binarny-spójnik formuła*)
binarny-spójnik-logiczny ::= $\wedge$ | $\vee$ | $\Rightarrow$ | $\Leftrightarrow$
formuła-elementarna ::= *stała-logiczna* | *zmienna-zdaniowa*
stała-logiczna ::= **true** | **false**
zmienna-zdaniowa ::= *identyfikator*
identyfikator ::= *litera* | *identyfikator cyfra* | *identyfikator litera*
litera ::= a | b | ... | z
cyfra ::= 0 | 1 | ... | 9 }

Generowany przez podaną gramatykę G_{RZ} język formalny $L(G_{RZ})$ jest konkretyzacją zdefiniowanego rekursywnie zbioru formuł *FORM*.

W celu skrócenia zapisów całe formuły będą oznaczane pojedynczymi symbolami i dlatego będzie przydatne pojęcie równości tekstowej formuł. Fakt, że dwie formuły α, β są *identyczne tekstowo*, będzie zapisywany w postaci $\alpha \equiv \beta$.

9.2. Semantyka

Język formalny rachunku zdań w postaci przedstawionej wyżej jest językiem bez interpretacji. Interpretacja języka polega na ustaleniu znaczenia elementów języka – jego jednostek leksykalnych oraz formuł. W celu przedstawienia interpretacji jest konieczne posiadanie pewnego zestawu pojęć i sposobu ich reprezentacji w jakimś

zrozumiałym języku, to znaczy potrzebne jest posiadanie *metajęzyka* – języka służącego do opisu innego języka. Używanym tu metajęzykiem będzie język teorii mnogości, który był przedstawiany w poprzednich rozdziałach.

Określenie interpretacji języka polega na ustaleniu dziedzin interpretacji, to jest zbiorów obiektów, które będą wyrażać znaczenie elementów języka, oraz na ustaleniu sposobu przyporządkowania elementom języka obiektów z dziedziny interpretacji.

Dziedziną interpretacji rachunku zdań jest zbiór wartości logicznych:

$$Logiczne =_{def} \{prawda, fałsz\}$$

Dalej, zamiast pisać *prawda*, *fałsz*, będą używane skróty **P**, **F**.

Przyporządkowanie znaczenia elementom języka obiektów nad dziedziną interpretacji dokonuje się w dwóch etapach: najpierw określa się znaczenie symboli stałych i spójników logicznych, a następnie określa się znaczenie formuł.

W pierwszym etapie wprowadza się *funkcję interpretacji bazowej* I, krótko – *interpretację*, która określa znaczenie symboli stałych i spójników logicznych.

Interpretacją (znaczeniem) symboli **true** oraz **false** są wartości logiczne, odpowiednio *prawda* oraz *fałsz*. Formalnie wyraża to funkcja interpretacji I w sposób następujący:

$$I(\mathbf{true}) =_{def} \mathbf{P}$$
$$I(\mathbf{false}) =_{def} \mathbf{F}$$

Symbolom spójników logicznym: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, interpretacja I przyporządkowuje funkcje o następujących sygnaturach:

$$I(\neg) : Logiczne \to Logiczne$$
$$\_I(\wedge)\_ , \_I(\vee)\_ , \_I(\Rightarrow)\_ , \_I(\Leftrightarrow)\_ : Logiczne^2 \to Logiczne$$

Funkcje te są określone szczegółowo przez tablicę 9.1.

Tablica 9.1

a	b	$I(\neg)(a)$	$a\ I(\wedge)\ b$	$a\ I(\vee)\ b$	$a\ I(\Rightarrow)\ b$	$a\ I(\Leftrightarrow)\ b$
P	P	F	P	P	P	P
F	P	P	F	P	P	F
F	F	P	F	F	P	P
P	F	F	F	P	F	F

Tablica określa tak zwaną *standardową* albo *główną* interpretację spójników logicznych. W dalszym ciągu symbolom spójników logicznych będzie przyporządkowywana wyłącznie standardowa interpretacja, dlatego w zapisie symboli stałych i spójników logicznych symbol interpretacji I będzie pomijany. W zależności od kontekstu symbole spójników będą traktowane wyłącznie jako symbole bądź jako wyżej zdefiniowane funkcje. Tak właśnie było w podrozdziale 1.2, w którym po raz pierwszy zdefiniowano znaczenie spójników logicznych; symbole spójników logicznych były tam traktowane jako funkcje.

Dziedzina interpretacji wraz z funkcją interpretacji stałych i spójników logicznych wyznaczają algebrę jednorodzajową postaci

$$<Logiczne, \{I(\mathbf{true}), I(\mathbf{false})\}, \{I(\neg), I(\wedge), I(\vee), I(\Rightarrow), I(\Leftrightarrow)\}>.$$

Drugim etapem definiowania interpretacji języka rachunku zdań jest określenie znaczenia (semantyki) dowolnych formuł.

Dokonuje się tego rekursywnie – podobnie jak dla termów (podrozdział 8.3) – rozpoczynając od formuł elementarnych. Stałe, które są formułami elementarnymi, mają już ustaloną interpretację. Formułami elementarnymi są też zmienne zdaniowe. Pojedyncza zmienna reprezentuje prostą, niepodzielną wypowiedź, której można dowolnie przypisać wartość logiczną *prawda* albo *fałsz*. Interpretacja (znaczenie) symbolu zmiennej zdaniowej polega więc na przypisaniu temu symbolowi wartości **P** (*prawda*) albo **F** (*fałsz*). Niech *ZmienneZdaniowe* oznacza zbiór zmiennych zdaniowych. Przypisanie wartości zmiennej będzie wyrażać funkcja *wartościowania* zmiennych

$$\nu : ZmienneZdaniowe \to Logiczne,$$

która zmiennej zdaniowej $p \in ZmienneZdaniowe$ przyporządkowuje pewną wartość logiczną $\nu(p) \in Logiczne$.

Mając ustaloną funkcję interpretacji bazowej I oraz funkcję wartościowania ν można jednoznacznie zdefiniować nową funkcję, która każdej formule $\alpha \in FORM$ przyporządkowuje wartość logiczną *prawda* albo *fałsz*. Ta nowa funkcja

$$INT_\nu : FORM \to Logiczne$$

jest zdefiniowana rekursywnie w sposób następujący:

- Jeżeli formuła α jest formułą w postaci stałej logicznej, to:
  $$INT_\nu(\mathbf{true}) =_{def} I(\mathbf{true}) = \mathbf{P}$$
  $$INT_\nu(\mathbf{false}) =_{def} I(\mathbf{false}) = \mathbf{F}$$
- Jeżeli formuła α ma postać zmiennej zdaniowej p, to
  $$INT_\nu(\alpha) =_{def} \nu(p).$$
- Jeżeli formuła jest formułą złożoną, to:
  $$INT_\nu(\neg\alpha) =_{def} I(\neg)(INT_\nu(\alpha)),$$
  $$INT_\nu(\alpha \circ \beta) =_{def} INT_\nu(\alpha)\ I(\circ)\ INT_\nu(\beta),$$

  gdzie: $\circ$ jest dowolnym binarnym spójnikiem logicznym, czyli $\circ \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$, a $I(\circ)$ jest jego interpretacją.

Interpretacja stałych logicznych nie zależy od wartościowania ν, a interpretacja zmiennych zdaniowych nie zależy od interpretacji bazowej I.

Uwaga

Do opisu semantyki formalnego języka rachunku zdań został użyty pewien metajęzyk. Warunkiem decydującym o wyborze danego metajęzyka jest jego zrozumiałość i dostateczna siła ekspresji, pozwalająca na wyrażenie odpowiednich faktów. W naszym przypadku metajęzykiem jest język elementarnej teorii zbiorów, który został wprowadzony w poprzednich rozdziałach. Do opisu języka elementarnej teorii mnogości był natomiast użyty język naturalny, który pełnił rolę metajęzyka względem języka teorii mnogości. Idea definiowania znaczenia języka za pomocą innego języka – metajęzyka pochodzi od polskiego logika Alfreda Tarskiego[18].

Rozróżnienie pomiędzy językiem a metajęzykiem wyeliminowało wiele, znanych jeszcze w starożytności, paradoksów lingwistycznych w rodzaju: *to zdanie jest prawdziwe* albo *to zdanie jest fałszywe*. Przypuścimy, że chcemy wykazać, że *Ziemia jest płaska*. W tym celu wystarczy rozpatrzyć zdanie:

Albo *całe to zdanie jest fałszywe*, albo *Ziemia jest płaska*.

Zdanie to jest albo prawdziwe, albo fałszywe. Jeśli jest fałszywe, to – zgodnie z jego treścią – Ziemia musi być płaska. Jeśli zaś jest prawdziwe, to prawdziwa musi być albo jego pierwsza część: *całe to zdanie jest fałszywe*, albo druga: *Ziemia jest płaska*. Skoro przyjęliśmy, że całe zdanie jest prawdziwe, prawdziwa więc musi być jego druga część, czyli że Ziemia jest płaska. Zastępując zdanie *Ziemia jest płaska* dowolnym innym zdaniem, mocą takiego samego rozumowania możemy wykazać jego prawdziwość.

Rozróżnienie pomiędzy językiem a metajęzykiem ma jeszcze jedną zdumiewającą konsekwencję – nie istnieje nic takiego jak prawda absolutna. Można prowadzić dowody w obrębie jednego języka, określające, które wyrażenia języka uznajemy za prawdziwe. Pojęcie prawdy nie jest sformułowane w tym języku, lecz w jego metajęzyku. Z kolei, w obrębie metajęzyka mamy do czynienia z innymi wyrażeniami i dowodami, które określają ich prawdziwość, ale pojęcie prawdziwości zdań metajęzyka jest określone w jego metajęzyku (czyli metametajęzyku).

Formuła α *spełnia interpretację* INT *przy wartościowaniu* v, co będzie zapisywane w postaci

$$INT_v \models \alpha$$

wtedy i tylko wtedy, gdy $INT_v(\alpha) = \mathbf{P}$. Symbol $\models$ jest nazywany symbolem spełniania.

Formuła α *spełnia interpretację* INT, co będzie zapisywane w postaci

$$INT \models \alpha$$

wtedy i tylko wtedy, gdy α spełnia interpretację INT przy dowolnym wartościowaniu v.

[18] Alfred Tarski (1901–1983).

Ponieważ jest rozważana tylko interpretacja standardowa, symbol INT będzie pomijany

$$\models \alpha$$

Formułę taką nazywa się *tautologią* rachunku zdań.

Dwie formuły α oraz β są *równoważne semantycznie*, jeżeli przy tej samej interpretacji i przy tym samym wartościowaniu są jednocześnie spełnione albo niespełnione. Fakt równoważności semantycznej formuł zapisuje się w postaci

$$\alpha = \beta.$$

Uwaga

Symbol równoważności semantycznej = należy odróżnić od symbolu równoważności tekstowej ≡. Dla dwóch dowolnych formuł α, β, jeżeli $\alpha \equiv \beta$, to oczywiście również $\alpha = \beta$, natomiast wynikanie odwrotne nie zachodzi.

Pomiędzy spójnikiem równoważności $\Leftrightarrow$ a równoważnością semantyczną = zachodzi związek wyrażający się przez własność

Formuła postaci $\alpha \Leftrightarrow \beta$ jest tautologią, wtedy i tylko wtedy, gdy $\alpha = \beta$.

9.3. Dowodzenie metodą zero-jedynkową

Bezpośrednio z definicji interpretacji wynika, że sprawdzenie, czy dana formuła jest tautologią, może polegać na wyliczeniu prawdziwości formuły dla wszystkich możliwych wartościowań zmiennych zdaniowych występujących w tej formule. Liczba takich wartościowań wynosi 2^n, gdzie n jest liczbą zmiennych zdaniowych. Postępowanie takie określa się mianem *metody zero-jedynkowej*. Jej istotę wyjaśnia przykład.

Przykład 9.2

W celu pokazania, że formuła

$$p \Rightarrow (q \Rightarrow p)$$

jest tautologią rachunku zdań, wystarczy zbudować tablicę prawdziwościową, w której są zestawione wszystkie możliwe wartościowania zmiennych i obliczone dla nich wartości formuły i ewentualnie jej podformuł.

p	q	$q \Rightarrow p$	$p \Rightarrow (q \Rightarrow p)$
F	**F**	**P**	**P**
F	**P**	**F**	**P**
P	**F**	**P**	**P**
P	**P**	**P**	**P**

Ponieważ formuła jest spełniona przy dowolnym wartościowaniu występujących w niej zmiennych, jest więc tautologią.