Computer Science • Vol. 6 • 2004

Leszek Dubiel*

FUNKCJE NORMALNE JAKO NOWY MODEL DEFINIOWANIA FUNKCJI OBLICZALNYCH

Artykuł prezentuje nową metodę definiowania funkcji obliczalnych. Metoda jest formalizacją tradycyjnego zapisu funkcji i pozwala na określanie funkcji w sposób intuicyjny. W systemie funkcji rekursywnych Hilberta nie wszystkie odwzorowania, które mają łatwe algorytmy obliczania, mogą być równie łatwo zdefiniowane formalnie, czego przykładem jest funkcja Ackermanna. Funkcje normalne są pozbawione tej wady.

Słowa kluczowe: *maszyna Turinga, funkcje rekursywne, funkcje normalne*

NORMAL FUNCTIONS AS A NEW WAY OF DEFINING COMPUTABLE FUNCTIONS

Report sets new method of defining computable functions. This is formalization of traditional function descriptions, so it allows to define functions in very intuitive way. Discovery of Ackermann function proved that not all functions that can be easily computed can be so easily described with Hilbert's system of recursive functions. Normal functions lack this disadvantage.

Keywords: *The Turing Machine, recursive functions, normal functions*

1. Geneza problemu

1.1. Historia funkcji rekursywnych

W 1900 roku w Paryżu na Drugim Międzynarodowym Kongresie Matematyków Dawid Hilbert przedstawił listę dwudziestu trzech problemów, z którymi nie mogła się uporać ówczesna matematyka [1, 16, 21]. Jedno z naważniejszych pytań, na które nie znano wówczas odpowiedzi, dotyczyło istnienia algorytmicznej metody pozwalającej efektywnie rozwiązać każde zadanie matematyczne. Hilbert uważał, że znalezienie algorytmu rozwiązującego dowolny problem wiąże się ze zbudowaniem formalnego systemu matematycznego opartego na aksjomatach, w którym dowodzenie twierdzeń przebiegałoby mechanicznie według pewnych reguł wnioskowania [16].

W swoich badaniach Hilbert analizował funkcje liczb naturalnych i potraktował je jak obiekty konstruktywne, gdzie funkcje bardziej skomplikowane mogły być obliczane poprzez odpowiednie zastosowanie algorytmów określonych dla prostszych funkcji.

*Katedra Informatyki, Akademia Górniczo-Hutnicza w Krakowie, leszek@dubiel.pl

Definicje najprostszych funkcji, czyli tak zwanych **funkcji bazowych**, podsunęła podana w 1889 roku przez włoskiego matematyka Giuseppe Peano aksjomatyzacja liczb naturalnych [1]. Były to funkcje następnika i poprzednika oraz funkcja zerowa. Aby umożliwić konstruowanie odwzorowań wieloargumentowych, do bazy wprowadzono dodatkowo funkcję projekcji. Do budowania nowych funkcji Hilbert zasugerował użycie dwóch strategii – uogólnionego składania i rekursji pierwotnej [1, 7].

Wszystkie funkcje, które można skonstruować na podstawie funkcji bazowych, opierając się na strategiach podanych przez Hilberta, nazwano funkcjami pierwotnie rekursywnymi. Dodawanie, mnożenie, potęgowanie, silnia i wiele innych są przykładami takich odwzorowań. W 1925 roku Hilbert zadał pytanie, czy istnieją funkcje w oczywisty sposób obliczalne, które nie są pierwotnie rekursywne. W roku 1928 Wilhelm Ackermann podał kontrprzykład [1, 23], czyli funkcję niemożliwą do skonstruowania przy użyciu podanych metod, dla której istniał algorytm obliczania. System funkcji pierwotnie rekursywnych okazał się niekompletny, a zatem należało go uzupełnić. Uczynił to dopiero w 1936 roku amerykański matematyk Stephen Kleen, który wprowadzając do zbioru strategii operator minimum, zdefiniował funkcje **ogólnie rekursywne**.

1.2. Definicja funkcji rekursywnych

System funkcji pierwotnie rekursywnych składa się z trzech funkcji bazowych oraz dwóch operatorów pozwalających tworzyć nowe funkcje z wcześniej zdefiniowanych. Funkcjami bazowymi są:

funkcja zerowa:

$$\mathcal{Z}(x) = 0,$$

funkcja następnika:

$$\mathcal{S}(x) = x + 1,$$

funkcja projekcji:

$$\mathcal{I}_n^i(x_1, x_2, \ldots, x_n) = x_i.$$

Pierwszą strategią tworzenia nowych funkcji jest **operator podstawienia**. Określa on nową n-argumentową funkcję f pierwotnie rekursywną w ten sposób, że jej wartość w punkcie $(x_1, x_2, \ldots, x_n)$ wynosi:

$$f(x_1, x_2, \ldots, x_n) = \\ = g\left(h_1(x_1, x_2, \ldots, x_n), h_2(x_1, x_2, \ldots, x_n), \ldots, h_m(x_1, x_2, \ldots, x_n)\right),$$

gdzie:

g – m-argumentowa funkcja pierwotnie rekursywna,

$h_1, h_2, \ldots, h_m$ – n-argumentowe funkcje pierwotnie rekursywne.

Druga strategia opracowana przez Hilberta nazywana jest **operatorem rekursji**. Nowa $(n+1)$-argumentowa funkcja pierwotnie rekursywna f jest definiowana na

podstawie n-argumentowej funkcji pierwotnie rekursywnej g oraz $(n+2)$-argumentowej funkcji pierwotnie rekursywnej h w ten sposób, że jeśli ostatni argument funkcji f jest zerem, to wartość funkcji wynosi:

$$f(x_1, x_2, \ldots, x_n, 0) = g(x_1, x_2, \ldots, x_n),$$

a w przeciwnym wypadku:

$$f(x_1, x_2, \ldots, x_n, y+1) = h\big(x_1, x_2, \ldots, x_n, y, f(x_1, x_2, \ldots, x_n, y)\big).$$

1.3. Funkcja Ackermanna

Funkcja następnika jest funkcją bazową. Korzystając z operatora podstawienia można zdefiniować funkcję s_2, która zwraca następnik następnika podanego argumentu, czyli dla dowolnej liczby naturalnej x prawdą jest:

$$s_2(x) = \mathcal{S}(\mathcal{S}(x)).$$

Stosując tę samą regułę dla funkcji następnika oraz s_2, powstaje kolejne odwzorowanie s_3 określone poprzez operator podstawienia, czyli:

$$s_3(x) = \mathcal{S}(s_2(x)),$$

co można również zapisać w postaci:

$$s_3(x) = \mathcal{S}(\mathcal{S}(\mathcal{S}(x))).$$

Uogólniając, funkcja pierwotnie rekursywna s_n powstaje w wyniku działania operatorem podstawienia na funkcje $\mathcal{S}$ oraz s_{n-1}, czyli:

$$s_n(x) = \mathcal{S}(s_{n-1}(x)),$$

co zapisuje się jako:

$$s_n = \underbrace{\mathcal{S}\big(\ldots\mathcal{S}(\mathcal{S}(}_{\text{n razy}} x\,)\,)\big).$$

Funkcja s_n danej liczbie naturalnej przyporządkowuje jej n-ty następnik, czyli można powiedzieć, że liczba n jest parametrem obliczenia, czyli powinna być argumentem funkcji. Niech f_1 będzie taką funkcją dwóch argumentów, że dla dwolnej liczby naturalnej x zachodzi:

$$f_1(x, n) = s_n(x).$$

Algorytmy obliczania wartości poszczególnych funkcji $s_1, s_2, \ldots, s_n$ istnieją, gdyż są to funkcje pierwotnie rekursywne. Operacja przeniesienia parametru n z nazwy funkcji na pozycję argumentu nie jest ujęta w systemie Hilberta. Zatem czy funkcja f_1 jest rekursywna?

Korzystając z definicji funkcji s_n, można powyższe równanie zapisać w postaci:

$$f_1(x,n) = \mathcal{S}\big(s_{n-1}(x)\big),$$

a po kolejnym podstawieniu, opierając się na definicji funkcji f_1, postać:

$$f_1(x,n) = \mathcal{S}\big(f_1(x,n-1)\big),$$

która przypomina warunek z operatora rekursji pierwotnej. Aby operator był w pełni zdefiniowany, należy określić wartość funkcji f_1 dla n równego zero. Przy założeniu, że:

$$f_1(x,0) = x,$$

czyli bardziej precyzyjnie:

$$f_1(x,0) = \mathcal{I}_1^1(x),$$

funkcja f_1 określa dodawanie liczb naturalnych. Jak widać, funkcja ta jest funkcją rekursywną, gdyż została określona poprzez operator rekursji i funkcję następnika.

Postępując w stosunku do funkcji f_1 tak, jak powyżej dla funkcji następnika, można zdefiniować funkcję f_2, która dla każdej liczby naturalnej x przyjmuje wartość:

$$f_2(x,n) = \underbrace{f_1\big(x,\dots(f_1(x,(f_1(x,}_{n \text{ razy}}0))))\dots\big),$$

czyli:

$$f_2(x,n) = f_1\big(x,(f_2(x,n-1))\big).$$

Przy dodatkowym założeniu:

$$f_2(x,0) = 0$$

funkcja f_2 staje się operacją mnożenia dwóch liczb naturalnych. Ponieważ została ona utworzona w wyniku działania operatora rekursji pierwotnej na funkcję pierwotnie rekursywną f_1, to funkcja f_2 również jest pierwotnie rekursywna.

Potęgowanie liczb naturalnych definiuje się poprzez funkcję:

$$f_3(x,n) = f_2\big(x, f_3(x,n-1)\big),$$

przy założeniu początkowym:

$$f_3(x,0) = 1.$$

Kontynuując powyższe rozumowanie, można zdefiniować cały ciąg funkcji $f_1, f_2, f_3, \dots$, w którym zależności pomiędzy kolejnymi elementami wyraża równanie:

$$f_k(x,n) = f_{k-1}\big(x, f_k(x,n-1)\big).$$

Do obliczenia konkretnej wartości danej funkcji z ciągu potrzebne są trzy parametry – liczby k, x oraz n, czyli można zdefiniować nową funkcję g, której wartość w dowolnym punkcie jest wyrażona równaniem:

$$g(k, x, n) = f_k(x, n),$$

czyli po dokonaniu odpowiednich podstawień:

$$g(k, x, n) = g\big(k - 1,\ x,\ g(k, x, n - 1)\big).$$

Aby można było obliczać wartości funkcji g, należy ustalić wartości, jakie ona przyjmuje dla argumentów k lub n równych zero. Równanie określające zachowanie funkcji g dla parametru n wynika z warunku początkowego dla funkcji f_k, który ma ogólną postać:

$$f_k(x, 0) = 1,$$

czyli w konsekwencji:

$$g(k, x, 0) = 1.$$

Parametr k oznacza numer funkcji w wyżej wymienionym ciągu. Założenie, iż funkcja f_0 jest takim odwzorowaniem, że:

$$f_0(x, n) = \mathcal{S}(n),$$

implikuje drugi z warunków:

$$g(0, x, n) = \mathcal{S}(n).$$

Powyższa funkcja została opisana w pracy doktorskiej niemieckiego matetmatyka Wilhelma Ackermanna w 1928 roku, której promotorem był sam David Hilbert [23]. W kolejnych latach definicja funkcji została uproszczona przez Rosza Petera i Raphaela Robinson, którzy liczbę argumentów zredukowali do dwóch, a warunki początkowe ustalili tak, aby funkcja rosła szybciej od wszystkich funkcji pierwotnie rekursywnych [23, 1]. Obecna definicja tej funkcji jest opisana równaniami:

$$\begin{aligned}
\mathcal{A}(0, y) &= \mathcal{S}(y) \\
\mathcal{A}(x, 0) &= \mathcal{A}(x - 1, 1) \\
\mathcal{A}(x, y) &= \mathcal{A}\big(x - 1,\ \mathcal{A}(x, y - 1)\big).
\end{aligned}$$

Funkcja Ackermanna jest odwzorowaniem, dla którego każdy z łatwością może podać algorytm obliczania, ale nie jest to funkcja pierwotnie rekursywna. Ackermann udowodnił, że model obliczenia zaprezentowany przez Hilberta jest niepełny.

1.4. Krytyka modelu funkcji rekursywnych

Każda teoria naukowa powinna być spójna i prosta. Hilbert wprowadzając system opisywania funkcji, wprowadził trzy funkcje pierwotne oraz dwa operatory. Przy użyciu tych konstrukcji można zdefiniować podstawowe operacje arytmetyczne. Co więcej definicje są proste, gdyż odwzorowują wyobrażenie człowieka o tych funkcjach – na

przykład mnożenie jest wielokrotnym dodawaniem i w systemie funkcji pierwotnie rekursywnych właśnie tak jest definiowane. Po wielu latach od publikacji Hilberta zdefiniowano funkcję Ackermanna, która nie jest funkcją pierwotnie rekursywną. Aby rozwiązać ten problem, system Hilberta uzupełniono o dodatkowy operator, określając tym samym zbiór ogólnie funkcji rekursywnych [1, 4, 7, 12]. Niestety takie poprawione rozwiązanie przestało być spójne i proste. Funkcja Akcermanna jest bardzo łatwa do zapamiętania, można dla niej bardzo szybko napisać algorytm obliczania, ale udowodnienie, że jest to funkcja rekursywna, jest skomplikowane. Być może, po odkryciu funkcji Ackermanna należało stworzyć zupełnie nowy model obliczenia, w którym zapis formalny byłby zgodny z intuicyjnym pojmowaniem funkcji.

2. Funkcje normalne

2.1. Cel rozważań

Niniejsze opracowanie przedstawia sposób określania funkcji obliczalnych, który jest bardziej intuicyjny niż system funkcji rekursywnych. Funkcje rekursywne mają bardzo ścisłe zasady konstruowania odwzorowań, w których określa się wartość dla argumentu y na podstawie wartości funkcji dla argumentu $y - 1$. Żadne inne zależności nie mogą być zapisane w systemie Hilberta. Sposób definiowania funkcji opisany poniżej eliminuje tę wadę, gdyż pozwala na opisywanie nowego odwzorowania poprzez porównanie go do innych funkcji. Zatem w systemie Hilberta definicja określa operacje na kolejnych wartościach funkcji, a w poniższym rozwiązaniu operuje się na całych zbiorach stanowiących funkcje.

Wywód został podzielony na kilka części. W pierwszej przeanalizowano tradycyjny sposób zapisywania funkcji z użyciem klamerki. W drugiej sformalizowano metody opisu jako operatory funkcyjne oraz ustalono zbiór funkcji pierwotnych. Funkcje zdefinowane przy zastosowaniu nowej metody będą nazywane **funkcjami normalnymi**, aby uwypuklić intuicyjny charakter rozwiązania – definicja tych odwzorowań została umieszczona w osobnym punkcie wywodu.

2.2. Analiza intuicyjnego zapisu

Takie funkcje jak dodawanie lub mnożenie są wszystkim znane i nie ma potrzeby ich definiowania – są to funkcje, których należy nauczyć się na pamięć. Jednak inne popularne odwzorowania, do których zaliczamy silnię, ciąg Fibonacciego czy funkcję Ackermanna muszą zostać opisane w taki sposób, aby inny człowiek potrafił stwierdzić, czy dana para należy do przedstawionej funkcji czy nie. Na przykład mówi się, że silnia liczby n jest iloczynem wszystkich liczb od 1 do n, a silnia liczby 0 wynosi 1. Taką zależność zapisuje się przy użyciu klamerki w postaci:

$$\mathcal{N}(n) = \begin{cases} 1, & \text{gdy } n = 0, \\ 1 * 2 * \ldots * n & \text{w przeciwnym wypadku,} \end{cases}$$

lub bardziej formalnie:

$$\mathcal{N}(n) = \begin{cases} 1, & \text{gdy } n = 0, \\ \mathcal{N}(n-1) * n & \text{w przeciwnym wypadku.} \end{cases}$$

Ciąg Fibonacciego opisuje się jako funkcję, której kolejna wartość jest sumą dwóch poprzednich, a wartości dla 0 oraz 1 wynoszą 1. Zatem opis przy użyciu klamerki przybiera postać:

$$\mathcal{F}(n) \begin{cases} 1, & \text{gdy } n \in \{0, 1\}, \\ \mathcal{F}(n-1) + \mathcal{F}(n-2) & \text{w przeciwnym wypadku.} \end{cases}$$

Funkcja Ackermanna, która wcześniej była definiowana przy użyciu trzech równań jest potocznie zapisywana jako:

$$\mathcal{A}(x, y) = \begin{cases} y + 1, & \text{gdy } x = 0, \\ \mathcal{A}(x-1, 1), & \text{gdy } y = 0, \\ \mathcal{A}\big(x-1,\ \mathcal{A}(x,\, y-1)\big) & \text{w przeciwnym wypadku.} \end{cases}$$

Zarówno ciąg Fibonacciego, jak i funkcja Ackermanna są przykładami funkcji, które łatwo zapisuje się z użyciem klamerki, ale aby udowodnić, że są one funkcjami rekursywnymi, trzeba zapisać je w nienaturalnej i skomplikowanej postaci.

2.3. Formalizacja opisu

Przedstawione przykłady pokazują, że tradycyjny opis funkcji jest wygodny i intuicyjny oraz wystarcza do określenia nawet takich odwzorowań, które nie mogą być zdefiniowane przy użyciu systemu Hilberta. Jednak taki sposób opisu musi zostać sformalizowany. Wymienione wcześniej definicje korzystają z operacji dodawania, mnożenia, odejmowania i testowania liczb. Zatem, aby dokonać formalizacji, należy ujednolicić zapis, i najlepiej, aby przyjął on formę funkcji, które są budowane z prostszych odwzorowań. Najprostsze z odwzorowań muszą zostać zdefiniowane wprost jako pewne zbiory par.

Ponieważ rozważane funkcje operują na liczbach naturalnych, więc definicje najprostszych funkcji powinny opisywać podstawowe własności zbioru liczb naturalnych. Zero jest szczególnym elementem tego zbioru, a w definicjach innych funkcji często występuje zero, więc jednym z podstawowych odwzorowań, jakie można zdefiniować jest **funkcja zerowa** $\mathcal{Z}$, która każdej liczbie przyporządkowuje zero. Funkcja zerowa jest zatem zbiorem par:

$$\mathcal{Z} = \big\{(0,0),\, (1,0),\, (2,0),\, (3,0),\, (4,0),\, \ldots\big\},$$

czyli:

$$\mathcal{Z} = \big\{(x, y) \colon\ x \in \mathbb{N} \ \wedge\ y = 0\big\}.$$

Kolejne najprostsze odwzorowanie odzwierciedla właściwość zbioru liczb naturalnych polegającą na tym, iż każdej liczbie naturalnej jest przyporządkowana dokładnie jedna liczba naturalna nazywana jej następnikiem. Odwzorowanie to będzie nazywane **funkcją następnika** $\mathcal{S}$ i jest określone jako zbiór par:

$$\mathcal{S} = \big\{(0,1),\ (1,2),\ (2,3),\ (3,4),\ (5,6),\ \ldots\big\},$$

czyli:

$$\mathcal{S} = \big\{(x,y)\colon\ x,y \in \mathbb{N}\ \wedge\ x \prec y\big\},$$

gdzie $\prec$ oznacza relację następnika w zbiorze liczb naturalnych. Każda liczba naturalna z wyjątkiem zera jest następnikiem innej liczby naturalnej. Innymi słowy, dla każdej liczby naturalnej z wyjątkiem zera istnieje dokładnie jedna liczba, której ta jest następnikiem. Zatem można zdefiniować **funkcję poprzednika** $\mathcal{P}$, która nie przyjmuje wartości dla zera i jest zbiorem par postaci:

$$\mathcal{P} = \big\{(1,0),\ (2,1),\ (3,2),\ (4,3),\ (5,4),\ \ldots\big\},$$

czyli:

$$\mathcal{P} = \big\{(x,y)\colon\ x,y \in \mathbb{N}\ \wedge\ y \prec x\big\}.$$

Mając dane pewne funkcje, można konstruować odwzorowania bardziej skomplikowane. Na przykład, gdy znane są n-argumentowe funkcje $f_1, f_2, \ldots, f_m$ oraz m-argumentowa funkcja g, to można określić n-argumentową funkcję h jako zbiór powstały poprzez **operację podstawienia**, co zapisuje się jako:

$$h(\bar{x}) = g\big(f_1(\bar{x}), f_2(\bar{x}), \ldots, f_m(\bar{x})\big),$$

gdzie $\bar{x}$ jest wektorem liczb naturalnych postaci $(x_1, x_2, \ldots, x_n)$. Taka definicja funkcji wydaje się równie nieformalna jak wyżej przedstawione przykłady, ale jest to mylne odczucie, gdyż jest to **ścisła relacja pomiędzy zbiorami** $g, f_1, f_2, \ldots, f_m$, która musi być rozumiana następująco. Wektor $(\bar{x}, z)$ należy do zbioru h wtedy i tylko wtedy, gdy istnieją takie liczby $y_1, y_2, \ldots, y_n$, że wektor $(y_1, y_2, \ldots, y_m, z)$ jest elementem zbioru g oraz dla każdego $i \in \{1, 2, \ldots, m\}$ wektor $(\bar{x}, y_i)$ jest elementem zbioru f_i, czyli:

$$h = \big\{(\bar{x}, z)\colon\ \bar{x} \in \mathbb{N}^n \wedge\ y \in \mathbb{N}\ \wedge\ \exists\, z_1, z_2, \ldots, z_m \in \mathbb{N}\colon \big(((z_1, z_2, \ldots, z_m),\ y) \in g\ \wedge\ \forall\, i \in \{1, 2, \ldots, m\}\colon\ (\bar{x}, z_i) \in f_i\big)\big\}. \quad (1)$$

W opisywaniu odwzorowań wieloargumentowych, często korzysta się z operacji polegającej na wybieraniu spośród wielu argumentów dokładnie jednego. Przykładem jest wykorzystanie w definicji funkcji Ackermanna liczby x jako jednego z dwóch argumentów funkcji. Aby sformalizować tę czynność, definiuje się n-argumentową **funkcję projekcji** $\mathcal{I}_k^n$, która każdemu wektorowi liczb naturalnych $(x_1, x_2, \ldots, x_n)$ przyporządkowuje liczbę x_k, gdzie $k \in \{1, 2, \ldots, n\}$, czyli:

$$\mathcal{I}_k^n = \big\{\big((x_1, x_2, \ldots, x_n),\ y\big)\colon\ x_1, x_2, \ldots, x_n, y \in \mathbb{N}\ \wedge\ y = x_k\big\}.$$

Jako przykład skomplikowanego podstawienia funkcji można wziąć warunek rekursji dla funkcji Ackermanna i zdefiniować nowe odwzorowanie h, które dla dowolnej pary (x, y) przyjmuje wartość:

$$h(x,y) = \mathcal{A}\big(x-1,\ \mathcal{A}(x,\ y-1)\big).$$

Aby zapisać tę definicję formalnie, należy zastąpić odejmowanie jedynki funkcją poprzednika:

$$h(x,y) = \mathcal{A}\big(\mathcal{P}(x),\ \mathcal{A}(x,\ \mathcal{P}(y))\big),$$

oraz wprowadzić funkcję projekcji w miejscach, gdzie wykorzystany jest tylko jeden z dwóch argumentów funkcji, czyli:

$$h(x,y) = \mathcal{A}\big(\mathcal{P}(\mathcal{I}_1^2(x,y)),\ \mathcal{A}(\mathcal{I}_1^2(x,y),\ \mathcal{P}(\mathcal{I}_2^2(x,y)))\big).$$

Aby funkcja h była zdefiniowana zgodnie z definicją operacji podstawienia, to otrzymane równanie należy zapisać w postaci:

$$h(x,y) = \mathcal{A}\big(f_1(x,y), f_2(x,y), f_3(x,y)\big),$$

gdzie funkcje f_1, f_2, f_3 są zdefinowane jako:

$$\begin{aligned}
f_1(x,y) &= \mathcal{P}\big(\mathcal{I}_1^2(x,y)\big),\\
f_2(x,y) &= \mathcal{A}\big(\mathcal{I}_1^2(x,y),\ f_4(x,y)\big),\\
f_3(x,y) &= \mathcal{P}\big(\mathcal{I}_2^2(x,y)\big),
\end{aligned}$$

a funkcja f_4 jako:

$$f_4(x,y) = \mathcal{P}\big(\mathcal{I}_1^2(x,y)\big).$$

Wykonując w stosunku do definicji funkcji Ackermanna, w której użyto klamerki, analizę analogiczną do powyższej, powstaje równanie postaci:

$$\mathcal{A}(x,y) = \begin{cases} f_1(x,y), & \text{gdy } \ \mathcal{I}_1^2(x,y) = 0,\\ f_2(x,y), & \text{gdy } \ \mathcal{I}_2^2(x,y) = 0,\\ f_6(x,y), & \text{gdy } \ f_5(x,y) = 0, \end{cases}$$

gdzie funkcje $f_1, f_2, \ldots, f_6$ są określone przez operator podstawienia jako:

$$\begin{aligned}
f_1(x,y) &= \mathcal{S}\big(\mathcal{I}_2^2(x,y)\big),\\
f_2(x,y) &= \mathcal{A}\big(f_3(x,y), f_4(x,y)\big),\\
f_3(x,y) &= \mathcal{P}\big(\mathcal{I}_1^2(x,y)\big),\\
f_4(x,y) &= \mathcal{S}\big(f_5(x,y)\big),\\
f_5(x,y) &= \mathcal{Z}\big(\mathcal{I}_1^2(x,y)\big),\\
f_6(x,y) &= \mathcal{A}\big(f_3(x,y), f_7(x,y)\big),\\
f_7(x,y) &= \mathcal{A}\big(\mathcal{I}_1^2(x,y), f_8(x,y)\big),\\
f_8(x,y) &= \mathcal{P}\big(\mathcal{I}_2^2(x,y)\big).
\end{aligned}$$

Zamieniając wszystkie skomplikowane złożenia funkcji tak jak powyżej, można każdą definicję n-argumentowej funkcji h, powstałą poprzez zastosowanie klamerki, zredukować do postaci:

$$h(\bar{x}) = \begin{cases} f_1(\bar{x}), & \text{gdy } \ g_1(\bar{x}) = 0, \\ f_2(\bar{x}), & \text{gdy } \ g_2(\bar{x}) = 0, \\ \dots & \\ f_m(\bar{x}), & \text{gdy } \ g_m(\bar{x}) = 0, \end{cases}$$

gdzie $\bar{x}$ jest wektorem liczb naturalnych postaci $(x_1, x_2, \dots, x_n)$. Taka konstrukcja będzie nazywana **operacją wyboru** i należy ją interpretować następująco. Jeśli $g_1(\bar{x})$ jest równe 0, to $h(\bar{x})$ jest równe $f_1(\bar{x})$. W przeciwnym wypadku, jeśli $g_2(\bar{x})$ jest równe 0, to $h(\bar{x})$ jest równe $f_2(\bar{x})$, a jeśli $g_3(\bar{x})$ jest równe 0, to $h(\bar{x})$ jest równe $f_3(\bar{x})$. W ten sposób rozpatrywane są kolejne przypadki zapisu. Funkcja h nie jest określona w punkcie $\bar{x}$, jeśli wszystkie funkcje $g_1, g_2, \dots, g_m$ przyjmują wartości różne od zera w tym punkcie.

Opis operacji wyboru może wydawać się nieprecyzyjny i niewystarczająco formalny, chociażby dlatego, że niektóre funkcje mogą nie mieć wartości w punkcie $\bar{x}$. Takie wątpliwości są bezpodstawne, gdyż operacja wyboru określa **relację pomiędzy zbiorami** $h, g_1, g_2, \dots, g_m, f_1, f_2, \dots, f_m$. Ścisła interpretacja jest następująca. Para $(\bar{x}, y)$ należy do zbioru h wtedy i tylko wtedy, gdy istnieje taki indeks $k \in \{1, 2, \dots, m\}$, że para $(\bar{x}, y)$ należy do zbioru f_k oraz para $(\bar{x}, 0)$ należy do zbioru g_k oraz dla każdego $i \in \{1, 2, \dots, k-1\}$ istnieje takie z różne od zera, że para $(\bar{x}, z)$ należy do zbioru g_k. Zatem funkcja h jest takim zbiorem, że:

$$h = \big\{(\bar{x}, y)\colon\ \bar{x} \in \mathbb{N}^n \ \wedge\ y \in \mathbb{N} \ \wedge\ \exists\, k \in \{1, 2, \dots, m\}\colon\ \big((\bar{x}, y) \in f_k \quad \wedge \\ (\forall\, i \in \{1, 2, \dots, k\}\colon\ \exists\, z \in \mathbb{N}\colon\ (i = k \Leftrightarrow z = 0) \ \wedge\ (\bar{x}, z) \in g_k)\big)\big\}.$$

Algorytm obliczania funkcji zdefiniowanej poprzez operację wyboru jest następujący. Niech parametr i zmienia się w zakresie $1, 2, \dots, m$. W pierwszym kroku algorytmu oblicza się liczbę $g_i(\bar{x})$. Jeśli ta liczba nie istnieje, to nie istnieje również $h(\bar{x})$. Jeśli $g_i(\bar{x})$ jest różne od zera, to inkrementuje się i i powtarza cały alborytm. Gdy $g_i(\bar{x})$ jest równe zero, to oblicza się wartość $f_i(\bar{x})$. Jeśli ta liczba nie istnieje, to nie istnieje również $h(\bar{x})$. W przeciwnym wypadku za wartość $h(\bar{x})$ przyjmuje się $f(\bar{x})$, co kończy obliczenie.

2.4. Operatory funkcyjne

N-argumentowy **operator funkcyjny** ψ jest takim odwzorowaniem, które wektorowi funkcji liczb naturalnych przyporządkowuje funkcję liczb naturalnych. Operator funkcyjny jest więc zbiorem par:

$$\psi = \big\{\big((f_1, f_2, \dots, f_n),\ h\big)\colon\ f_1, f_2, \dots, f_n, h \text{ – funkcje liczb naturalnych}\big\}.$$

Operacja podstawienia omawiana wcześniej ma tę właściwość, że dla danych funkcji $f_1, f_2, \ldots, f_m$ i g istnieje dokładnie jedna funkcja h, dla której prawdziwe jest zdanie:

$$h(\bar{x}) = g\big(f_1(\bar{x}), f_2(\bar{x}), \ldots, f_m(\bar{x})\big).$$

Operacja ta jest więc odwzorowaniem, które funkcjom przyporządkowuje funkcję, czyli jest operatorem funkcyjnym. Operator ten będzie nazywany **operatorem podstawienia** i oznaczany symbolem α^m, gdzie m jest związane z liczbą argumentów operatora. Biorąc pod uwagę, że operator jest funkcją, która działa na funkcjach liczb naturalnych, można zapisać:

$$\alpha^m(g, f_1, f_2, \ldots, f_m) = h,$$

lub bardziej formalnie w postaci mnogościowej:

$$\big((g, f_1, f_2, \ldots, f_m),\ h\big) \in \alpha^m.$$

Ponieważ mając dowolne funkcje $f_1, f_2, \ldots, f_m, g_1, g_2, \ldots, g_m$, w wyniku operacji wyboru otrzymuje się dokładnie jedną funkcję h, to operacja ta jest operatorem funkcyjnym, który będzie nazywany **operatorem wyboru** β^m, gdzie m jest związane z liczbą argumentów. Dzięki temu nieformalny zapis z użyciem klamerki:

$$h(\bar{x}) = \begin{cases} f_1(\bar{x}), & \text{gdy } \ g_1(\bar{x}) = 0, \\ f_2(\bar{x}), & \text{gdy } \ g_2(\bar{x}) = 0, \\ \ldots & \\ f_m(\bar{x}), & \text{gdy } \ g_m(\bar{x}) = 0, \end{cases}$$

może być przedstawiony jako:

$$\beta^m(f_1, g_1, f_2, g_2, \ldots, f_m, g_m) = h,$$

lub jeszcze lepiej w postaci mnogościowej:

$$\big((f_1, g_1, f_2, g_2, \ldots, f_m, g_m),\ h\big) \in \beta^m.$$

N-argumentowy **operator stały** $\delta^n_{\mathcal{F}}$ jest takim przyporządkowaniem, które wektorowi funkcji $(f_1, f_2, \ldots, f_n)$ przyporządkowuje funkcję $\mathcal{F}$, czyli:

$$\delta^n_{\mathcal{F}}(f_1, f_2, \ldots, f_n) = \mathcal{F}$$

lub formalnie:

$$\big((f_1, f_2, \ldots, f_n),\ \mathcal{F}\big) \in \delta^n_{\mathcal{F}}.$$

W dalszym toku wywodu dużą rolę odgrywają operatory stałe dla funkcji pierwotnych, czyli $\delta^n_{\mathcal{S}}$, $\delta^n_{\mathcal{P}}$, $\delta^n_{\mathcal{Z}}$, $\delta^n_{\mathcal{I}^m_k}$.

N-argumentowy **operator projekcji** γ_k^m, gdzie $k \in \{1, 2, \ldots, m\}$, wektorowi funkcji $(f_1, f_2, \ldots, f_m)$ przyporządkowuje funkcję f_k, czyli:

$$\gamma_k^m(f_1, f_2, \ldots, f_m) = f_k,$$

co jest równoważne zapisowi:

$$\big((f_1, f_2, \ldots, f_m),\ f_k\big) \in \gamma_k^m.$$

Złożenie operatorów funkcyjnych polega na tym, że n-argumentowy operator funkcyjny π jest w takiej relacji m-argumentowego operatora σ oraz n-argumentowych operatorów $\kappa_1, \kappa_2, \ldots, \kappa_m$, że dla dowolnego wektora $\bar{f}$ postaci $(f_1, f_2, \ldots, f_k)$, gdzie $f_1, f_2, \ldots, f_k$ są funkcjami, zachodzi:

$$\pi(\bar{f}) = \sigma\big(\kappa_1(\bar{f}), \kappa_2(\bar{f}), \ldots, \kappa_m(\bar{f})\big).$$

Wynikiem złożenia operatorów funkcyjnych $\sigma, \kappa_1, \kappa_2, \ldots, \kappa_m$ jest operator funkcyjny π. Precyzyjnie mówiąc, para postaci $(\bar{f}, h)$, gdzie $\bar{f}$ jest wektorem funkcji, a h jest funkcją, należy do zbioru π wtedy i tylko wtedy, gdy istnieją takie funkcje $g_1, g_2, \ldots, g_m$, że para $\big((g_1, g_2, \ldots, g_m), h\big)$ należy do zbioru σ oraz dla dowolnego $i \in \{1, 2, \ldots, m\}$ para $(\bar{f}, g_i)$ należy do zbioru κ_i, czyli:

$$\begin{aligned}\pi = \big\{(\bar{f}, h)\colon\ & \bar{f} = (f_1, f_2, \ldots, f_n)\ \wedge\ f_1, f_2, \ldots, f_n, h \text{ – funkcje}\ \wedge \\ & \exists\, g_1, g_2, \ldots, g_m\colon\ \big(g_1, g_2, \ldots, g_m \text{ – funkcje}\ \wedge \\ & ((g_1, g_2, \ldots, g_m),\ h) \in \sigma\ \wedge\ \forall\, i \in \{1, 2, \ldots, m\}\colon\ (\bar{f}, g_i) \in \kappa_i\big)\big\}.\end{aligned}$$

2.5. Definicja funkcji normalnej

W niniejszym podrozdziale przedstawiona jest definicja zbioru funkcji normalnych. W przeciwieństwie do systemu Hilberta, gdzie zbiór funkcji rekursywnych jest generowany poprzez użycie operatorów działających na wcześniej określonych funkcjach rekursywnych, każda funkcja normalna jest opisywana niezależnie od innych funkcji normalnych.

Wychodząc od najprostszych operatorów zdefiniowanych wcześniej, można budować odwzorowania bardziej skomplikowane – odwzorowania, które również są operatorami funkcyjnymi. Niektóre z nich będą nazywane **operatorami normalnymi** i są definiowane następująco. Operator powstały w wyniku złożenia operatorów normalnych jest operatorem normalnym. Operatory podstawienia α^m, wyboru β^m, funkcji pierwotnych $\delta_{\mathcal{S}}^m$, $\delta_{\mathcal{P}}^m$, $\delta_{\mathcal{Z}}^m$, $\delta_{\mathcal{I}_k^n}^m$ i projekcji γ_k^m są operatorami normalnymi.

Normalnym równaniem funkcyjnym dla i-tej współrzędnej z niewiadomą $(g_1, g_2, \ldots, g_m)$ i operatorem funkcyjnym ψ^m nazywane jest zdanie postaci:

$$((g_1, g_2, \ldots, g_m),\ g_i) \in \psi^m,$$

gdzie $i \in \{1, 2, \ldots, m\}$, a ψ^m jest normalnym m-argumentowym operatorem funkcyjnym. Jeśli podstawiając za niewiadomą wektor funkcji $(f_1, f_2, \ldots, f_m)$, otrzymuje się zdanie prawdziwe, to wektor ten jest nazywany **rozwiązaniem równania**.

Normalnym układem równań z niewiadomą $(g_1, g_2, \ldots, g_m)$ i operatorami $(\psi_1, \psi_2, \ldots, \psi_k)$ nazywany jest ciąg k równań funkcyjnych, którego i-ty wyraz, gdzie $i \in \{1, 2, \ldots, k\}$, jest normalnym równaniem funkcyjnym dla i-tej współrzędnej, czyli ma postać:

$$\big((g_1, g_2, \ldots, g_m),\ g_i\big) \in \psi_i.$$

Wektor $(f_1, f_2, \ldots, f_m)$ jest nazywany **rozwiązaniem układu równań**, jeśli jest rozwiązaniem każdego równania wchodzącego w jego skład.

Jeśli dany układ równań funkcyjnych jest normalny i ma dokładnie jedno rozwiązanie postaci $(f_1, f_2, \ldots, f_m)$, to znaczy, że jest on **normalną definicją funkcji** $f_1, f_2, \ldots, f_m$. Funkcja f jest **funkcją normalną**, jeśli istnieje normalna definicja tej funkcji.

Definicja funkcji normalnej wydaje się znacznie bardziej skomplikowana niż system Hilberta. Wynika to stąd, że trudność definiowania funkcji została przeniesiona na teorię samej metody definiowania. Łatwiej jest opisać odwzorowanie obliczalne jako funkcję normalną niż jako funkcję rekursywną przy użyciu równań Hilberta. Z drugiej strony w systemie Hilberta bardzo łatwo wykazać, że zdefiniowana funkcja jest rekursywna, podczas gdy formalna analiza definicji funkcji normalnej wymaga żmudnego dowodzenia.

3. Przykłady funkcji normalnych

3.1. Funkcje pierwotne

Definicja funkcji normalnych nie odnosi się do funkcji pierwotnych wprost, w przeciwieństwie do systemu Hilberta, gdyż na podstawie definicji można udowodnić, że funkcje pierwotne są normalne. Każda funkcja pierwotna f jest funkcją normalną, gdyż definicją takiej funkcji jest układ równań zawierający tylko jedno równanie:

$$\big((f), f\big) \in \delta^1_f.$$

3.2. Funkcja Ackermanna

Niniejsza część rozważań zawiera ponowną analizę funkcji Ackermanna i stanowi dowód, iż funkcja ta jest normalna. W pierwszej kolejności zostanie fomalnie przeanalizowany opis funkcji Ackermanna przedstawiony wcześniej. Ów opis zawiera równania z użyciem symboli $\mathcal{A}$, f_1, $f_2, \ldots, f_8$, $\mathcal{S}$, $\mathcal{P}$, $\mathcal{Z}$, $\mathcal{I}^2_1$, $\mathcal{I}^2_2$. Pierwsze z nich wykorzystuje operator wyboru, którego argumentami są funkcje f_1, f_2, f_5, f_6, $\mathcal{I}^2_1$, $\mathcal{I}^2_2$, a wartością $\mathcal{A}$. Mnogościowo można ten fakt wyrazić w postaci:

$$\big((f_1, \mathcal{I}^2_1, f_2, \mathcal{I}^2_2, f_6, f_5),\ \mathcal{A}\big) \in \beta^3.$$

Jest to równanie funkcyjne, którego niewiadomymi są funkcje $\mathcal{A}$, f_1, f_2, f_5, f_6. Pozostałe równania opisu również należy zapisać formalnie i wówczas przybierają one postać:

$$
\begin{aligned}
\big((\mathcal{S}, \mathcal{I}_2^2),\ f_1\big) &\in \alpha^1,\\
\big((\mathcal{A}, f_3, f_4),\ f_2\big) &\in \alpha^2,\\
\big((\mathcal{P}, \mathcal{I}_1^2),\ f_3\big) &\in \alpha^1,\\
\big((\mathcal{S}, f_5),\ f_4\big) &\in \alpha^1,\\
\big((\mathcal{Z}, \mathcal{I}_1^2),\ f_5\big) &\in \alpha^1,\\
\big((\mathcal{A}, f_3, f_7),\ f_6\big) &\in \alpha^2,\\
\big((\mathcal{A}, \mathcal{I}_1^2, f_8),\ f_7\big) &\in \alpha^2,\\
\big((\mathcal{P}, \mathcal{I}_2^2),\ f_8\big) &\in \alpha^1.
\end{aligned}
$$

Wszystkie równania tworzą układ równań funkcyjnych, w którym niewiadomą jest wektor funkcji $\bar{f}$ zdefiniowany jako:

$$\bar{f} = (\mathcal{A}, f_1, f_2, \ldots, f_8).$$

Aby ten układ równań był zgodny z formalną definicją, należy go odpowiednio zmodyfikować. Najpierw należy wszystkie niewiadome zastąpić odpowiednimi konstrukcjami z udziałem operatora projekcji, czyli:

$$
\begin{aligned}
\mathcal{A} &= \gamma_1^9(\bar{f}),\\
f_1 &= \gamma_2^9(\bar{f}),\\
f_2 &= \gamma_3^9(\bar{f}),\\
&\ldots\\
f_8 &= \gamma_9^9(\bar{f}),\\
\mathcal{S} &= \delta_{\mathcal{S}}^9(\bar{f}),\\
\mathcal{P} &= \delta_{\mathcal{P}}^9(\bar{f}),\\
\mathcal{Z} &= \delta_{\mathcal{Z}}^9(\bar{f}),\\
\mathcal{I}_1^2 &= \delta_{\mathcal{I}_1^2}^9(\bar{f}),\\
\mathcal{I}_2^2 &= \delta_{\mathcal{I}_2^2}^9(\bar{f}).
\end{aligned}
$$

Następnie trzeba zdefiniować nowe operatory poprzez złożenia operatorów funkcji pierwotnych i operatora projekcji. Dla poszczególnych równań są to takie operatory, że dla dowolnego wektora funkcji $\bar{f}$ zachodzi:

$$
\begin{aligned}
\pi_1(\bar{f}) &= \beta^3(\gamma_2^9(\bar{f}),\ \delta_{\mathcal{I}_1^2}^9(\bar{f}),\ \gamma_3^9(\bar{f}),\ \delta_{\mathcal{I}_2^2}^9(\bar{f}),\ \gamma_7^9(\bar{f}),\ \gamma_6^9(\bar{f})),\\
\pi_2(\bar{f}) &= \alpha^1(\delta_{\mathcal{S}}^9(\bar{f}),\ \delta_{\mathcal{I}_2^2}^9(\bar{f}),\ \gamma_2^9(\bar{f})),\\
\pi_3(\bar{f}) &= \alpha^2(\gamma_1^9(\bar{f}),\ \gamma_4^9(\bar{f}),\ \gamma_5^9(\bar{f})),\\
\pi_4(\bar{f}) &= \alpha^1(\delta_{\mathcal{P}}^9(\bar{f}),\ \delta_{\mathcal{I}_1^2}^9(\bar{f})),\\
\pi_5(\bar{f}) &= \alpha^1(\delta_{\mathcal{S}}^9(\bar{f}),\ \gamma_6^9(\bar{f}),\ \gamma_5^9(\bar{f})),\\
\pi_6(\bar{f}) &= \alpha^1(\delta_{\mathcal{Z}}^9(\bar{f}),\ \delta_{\mathcal{I}_1^2}^9(\bar{f}),\ \gamma_6^9(\bar{f})),
\end{aligned}
$$

$$\pi_7(\bar{f}) = \alpha^2(\gamma_1^9(\bar{f}),\ \gamma_4^9(\bar{f}),\ \gamma_8^9(\bar{f})),$$
$$\pi_8(\bar{f}) = \alpha^2(\gamma_1^9(\bar{f}),\ \delta_{\mathcal{I}_1^2}^9(\bar{f}),\ \gamma_9^9(\bar{f})),$$
$$\pi_9(\bar{f}) = \alpha^1(\delta_{\mathcal{P}}^9(\bar{f}),\ \delta_{\mathcal{I}_2^2}^9(\bar{f})).$$

Wprowadzając modyfikacje do wszystkich równań, cały układ przybiera postać:

$$\big((\mathcal{A}, f_1, f_2, \ldots, f_9),\ \mathcal{A}\big) \in \pi_1,$$
$$\big((\mathcal{A}, f_1, f_2, \ldots, f_9),\ f_1\big) \in \pi_2,$$
$$\big((\mathcal{A}, f_1, f_2, \ldots, f_9),\ f_2\big) \in \pi_3,$$
$$\big((\mathcal{A}, f_1, f_2, \ldots, f_9),\ f_3\big) \in \pi_4,$$
$$\big((\mathcal{A}, f_1, f_2, \ldots, f_9),\ f_4\big) \in \pi_5,$$
$$\big((\mathcal{A}, f_1, f_2, \ldots, f_9),\ f_5\big) \in \pi_6,$$
$$\big((\mathcal{A}, f_1, f_2, \ldots, f_9),\ f_6\big) \in \pi_7,$$
$$\big((\mathcal{A}, f_1, f_2, \ldots, f_9),\ f_7\big) \in \pi_8,$$
$$\big((\mathcal{A}, f_1, f_2, \ldots, f_9),\ f_8\big) \in \pi_9.$$

Aby wykazać, że powyższy układ równań jest ścisłą normalną definicją funkcji Ackermanna, należy udowodnić, że jest to normalny układ równań oraz ma on dokładnie jedno rozwiązanie, w którym funkcja $\mathcal{A}$ jest funkcją Ackermanna. Dowód jest łatwy do wykonania, ale wykracza poza zakres niniejszej publikacji. Wszystkie operatory $\pi_1, \pi_2, \ldots, \pi_9$ są normalne, gdyż są złożeniami operatorów stałych dla funkcji pierwotnych i operatora projekcji. Dowód, że układ ten ma jedno rozwiązanie, można oprzeć na indukcji matematycznej oraz nieformalnym opisie funkcji Ackermanna z użyciem klamerki. Następnie tok rozumowania należy przełożyć na formalne zależności pomiędzy zbiorami.

Dla każdej funkcji normalnej istnieje wiele normalnych definicji, gdyż można stosować dowolne złożenia operatorów. Im operatory są prostsze tym więcej równań definiuje funkcję. Poprzednio funkcja Ackermanna była zdefiniowana przy użyciu dziewięciu operatorów, ale można przedstawić definicję z jednym bardziej złożonym operatorem. Punktem wyjścia są wcześniej przedstawione równania, które można zapisać w postaci:

$$\mathcal{A} = \beta^3(f_1, \mathcal{I}_1^2, f_2, \mathcal{I}_2^2, f_6, f_5),$$
$$f_1 = \alpha^1(\mathcal{S}, \mathcal{I}_2^2),$$
$$f_2 = \alpha^2(\mathcal{A}, f_3, f_4),$$
$$f_3 = \alpha^1(\mathcal{P}, \mathcal{I}_1^2),$$
$$f_4 = \alpha^1(\mathcal{S}, f_5),$$
$$f_5 = \alpha^1(\mathcal{Z}, \mathcal{I}_1^2),$$

$$
\begin{aligned}
f_6 &= \alpha^2(\mathcal{A}, f_3, f_7),\\
f_7 &= \alpha^2(\mathcal{A}, \mathcal{I}_1^2, f_8),\\
f_8 &= \alpha^1(\mathcal{P}, \mathcal{I}_2^2).
\end{aligned}
$$

Ostatnie dwa równania można zredukować poprzez podstawienie, otrzymując zależność:

$$f_7 = \alpha^2\big(\mathcal{A}, \mathcal{I}_1^2,\ \alpha^1(\mathcal{P}, \mathcal{I}_2^2)\big).$$

Korzystając z tego równania oraz równania dla funkcji f_3, można zapisać równanie: dla funkcji f_6 w postaci

$$f_6 = \alpha^2\big(\mathcal{A},\ \alpha^1(\mathcal{P}, \mathcal{I}_1^2),\ \alpha^2(\mathcal{A}, \mathcal{I}_1^2,\ \alpha^1(\mathcal{P}, \mathcal{I}_2^2))\big).$$

W dalszej kolejności można zredukować równania dla funkcji f_2, f_3, f_4, f_5, otrzymując:

$$f_2 = \alpha^2(\mathcal{A},\ \alpha^1(\mathcal{P}, \mathcal{I}_1^2),\ \alpha^1(\mathcal{S},\ \alpha^1(\mathcal{Z}, \mathcal{I}_1^2))).$$

Ostatecznie podstawiając otrzymane zależności do pierwszego równania, powstaje zależność:

$$
\begin{aligned}
\mathcal{A} = \beta^3(&\alpha^1(\mathcal{S}, \mathcal{I}_2^2),\\
&\mathcal{I}_1^2,\\
&\alpha^2\big(\mathcal{A},\ \alpha^1(\mathcal{P}, \mathcal{I}_1^2),\ \alpha^1(\mathcal{S},\ \alpha^1(\mathcal{Z}, \mathcal{I}_1^2))\big),\\
&\mathcal{I}_2^2,\\
&\alpha^2\big(\mathcal{A},\ \alpha^1(\mathcal{P}, \mathcal{I}_1^2),\ \alpha^2(\mathcal{A}, \mathcal{I}_1^2,\ \alpha^1(\mathcal{P}, \mathcal{I}_2^2)),\\
&\alpha^1(\mathcal{Z}, \mathcal{I}_1^2)\big).
\end{aligned}
$$

Niech π będzie takim operatorem funkcyjnym, że dla wektora funkcji postaci (f), czyli wektora z jedną współrzędną f, przyjmuje wartość:

$$
\begin{aligned}
\pi(f) = \beta^3(&\alpha^1(\delta^1_{\mathcal{S}}(f), \delta^1_{\mathcal{I}_2^2}(f)),\\
&\delta^1_{\mathcal{I}_1^2}(f),\\
&\alpha^2(\gamma^1_1(f),\ \alpha^1(\delta^1_{\mathcal{P}}(f), \delta^1_{\mathcal{I}_1^2}(f),\ \alpha^1(\delta^1_{\mathcal{S}}(f),\ \alpha^1(\delta^1_{\mathcal{Z}}(f), \delta^1_{\mathcal{I}_1^2}(f))))),\\
&\delta^1_{\mathcal{I}_2^2}(f),\\
&\alpha^2(\gamma^1_1(f),\ \alpha^1(\delta^1_{\mathcal{P}}(f), \delta^1_{\mathcal{I}_1^2}(f)),\ \alpha^2(\gamma^1_1(f, \delta^1_{\mathcal{I}_1^2}(f),\ \alpha^1(\delta^1_{\mathcal{P}}(f), \delta^1_{\mathcal{I}_2^2}(f))),\\
&\alpha^1(\delta^1_{\mathcal{Z}}(f), \delta^1_{\mathcal{I}_1^2}(f))).
\end{aligned}
$$

Wówczas definicja funkcji Ackermanna jest układem równań z jednym operatorem π postaci:

$$\big((\mathcal{A}),\ \mathcal{A}\big) \in \pi.$$

3.3. Podstawowe działania arytmetyczne

Poniżej zostaną przedstawione przykłady definicji funkcji normalnych. Ze względu na dużą liczbę przykładów, funkcje nie będą identyfikowane przy użyciu pojedynczej litery, ale poprzez całe nazwy. Nie tracąc czytelności i prostoty opisu funkcje będą opisywane równaniami z użyciem klamerki, w których argumenty będą zapisywane wprost bez użycia funkcji projekcji, a pozostałe obliczenia takie, jak dodawanie lub odejmowanie będą oznaczane przy użyciu wcześniej zdefiniowanych funkcji. Udowodnienie, iż dane odwzorowanie jest funkcją normalną, należy przeprowadzić podobnie jak dla funkcji Akcermanna, wprowadzając do definicji wszystkie potrzebne równania.

Dzieci rozpoczynają naukę matematyki od dodawania, gdyż jest to najprostsze działanie, które polega na inkrementacji pierwszego argumentu tyle razy, ile zapisano w drugim argumencie, czyli:

$$\mathrm{add}(x,y) = \begin{cases} x, & \text{gdy } \ y = 0, \\ 1 + \mathrm{add}(x, y-1) & \text{w przeciwnym wypadku,} \end{cases}$$

czyli:

$$\mathrm{add}(x,y) = \begin{cases} 1, & \text{gdy } \ y = 0, \\ \mathcal{S}(\mathrm{add}(x, \mathcal{P}(y))), & \text{gdy } \ \mathcal{Z}(x) = 0. \end{cases}$$

Aby określić funkcję dodawania formalnie to najpierw należy wprowadzić funkcję projekcji, obliczyć liczbę 1 oraz oznaczyć wektor (x, y) symbolem $\bar{f}$ otrzymując równanie:

$$\mathrm{add}(\bar{f}) = \begin{cases} \mathcal{S}(\mathcal{Z}(\mathcal{I}_1^2(\bar{f}))), & \text{gdy } \ \mathcal{I}_2^2(\bar{f}) = 0, \\ \mathcal{S}(\mathrm{add}(\mathcal{I}_1^2(\bar{f}),\, \mathcal{P}(\mathcal{I}_1^2(\bar{f})))), & \text{gdy } \ \mathcal{Z}(\mathcal{I}_1^2(\bar{f})) = 0, \end{cases}$$

które potem należy przekształcić w układ równań:

$$\begin{aligned} \mathrm{add} &= \beta^2(f_1, \mathcal{I}_2^2, f_2, f_3), \\ f_1 &= \alpha^1(\mathcal{S}, f_4), \\ f_2 &= \alpha^1(\mathcal{S}, f_5), \\ f_4 &= \alpha^1(\mathcal{Z}, \mathcal{I}_1^2), \\ f_5 &= \alpha^2(\mathrm{add}, \mathcal{I}_1^2, f_6), \\ f_6 &= \alpha^1(\mathcal{P}(\mathcal{I}_2^2)). \end{aligned}$$

Działanie mnożenia jest rozumiane intuicyjnie jako wielokrotne dodawanie, zatem definicja mnożenia jako funkcji normalnej może być określona równaniem:

$$\mathrm{mul}(x,y) = \begin{cases} 0, & \text{gdy } \ y = 0, \\ \mathrm{add}(x, mul(x, \mathcal{P}(y))), & \text{gdy } \ \mathcal{Z}(x) = 0. \end{cases}$$

W systemie Hilberta mnożenie jest określane równie prosto. Jednak w przypadku, gdy funkcja korzysta z bardziej zaawansowanego dodawania, to używanie systemu

Hilberta może być kłopotliwe, podczas gdy funkcje normalne wciąż pozwalają na łatwe przedstawianie definicji. Na przykład w ciągu Fibonacciego kolejny wyraz jest sumą dwóch poprzednich, więc:

$$\mathrm{fib}(x,y) = \begin{cases} 1, & \text{gdy } \ x = 0, \\ 1, & \text{gdy } \ \mathcal{P}(x) = 0, \\ \mathrm{add}\big(\mathrm{fib}(\mathcal{P}(x)),\ \mathrm{fib}(\mathcal{P}(\mathcal{P}(x)))\big), & \text{gdy } \ \mathcal{Z}(x) = 0. \end{cases}$$

W nieformalnej definicji funkcji Fibonacciego w pierwszej linii operatora wyboru zapisano warunek określający przynależność argumentu n do zbioru $0, 1$. W definicji formalnej zapisano ten fakt w dwóch liniach. Jeśli warunek obliczenia funkcji będzie bardziej skomplikowanym zbiorem, to takie wyspecyfikowanie wartości może być trudne i nienaturalne w zapisie. Dlatego stosuje się funkcje charakterystyczne zbiorów – funkcje, które dla elementu z danego zbioru przyjmują wartość zero, a dla elementów spoza zbioru wartości większe od zera. Aby budowanie takich funkcji było możliwe, należy odwzorować operacje łączenia zbiorów, części wspólnej i dopełnienia za pomocą operatorów boolowskich. Negacja jest najprostsza, gdyż może być opisana równaniem:

$$\mathrm{not}(x) = \begin{cases} 1, & \text{gdy } \ x = 0, \\ 0, & \text{gdy } \ \mathcal{Z}(x) = 0. \end{cases}$$

Koniunkcja ma wartość zerową, jeśli wszystkie argumenty są różne od zera, czyli:

$$\mathrm{and}(x,y) = \begin{cases} 0 & \text{dla } \ x = 0, \\ 0 & \text{dla } \ y = 0, \\ 1 & \text{dla } \ \mathcal{Z}(x) = 0. \end{cases}$$

Mając funkcję negacji i koniunkcji, z łatwością wyznacza się alternatywę:

$$\mathrm{or}(x,y) = \mathrm{not}\big(\mathrm{and}(\mathrm{not}(x),\ \mathrm{not}(y))\big).$$

Określanie funkcji charakterystycznych zbioru może wymagać użycia funkcji, które porównują liczby naturalne. Na przykład funkcja equ określona równaniem:

$$\mathrm{equ}(x,y) = \begin{cases} y & \text{dla } \ x = 0, \\ x & \text{dla } \ y = 0, \\ \mathrm{equ}(\mathcal{P}(x),\ \mathcal{P}(y)) & \text{dla } \ \mathcal{Z}(x) = 0, \end{cases}$$

ma wartość zero, jeżeli argumenty są równe, oraz wartość różną od zera, gdy argumenty są różne. Funkcja les ma wartość równą zero, jeśli pierwszy argument jest mniejszy od drugiego. Definiuje się ją następująco:

$$\mathrm{les}(x,y) = \begin{cases} 1 & \text{dla } \ y = 0, \\ 0 & \text{dla } \ x = 0, \\ \mathrm{les}(\mathcal{P}(x),\ \mathcal{P}(y)) & \text{dla } \ \mathcal{Z}(x) = 0. \end{cases}$$

Korzystając z powyższych odwzorowań, można zbiór $0,1$ określić przy użyciu funkcji charakterystycznej:

$$\mathrm{chr}_{\{0,1\}}(x) = \mathrm{les}(x, 2),$$

czyli innym sposobem na zdefiniowanie funkcji Fibonacciego jest użycie równania:

$$\mathrm{fib}(x, y) = \begin{cases} 1, & \text{gdy } \ \mathrm{chr}_{\{0,1\}}(x) = 0, \\ \mathrm{add}\big(\mathrm{fib}(\mathcal{P}(x)),\ \mathrm{fib}(\mathcal{P}(\mathcal{P}(x)))\big), & \text{gdy } \ \mathcal{Z}(x) = 0. \end{cases}$$

Operacja odejmowania jest przykładem odwzorowania, które nie przyjmuje wartości, jeśli pierwszy argument jest mniejszy od drugiego. W systemie funkcji normalnych odejmowanie opisuje się równaniem:

$$\mathrm{sub}(x, y) = \begin{cases} y & \text{dla } \ x = 0, \\ \mathrm{sub}(\mathcal{P}(x),\ \mathcal{P}(y)) & \text{dla } \ \mathcal{Z}(x) = 0. \end{cases}$$

Wydaje się, że taka postać definicji funkcji jest niedopuszczalna, gdyż w trakcie ustalania kolejnych wartości zachodzi potrzeba obliczenia poprzednika liczby zero, który nie istnieje. Jednak trzeba wziąć pod uwagę, że operator wyboru jest zdefiniowany na gruncie arytmetyki zbiorów, więc obliczenie sprowadza się do sprawdzenia, czy istnieje taka liczba z, że para $(0, z)$ należy do funkcji poprzednika. Ponieważ taka liczba nie istnieje, więc funkcja sub nie ma wartości dla danego argumentu.

Operacja dzielenia div może być zdefiniowana na podstawie funkcji odejmowania. Jednak opisanie obliczenia jako funkcji normalnej wymaga użycia dwóch równań:

$$\mathrm{div}(x, y) = \mathrm{aux}(x, y, 0)$$

$$\mathrm{aux}(x, y, z) = \begin{cases} z & \text{dla } \ x = 0 \\ \mathrm{aux}(\mathrm{sub}(x, y),\ y,\ \mathcal{S}(z)) & \text{dla } \ \mathcal{Z}(x) = 0. \end{cases}$$

Od pierwszego argumentu x odejmuje się drugi y, zwiększając za każdym razem wynik z dzielenia z. Gdy y nie jest dzielnikiem x, to funkcja nie jest określona dla tych argumentów.

3.4. Programowanie funkcyjne

Najstarszymi algorytmami z dziedziny matematyki jest znajdowanie wspólnego dzielnika według Euklidesa oraz znajdowanie liczb pierwszych według Eratostenesa. Obie te metody obliczeniowe przedstawia się w literaturze jako przepisy wykonywania pewnych czynności w celu ustalenia wyniku. Poniżej zdefiniowane zostaną funkcje normalne, które odwzorowują te algorytmy.

Reszta z dzielenia liczb x i y jest taką liczbą naturalną r, że dla pewnej liczby naturalnej k zachodzi:

$$x = \mathrm{add}\big(\mathrm{mul}(k, y),\ r\big) \ \wedge \ r < y.$$

Gdy y jest równe zero, to liczba k nie istnieje. Algorytm obliczania reszty z dzielenia jest następujący. Jeśli x jest mniejsze od y, to liczba x jest wynikiem obliczenia, gdyż dla k równego zero powyższe twierdzenie jest prawdziwe. W przeciwnym wypadku liczbę x należy pomniejszyć o y i ponownie sprawdzić porzedni warunek. Funkcje normalne pozwalają na łatwe kodowanie takich algorytmów. W rozważanym przypadku wystarczy zapisać:

$$\mathrm{mod}(x,y) = \begin{cases} x & \text{dla } \ \mathrm{les}(x,y) = 0, \\ \mathrm{mod}(\mathrm{sub}(x,y),\ y) & \text{dla } \ \mathcal{Z}(x) = 0. \end{cases}$$

Algorytm Euklidesa [16, 21] służy obliczaniu największego wspólnego dzielnika dwóch liczb naturalnych. Obliczenie polega na odejmowaniu reszty z dzielenia tych liczb od większej z nich, gdyż wspólny dzielnik jest również dzielnikiem tej reszty. Gdy jedna z liczb jest równa zero, to druga z nich jest poszukiwanym wynikiem:

$$\mathrm{nwd}(x,y) = \begin{cases} x & \text{dla } \ \mathrm{mod}(y,x) = 0. \\ \mathrm{nwd}(\mathrm{mod}(y,x),\ x) & \text{dla } \ \mathcal{Z}(x) = 0. \end{cases}$$

Liczba naturalna x jest pierwsza, jeśli ma dokładnie dwa podzielniki – jeden i samą siebie. Zatem dla dowolnej liczby naturalnej y prawdziwa jest implikacja:

$$1 < y \ \wedge \ y < x \quad \Longrightarrow \quad \mathrm{mod}(x,y) \neq 0.$$

Algorytm sprawdzający, czy dana liczba jest pierwsza, polega na testowaniu kolejnych wartości liczby y w zakresie 2 do $(x-1)$. Funkcja isp przyjmuje wartość 0, jeśli x jest liczbą pierwszą, a wartość 1 jeśli x jest liczbą złożoną. Dla x równego 0 funkcja nie przyjmuje żadnej wartości. Podana dalej konstrukcja wykorzystuje dodatkową funkcję aux, która sprawdza reszty z dzielenia liczby x przez kolejne wartości y i w przypadku, gdy reszta będzie równa zero, zwraca wartość 1, gdyż wtedy liczba x jest liczbą złożoną, a gdy x jest równe y, to kończy obliczenie z wynikiem pozytywnym – zwraca wartość 0. Ostatecznie algorytm testujący liczby pierwsze może być zdefiniowany następująco:

$$\mathrm{isp}(x) = \begin{cases} 0 & \text{dla } \ \mathcal{P}(x) = 0, \\ \mathrm{aux}(x,2) & \text{dla } \ \mathcal{Z}(x) = 0, \end{cases}$$

$$\mathrm{aux}(x,y) = \begin{cases} 0 & \text{dla } \ \mathrm{equ}(x,y) = 0, \\ 1 & \text{dla } \ \mathrm{mod}(x,y) = 0, \\ \mathrm{aux}(x,\ \mathcal{S}(y)) & \text{dla } \ \mathcal{Z}(x) = 0, \end{cases}$$

Sito Eratostenesa trzeba zdefiniować przy użyciu dwóch funkcji, z których pierwsza znajduje kolejne liczby pierwsze, a druga je odpowiednio numeruje. Poniższa funkcja nth dla danej liczby naturalnej n zwraca n-tą liczbę pierwszą, a pomocnicza funkcja aux dla zadanej liczby n znajduje najmniejszą liczbe pierwszą, która jest większa lub równa liczbie n. Liczba 1 jest liczbą pierwszą o numerze 1. Jeśli x jest n-tą liczbą pierwszą, to liczba pierwsza o numerze $(n+1)$ ma wartość $aux(x+1)$. Działanie

funkcji aux polega na testowaniu, czy n jest liczbą pierwszą przy użyciu wcześniej zdefiniowanej funkcji *isp*. Jeśli test okaże się pozytywny, to funkcja aux zwraca n jako wynik. W przeciwnym wypadku obliczenie wykonywane jest dla następnika liczby n. Zatem sito Eratostenesa można przedstawić w postaci układu równań:

$$\mathrm{nth}(n) = \begin{cases} 1 & \text{dla } \mathcal{P}(n) = 0, \\ \mathrm{aux}(\mathcal{S}(\mathrm{nth}(\mathcal{P}(n)))) & \text{dla } \mathcal{Z}(n) = 0, \end{cases}$$

$$\mathrm{aux}(n) = \begin{cases} n & \text{dla } \mathrm{isp}(n) = 0, \\ \mathrm{aux}(\mathcal{S}(n)) & \text{dla } \mathcal{Z}(n) = 0. \end{cases}$$

4. Podsumowanie

Niniejsze opracowanie przedstawia nową metodę definiowania funkcji, które są obliczalne. Metoda ta pozwala na łatwe zapisywanie funkcji, które w systemie Hilberta są trudne do zdefiniowania. Wadą rozwiązania jest złożona definicja formalna. Można powiedzieć, że trudność opisywania funkcji w systemie Hilberta została przeniesiona na definicję nowej metody.

Relację funkcji rekursywnych i funkcji normalnych można porównać do relacji maszyny Turinga i maszyny RAM. Pierwsze rozwiązanie charakteryzuje się prostą definicją formalną, co sprawia, że lepiej nadaje się do wywodów naukowych. Drugie rozwiązanie jest bliższe intuicyjnemu pojmowaniu obliczenia, jednak definicja formalna jest bardziej skomplikowana, czyli trudna w opisie formalnym.

Dalsze badania nad funkcjami normalnymi powinny ustalić związki nowego systemu z modelami istniejącymi w teorii obliczeń. Niektóre zależności są bardzo proste, inne wymagają dokładniejszej analizy. Do tych pierwszych można zaliczyć twierdzenie, że funkcje normalne są zbiorem nie mniejszym niż funkcje ogólnie rekursywne. Niech na przykład funkcja f będzie zdefiniowana za pomocą funkcji g i operator minimum [7, 12] w taki sposób, że:

$$f(x) = \mu_y[g(x, y) = 0].$$

Oznacza to, że wartością funkcji f w punkcie x jest najmniejsza liczba y, dla której funkcja g przyjmuje wartość zero. Gdy taka liczba nie istnieje, to f nie przyjmuje wartości w punkcie x. W systemie funkcji normalnych funkcję f definiuje się poprzez układ dwóch równań:

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

$$h(x, y) = \begin{cases} y & \text{dla } g(x, y) = 0, \\ h(x, \mathcal{S}(y)) & \text{dla } \mathcal{Z}(x) = 0. \end{cases}$$

Zatem wszystkie funkcje obliczalne są funkcjami normalnymi. Z drugiej strony powstaje pytanie, czy każda funkcja normalna jest obliczalna. Z jednej strony dla każdego normalnego układu równań z łatwością można napisać odpowiedni algorytm obliczający użyte funkcje nawet na maszynie Turinga, a z drugiej strony nie jest znana funkcja normalna, która nie jest obliczalna.

Literatura

[1] Brady J. M.: *Informatyka teoretyczna w ujęciu programistycznym.* Warszawa WNT, 1983

[2] Birkhoff G., Bartee T. C.: *Współczesna algebra stosowana.* Warszawa PWN, 1983

[3] Ciesielski K., Pogoda Z.: *Bezmiar matematycznej wyobraźni.* Warszawa, Wiedza Powszechna 1995

[4] Davis M. D., Weyuker E. J.: *Computability, complexity and languages – fundamentals of theoretical computer science.* Orlando, Academic Press 1983

[5] Dziubyński I., Świątkowski T.: *Poradnik matematyczny.* Warszawa, PWN 1982

[6] Empacher A., Sęp Z., Żakowska Z., Żakowski W.: *Mały Słownik Matematyczny.* Warszawa Wiedza Powszechna 1975

[7] Engeler E.: *Introduction to the Theory of Computation.* Nowy Jork, Academic Press 1973

[8] Gårding L.: *Spotkanie z matematyką.* Warszawa, Wydawnictwo Naukowe PWN 1993

[9] Graham R. L., Knuth D. E., Patashnik O.: *Matematyka konkretna.* Warszawa, PWN 1996

[10] Harel D.: *Rzecz o istocie informatyki – algorytmika.* Warszawa, WNT 1992

[11] Kordos M., Skwarczyński M., Zawadowski W.: *Leksykon matematyczny.* Warszawa, Wiedza Powszechna 1993

[12] Kościelski A.: *Teoria obliczeń – wykłady z matematycznych podstaw informatyki.* Wrocław, Wydawnictwo Uniwersytetu Wrocławskiego 1997

[13] Kowal S.: *500 zagadek matematycznych.* Warszawa, Wiedza Powszechna 1975

[14] Kuratowski K.: *Teoria mnogości wraz ze wstępem do opisowej teorii mnogości.* Warszawa, PWN 1978

[15] Lewis H., Papadimitriou H.: *Elements of the theory of computation.* Prentice-Hall International 1998

[16] Penrose R.: *Nowy umysł cesarza – o komputerach, umyśle i prawach fizyki.* Warszawa, Wydawnictwo Naukowe PWN 1996

[17] Pogoda Z., Ciesielski K.: *Diamenty matematyki.* Warszawa, Prószyński 1996

[18] Rasiowa H.: *Wstęp do matematyki współczesnej.* Warszawa, PWN 1975

[19] Sierpiński W.: *Wstęp do teorii liczb.* Bydgoszcz, Państwowe Zakłady Wydawnictw Szkolnych 1965

[20] Słupecki J., Borkowski L.: *Elementy logiki matematycznej i teorii mnogości.* Warszawa, PWN 1984

[21] Trachtenbrot B. A.: *Algorytmy i automatyczne rozwiązywanie zadań.* Warszawa, PWN 1961

[22] Wirth N.: *Algorytmy + struktury danych = programy.* Warszawa, WNT 1999

[23] Zasoby internetu