Studia Semiotyczne, t. XXXII, nr 2 (2018), s. 131–152
ISSN 0137-6608, e-ISSN 2544-073X
DOI: 10.26333/sts.xxxii2.08
Rozprawa

Paweł Stacewicz*

LICZBY NIEOBLICZALNE A GRANICE KODOWANIA W INFORMATYCE[1]

Streszczenie: Opis danych i programów komputerowych za pomocą liczb jest epistemologicznie użyteczny, ponieważ pozwala określać granice różnego typu obliczeń. Dotyczy to w szczególności obliczeń dyskretnych (cyfrowych), opisywalnych za pomocą liczb obliczalnych w sensie Turinga. Matematyczny fakt istnienia liczb rzeczywistych innego typu, tj. nieobliczalnych, wyznacza minimalne ograniczenia technik cyfrowych; z drugiej strony jednak, wskazuje na możliwość teoretycznego opracowania i fizycznej implementacji technik obliczeniowo silniejszych, takich jak obliczenia analogowe-ciągłe. Przedstawione w artykule analizy prowadzą do wniosku, że fizyczne implementacje obliczeń niekonwencjonalnych (niecyfrowych) wymagają występowania w przyrodzie wielkości nieskończonych aktualnie (a nie tylko potencjalnie). Za fizycznym istnieniem takich wielkości przemawiają wprawdzie pewne argumenty fizyki teoretycznej, nie są one jednak ostateczne.

Słowa kluczowe: kodowanie liczbowe, liczby obliczalne, liczby nieobliczalne, maszyna Turinga, obliczenia cyfrowe, obliczenia analogowe, nieskończoność.

* Politechnika Warszawska, Wydział Administracji i Nauk Społecznych. E-mail: p.stacewicz@ans.pw.edu.pl. ORCID: 0000-0003-2500-4086.

[1] Dziękuję dwóm anonimowym recenzentom za cenne uwagi i wskazówki, które pozwoliły istotnie ulepszyć pierwotną wersję tekstu. Dziękuję także profesorom Witoldowi Marciszewskiemu i Andrzejowi Biłatowi za owocną merytorycznie i redakcyjnie dyskusję nad tekstem. Za wszelkie pozostałe w pracy błędy, niejasności i niedociągnięcia ponoszę odpowiedzialność ja sam i z góry przepraszam za nie wszystkich Czytających.

Z przyjętego w niniejszej pracy punktu widzenia obiekty informatyczne, w szczególności zaś programy komputerowe, pośredniczą między matematyczną sferą liczb a fizykalną rzeczywistością. Przykładowo: program odtwarzający dźwięki operuje na liczbowych reprezentacjach fal akustycznych, a jego instrukcje powodują, wskutek odpowiedniej konstrukcji komputera, realne fizyczne drgania cząsteczek powietrza. Co więcej, i ten i każdy inny program można analizować na dwóch poziomach, czyli jako obiekt dwojakiego rodzaju: z jednej strony jako ciąg sprowadzalnych do liczb symboli, z drugiej zaś – jako ściśle określony układ fizycznych stanów maszyny (które po uruchomieniu programu powodują regularne zmiany jej kolejnych stanów)[2].

Z uwagi na wskazane odpowiedniości wiele kwestii dotyczących komputerów można rozstrzygać, odnosząc się do własności liczb – takich liczb, które zgodnie z właściwym danej maszynie modelem obliczeń (np. cyfrowym lub analogowym) odpowiadają danym, tekstom i wynikom działania programów.

W obecnej pracy skupię się na programach dla maszyn cyfrowych. Są one opisywane teoretycznie za pomocą Turingowskiego modelu obliczeń (uniwersalnej maszyny Turinga), a ujmując rzecz „liczbowo", za pomocą liczb obliczalnych w sensie Alana Turinga. Powołując się na pewne własności liczb obliczalnych i nieobliczalnych, w szczególności zaś na fakt, że reprezentacje cyfrowe liczb nieobliczalnych cechuje nieskończoność aktualna, określę teoretyczne powody istnienia obliczeniowych ograniczeń takich programów. Przedyskutuję także możliwość pokonania tych ograniczeń za pomocą technik informatycznych, które dopuszczają (teoretycznie) przetwarzanie sygnałów opisywanych przy użyciu liczb nieobliczalnych w sensie Turinga.

Przedstawiony tekst ma w przeważającej części charakter przeglądowy. Zawiera jednak szereg autorskich interpretacji wyników badań informatycznych i metainformatycznych (np. A. Turinga i G. Chaitina), w szczególności zaś interpretacje dotyczące infinitystycznego charakteru liczb nieobliczalnych i rozważanych w informatyce teoretycznej kodów.

---

[2] Niektórzy filozofowie informatyki mówią wprost – przyjmując nastawienie ontologiczne, a nie epistemologiczne – o dualnej, tj. abstrakcyjno-fizycznej, naturze programów komputerowych (Moor, 1978; Colburn, 2000; zob. też Angius, Turner, 2013).

1. Liczby, obliczenia i kodowanie liczbowe

Najważniejszą i najstarszą zarazem ideą, która zaowocowała powstaniem komputerów, a następnie informatyki, jest idea kodowania liczbowego[3]. Stoi za nią przekonanie, że świat liczb (być może nawet tylko naturalnych) i stosunkowo prostych operacji na nich (jak porównywanie, dodawanie czy dzielenie) jest wystarczająco bogaty, aby można w nim było reprezentować różne aspekty świata rzeczywistego.

We współczesnej informatyce *kodowanie liczbowe*, rozumiane jako zapisywanie danych przetwarzanych przez komputery za pomocą liczb[4], jest czynnością powszechną, a być może teoretycznie niezbędną[5]. Jest ono obecne już na poziomie wstępnej formalizacji niektórych zadań, kiedy to występujące w tych zadaniach obiekty (np. tekstowe, dźwiękowe czy graficzne) opisuje się za pomocą specjalnie dobranych i ujętych w odpowiednie struktury liczb. Dla przykładu: znakom przetwarzanym przez edytory tekstowe przypisuje się ściśle określone liczby (zgodnie np. z kodem ASCII), zaś wyświetlane na monitorach obrazy koduje się często w postaci sekwencji liczb, które określają współrzędne i kolory punktów na rastrowej matrycy. Na najniższym poziomie wewnątrzkomputerowych struktur odpowiednie kody powstają w sposób automatyczny, za sprawą specjalnie zaprojektowanych programów

[3] Jej najstarszym bodaj przejawem była filozofia starożytnych pitagorejczyków, która postulowała sprowadzalność wszelkich fragmentów rzeczywistości do pewnego rodzaju liczb (co streszcza się w krótkim haśle, że „wszystko jest liczbą"). We współczesnym myśleniu filozoficznym, zwłaszcza w kontekście filozofii informatyki, idee pitagorejskie odżywają, co niektórzy określają mianem neopitagoreizmu. Dzieje się tak za sprawą swoistego sprzężenia zwrotnego: idee pitagorejskie przyczyniły się do powstania informatyki, a sukcesy tejże, m. in. na polu symulacji zjawisk fizycznych za pomocą operacji na reprezentowanych komputerowo liczbach, wzmacniają pitagorejską wizję świata. (Krajewski, 2014).

[4] W kontekście informatycznym sformułowanie „zapisywanie danych przetwarzanych przez komputery za pomocą liczb" ma najczęściej sens syntaktyczny, a nie abstrakcyjny. Znaczy to, że chodzi w nim o zapisywanie danych za pomocą symbolicznych (i fizycznych) reprezentacji liczb, np. ciągów zero-jedynkowych. W obecnym tekście będę odwoływał się również do abstrakcyjnych (*stricte* matematycznych) własności liczb i ich zbiorów, takich jak ciągłość zbioru liczb rzeczywistych. W przypadku niewystarczającego kontekstu będę sygnalizował jednak, czy w danym miejscu chodzi o abstrakcyjny czy syntaktyczny wymiar pojęcia liczby (pisząc np. że chodzi o rozwinięcie dziesiętne liczby).

[5] Por. dyskusję internetową na blogu Cafe Aleph, która wynikła przy okazji powstawania tej pracy (Stacewicz, 2018b).

(np. kompilatorów). Najważniejsze jednak, że w ujęciu matematycznym, abstrahującym od fizycznej konstrukcji komputera i fizycznych procesów przetwarzania sygnałów, da się je przedstawić liczbowo, na przykład binarnie.

Trzy ostatnie słowa poprzedniego akapitu wskazują, że termin „kodowanie liczbowe" rozumiem w obecnej pracy szeroko. W szczególności rozumiem go szerzej niż termin „kodowanie cyfrowe", który rezerwuję dla sposobu reprezentowania informacji w komputerach cyfrowych, będących maszynami o stanach dyskretnych, operującymi na sygnałach binarnych. Szeroki termin „kodowanie liczbowe" uznaję za zasadny, ponieważ w ogólnie pojętej informatyce rozważa się szerszą klasę maszyn niż cyfrowe. Do owej szerszej klasy należą układy analogowe, które pozwalają (przynajmniej teoretycznie) operować na sygnałach ciągłych opisywanych przez liczby rzeczywiste[6], a także komputery kwantowe, w przypadku których podstawową jednostkę informacji stanowi q-bit, definiowany matematycznie przy użyciu liczb zespolonych[7].

Z pojęciem kodowania liczbowego wiąże się ściśle kluczowe dla informatyki pojęcie *obliczania*. W kontekście rozwiązywania problemów oznacza ono mechaniczną realizację procesu wyznaczania wartości funkcji, która przyporządkowuje danym wejściowym problemu jego konkretne rozwiązania (rozwiązania dla konkretnych danych)[8]. Jeśli dane są kodowane liczbowo, to argumentami i wartościami tejże funkcji są tego typu liczby (np. naturalne lub rzeczywiste), które dopuszcza właściwy danej maszynie sposób kodowania. Ten zaś jest wyznaczony

[6] Por. prace Shannona (1941) oraz Rubela (1993).

[7] Terminu „kodowanie liczbowe" używam także w innej pracy (Marciszewski, Stacewicz, 2011, s. 75–77). Podobnej konwencji pojęciowej jest bliski S. Krajewski, który nie stosuje wprawdzie terminu „kodowanie liczbowe", ale wyróżnia digitalizację jako jeden tylko z typów kodowania (choć najbardziej powszechny), zasadniczo różny od kodowania danych w układach analogowych przetwarzających sygnały opisywane przez liczby rzeczywiste (Krajewski, 2014).

[8] Historycznie rzecz biorąc, pierwsze dojrzałe rozważania o rozwiązywaniu problemów za pomocą obliczeń, tj. mechanicznych operacji na fizycznych odpowiednikach liczb, zawdzięczamy G.W. Leibnizowi. Dla współczesnej koncepcji obliczania szczególnie ważne są następujące jego pomysły i dokonania: konstrukcja maszyny liczącej (wykonującej cztery podstawowe działania arytmetyczne), wynalazek binarnego systemu arytmetycznego, projekt maszyny operującej na binarnych zapisach liczb, a ponadto koncepcja uniwersalnego języka symbolicznego (*lingua charakteristica*) oraz sprzężonego z nim niezawodnego rachunku (*calculus ratiocinator*). Zob. Trzęsicki (2006).

przez odpowiedni model obliczeń (np. cyfrowy lub analogowy). Dopowiedzmy jeszcze, że informatycznym, a nie czysto matematycznym, zapisem obliczanej funkcji jest albo tekst programu (jeśli dana maszyna akceptuje programy napisane w pewnym języku programowania), albo schemat połączeń między elementarnymi układami maszyny (o ile maszynę programuje się fizycznie, tak jak układy analogowe czy pierwsze komputery cyfrowe).

Ponieważ zdecydowana większość dzisiejszych komputerów realizuje obliczenia cyfrowe, w kolejnych akapitach przyjrzę się bliżej „liczbowej" charakterystyce powierzanych im zadań. W szczególności rozważę pytanie o to, czy pożądane w ich opisie kody liczbowe muszą mieć charakter skończony, czy też niekiedy trzeba odwołać się do pojęcia kodu nieskończonego[9]?

Na pierwszy rzut oka wszelkie wchodzące w grę kody są skończone, tym samym zaś sprowadzalne do liczb naturalnych. Sugeruje to obserwacja, że wprowadzane do komputera cyfrowego dane mają reprezentację skończoną, a stosowane do ich przetwarzania programy są skończonymi sekwencjami instrukcji, które po zakodowaniu w postaci binarnej można interpretować jako liczby naturalne. Głębszy namysł nad funkcjami komputerów cyfrowych prowadzi jednak do stwierdzenia, że teoretyczna analiza możliwości tych komputerów musi odwoływać się do pojęcia kodu n i e s k o ń c z o n e g o (nawet jeśli kodów tego rodzaju nie da się zaimplementować wewnątrz realnych maszyn cyfrowych). Należy przy tym rozróżnić dwa konteksty możliwych odwołań.

[9] Pojęcie kodu nieskończonego – czyli takiego wyniku procesu kodowania, który ma nieskończoną (aktualnie) długość – jest pojęciem niestandardowym, wykraczającym poza standardową teorię obliczalności, wyrażoną np. w terminach maszyn Turinga. Niemniej, we współczesnej metodologii informatyki, która uwzględnia również pewne niestandardowe modele obliczeń, pojęcia tego się używa – mówiąc np. o nieskończonej długości kodach programów czy zapisanej w całości nieskończonej taśmie maszyny Turinga (Ord, 2002, s. 17; Ord, 2016, s. 146; Mycka, Olszewski, 2015, s. 58–59). Podkreślmy jednak, że pojęcie to nabiera sensu wówczas, gdy założy się (nawet roboczo) możliwość wykroczenia poza tradycyjny Turingowski model obliczeń. Użycie pojęcia nieskończonego kodu jest w obecnej pracy uzasadnione, ponieważ w dalszej jej części (zwłaszcza w rozdziale 4.) będę analizował możliwość fizycznej realizacji obliczeń pozaturingowskich, również takich, które obejmują elementy infinitystyczne. Niezależnie od tej intencji, już w tym rozdziale jednak pokażę, jak (ogólna) analiza problemów, które chcielibyśmy rozwiązywać tradycyjnie (tj. cyfrowo), prowadzi do konieczności krytycznego przynajmniej rozważenia kodów nietradycyjnych (tj. nieskończonych).

Po pierwsze, w przypadku wielu realnych problemów (np. z zakresu dynamiki czy mechaniki) wyniki uzyskiwane dla konkretnych danych wejściowych mogą wyrażać się *liczbami niewymiernymi*, a więc takimi, które mają nieskończone i nieregularne rozwinięcia (np. dziesiętne). Dzieje się tak na przykład wtedy, gdy dany problem jest sformułowany matematycznie za pomocą pewnego równania (np. różniczkowego), a pierwiastkiem tego równania jest liczba niewymierna (jak $\sqrt{2}$, π czy e). W takim przypadku poszukiwany wynik jest *de facto* reprezentowany przez liczbę o nieskończonym rozwinięciu. Zauważmy wstępnie, przed dokładniejszymi wyjaśnieniami w rozdziale 3, że najbardziej kłopotliwa sytuacja występuje wtedy, gdy mamy do czynienia z tego rodzaju liczbą niewymierną, która jest przestępna, a dodatkowo nieobliczalna w sensie Turinga.

Po drugie jednak, co dla dalszych analiz jest kluczowe, każde bardziej skomplikowane zadanie programistyczne ma strukturę *infinitystyczną*. Znaczy to, że zbiór jego danych początkowych, a niekiedy także zbiór jego potencjalnych wyników, jest nieograniczony. Jako prosty przykład rozważmy problem wyznaczania pierwiastków równań kwadratowych $ax^2 + bx + c = 0$, gdzie zakres możliwych do wprowadzenia współczynników *a, b, c* jest nieograniczony. W przypadku tego akurat problemu istnieje, mimo nieograniczonej dziedziny, skończona metoda znajdowania szukanych wartości *x*, którą jest powszechnie znany algorytm „delty". Istnieje również skończony program (niejeden), który dla dowolnej danej wejściowej (tj. układu współczynników *a, b, c*) pozwala, w skończonej liczbie kroków, wygenerować poprawny wynik. Ów program trzeba potraktować jako ogólne (komputerowe) rozwiązanie postawionego problemu, któremu to rozwiązaniu odpowiada skończony kod liczbowy programu (mówiąc krótko: pewna liczba)[10].

Niestety, w przypadku innych problemów o nieograniczonej dziedzinie danych wejściowych liczbowy kod ogólnego rozwiązania – bę-

[10] Podkreślmy tutaj, że jakkolwiek kwestia nieskończonej dziedziny danych wejściowych może być nieistotna z punktu widzenia rozwiązania zadania algorytmicznego, to fakt, że rozwiązanie to obowiązuje dla nieograniczonej liczby danych wejściowych, stanowi o jego sile. Jest to rozwiązanie w pewnym sensie uniwersalne (na podobieństwo twierdzeń matematycznych ma ono zastosowanie do nieskończonej liczby przypadków szczególnych). W niektórych jednak sytuacjach nieskończona dziedzina może prowadzić do kłopotów – o czym dalej w głównym tekście (zob. także Stacewicz, 2015).

dący cyfrowym zapisem wszystkich możliwych par <DANE WEJŚCIOWE, WYNIK>, a mówiąc inaczej, funkcji przyporządkowującej danym wyniki – musi pozostać nieskończony. Dzieje się tak, kiedy nie istnieje skończony program rozwiązujący ów problem. Jeśli program taki istnieje, stanowi on „zrozumiałą" dla maszyny cyfrowej formę zakodowania zbioru wspomnianych par w postaci procedury generującej poprawne wyniki (dla wszelkich możliwych danych wejściowych). Kodowi takiej procedury odpowiada przy tym jakaś liczba naturalna (zapisana np. jako ciąg zer i jedynek). Jeśli program taki nie istnieje, trzeba przyjąć, że całościowemu rozwiązaniu problemu odpowiada jakaś liczba nieobliczalna w sensie Turinga (tj. pewna specjalna liczba niewymierna o nieskończonym i niewyznaczalnym za pomocą maszyn cyfrowych rozwinięciu; zob. dalej w rozdziale 2.).

W interesującym nas tu kontekście rozwiązywania problemów za pomocą obliczeń, nieskończone kody liczbowe mogą wystąpić zatem na dwóch poziomach: 1) na poziomie kodu dokładnego jednostkowego wyniku, 2) na poziomie kodu całościowego rozwiązania problemu. W obydwu przypadkach może się zdarzyć, że odpowiedni kod ma postać liczby nieobliczalnej i wówczas – jak zobaczymy w rozdziale 3. – poszukiwana metoda rozwiązania problemu leży poza granicami możliwości kodowania cyfrowego (co nie wyklucza jednak istnienia takiej metody, która byłaby implementowalna na innego rodzaju maszynach niż cyfrowe).

2. Liczby nieobliczalne w sensie Turinga

Uwypuklone w tytule niniejszego artykułu *liczby nieobliczalne* zdefiniował Alan Turing w pracy z roku 1936 pt. *On Computable Numbers, with an Application to the Entscheidungsproblem*. Określił je jako tego rodzaju liczby niewymierne, których przedstawienia dziesiętnego nie może wyznaczyć, z dowolną zadaną dokładnością, żaden układ do obliczeń mechanicznych, zwany dziś maszyną Turinga[11]. We współczesnej stylistyce powiedzielibyśmy, że są to liczby niewyznaczalne za po-

[11] Warto dodać, że Turing podał w pierwszej kolejności ścisłą definicję zbioru liczb obliczalnych (liczb, których zapis dziesiętny można wyznaczyć ostatecznie lub z dowolną zadaną dokładnością za pomocą skończonego programu dla maszyny Turinga), a następnie udowodnił istnienie liczb rzeczywistych innego typu (zob. dalej w głównym tekście), czyli liczb nieobliczalnych (Turing, 1936).

mocą algorytmów dla maszyn cyfrowych, a zatem takie, dla których nie istnieją skończone programy komputerowe pozwalające obliczać krok po kroku kolejne cyfry ich dziesiętnych lub innych reprezentacji (mimo że reprezentacje takie są ściśle określone, zob. Stacewicz, 2012). Przykładowo: niewymierna liczba *e* nie wykazuje powyższych właściwości, ponieważ stosunkowo łatwo jest generować kolejne cyfry jej rozwinięcia za pomocą programu obliczającego kolejne sumy częściowe odpowiedniego szeregu (przypomnijmy, że $e = \sum_{n=0}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots$). Nie jest to zatem liczba nieobliczalna, choć cechuje ją niewymierność.

W odróżnieniu od niewymiernej liczby *e* wielkości nieobliczalne w sensie Turinga są definiowane w sposób wykluczający możliwość ich sukcesywnego przybliżania za pomocą maszyn Turinga lub równoważnych im mechanizmów obliczeniowych.

Przesądza o tym oryginalne rozumowanie Turinga, który zdefiniowawszy liczby obliczalne, udowodnił, że istnieją liczby rzeczywiste innego typu, a następnie określił zbiór liczb nieobliczalnych jako dopełnienie zbioru liczb obliczalnych do zbioru liczb rzeczywistych. Rozumowanie to przedstawię w sposób szkicowy i poglądowy – ograniczając je do liczb rzeczywistych z przedziału (0,1)[12].

Punktem wyjścia wywodu jest stwierdzenie, że wynikowi działania każdej maszyny Turinga dla określonych danych wejściowych – maszyny generującej ciągi cyfr ze zbioru {0,1…,9} – odpowiada jednoznacznie pewna liczba rzeczywista z przedziału (0,1). Jest to taka liczba, której rozwinięcie dziesiętne jest tożsame z generowanym przez maszynę ciągiem cyfr, skończonym bądź nieskończonym.

Ze względu na fakt, że każdą maszynę wraz z danymi wejściowymi określa jednoznacznie pewien unikatowy ciąg symboli (reprezentujących jej program oraz początkową zawartość taśmy), każdej z nich można przypisać unikatowy numer, zaś wszystkie maszyny można ustawić w nieskończony przeliczalny ciąg. Zgodnie z kolejnością w tym ciągu można ustawić następnie wszystkie sekwencje cyfr generowane przez kolejne maszyny. Sekwencje te tworzą nieskończony zbiór przeliczalny i wyznaczają jednoznacznie wszystkie liczby o b l i c z a l n e

[12] W przedstawionym rozumowaniu Turing posłużył się umiejętnie techniką przekątniową, którą po raz pierwszy zastosował G. Cantor w dowodzie nieprzeliczalności zbioru liczb rzeczywistych.

z przedziału (0,1). Są to liczby o rozwinięciach dziesiętnych tożsamych z kolejnymi sekwencjami.

Dysponując określoną wyżej listą sekwencji, można spytać, czy występuje na niej taka sekwencja S, że jej *n*-ta cyfra różni się (np. o 1) od *n*-tej cyfry *n*-tej sekwencji na liście (o ile *n*-ta sekwencja jest wystarczająco długa). Postulowana sekwencja S nie może występować na liście, ponieważ różni się (co najmniej jedną cyfrą) od każdej z sekwencji ustawionych w ciąg. Różni się zatem od dowolnej sekwencji, generowanej przez dowolną maszynę. Sekwencja ta musi zatem określać liczbę z przedziału (0,1), której żadna maszyna nie jest w stanie wygenerować, a więc rzeczywistą liczbę nieobliczalną (Turing, 1936; Marciszewski, Stacewicz, 2011)[13].

Pierwszą ścisłą definicję liczb nieobliczalnych pewnego rodzaju podał współczesny matematyk, Gregory Chaitin. Są to liczby *Omega*, które dla uniwersalnej maszyny Turinga danego typu (tzn. maszyny o określonej liczbie stanów i symboli alfabetu) określają prawdopodobieństwo, że losowo wybrany program działania takiej maszyny zatrzyma się[14]. Wyjaśnijmy dodatkowo, że przez program działania rozumie się tutaj początkową zawartość taśmy maszyny uniwersalnej, na którą składa się odpowiednio zakodowany program maszyny symulowanej oraz jego dane początkowe[15]. Chodzi zatem o dane wejściowe maszyny uniwersalnej, które jednak określają ściśle jej kolejne działania (maszyna uniwersalna realizuje program maszyny konkretnej dla

[13] Zauważmy jeszcze, że zaproponowana wyżej procedura określania sekwencji *S* jest nieefektywna (choć teoretycznie dozwolona), ponieważ ze względu na nierozwiązywalność problemu stopu maszyn Turinga (w cytowanej pracy Turinga znajdziemy odpowiedni dowód) nie wiemy, która z maszyn generujących sekwencje zestawione na liście zatrzymuje się, a która nie (co więcej: w drugim przypadku nie wiemy, czy głowica maszyny nie „nawróci" w którymś cyklu i nie zmieni wspomnianej wyżej *n*-tej cyfry). Do zagadnienia stopu nawiążemy dalej, definiując liczbę nieobliczalną *L*.

[14] W literaturze przedmiotu pisze się często o jednej liczbie Omega (zob. np. Trzęsicki, 2006a, s. 125–126). Jest to jednak mylące, ponieważ dla każdej uniwersalnej maszyny Turinga (maszyn takich jest przeliczalnie nieskończenie wiele) istnieje osobna, mająca inną reprezentację symboliczną, liczba Omega.

[15] Oprócz tak rozumianego programu każda maszyna uniwersalna ma swój unikatowy (definiujący ją) program „wykonawczy", który określa sposób realizacji każdego programu umieszczanego na taśmie (reguluje on m.in. to, w jaki sposób głowica maszyny przemieszcza się między kodem programu maszyny symulowanej i jego danymi wejściowymi).

konkretnych danych). Ponieważ konstrukcja podana przez Chaitina jest dosyć złożona i służy określeniu formuły na wspomniane prawdopodobieństwo (Chaitin, 1993; Chaitin, 2005), zaproponuję tu koncepcyjnie prostszą definicję innej wielkości nieobliczalnej. Zachowam przy tym oryginalny pomysł Chaitina, polegający na odwołaniu się do zagadnienia stopu maszyn Turinga[16].

Punktem wyjścia anonsowanej definicji jest sporządzenie uporządkowanej listy programów dla uniwersalnej maszyny Turinga pewnego typu. Podobnie jak w przypadku konstrukcji Chaitina przez program rozumiem tu zawartość początkową taśmy maszyny uniwersalnej (obejmującą kod programu pewnej maszyny konkretnej i jego dane wejściowe). Ponieważ wspomniana lista jest przeliczalnie nieskończona[17], to znajdujące się na niej programy (z danymi) można ponumerować jako p_1, p_2, p_3 itd. Odnosząc się do tejże listy, można zdefiniować następującą, zapisaną binarnie liczbę L z przedziału (0,1): $L = 0{,}b_1b_2b_3\ldots$, gdzie bit $b_i = 1$, jeśli program p_i zatrzymuje się, zaś $b_i = 0$, jeśli program p_i nie zatrzymuje się (dopowiedzmy, że $i \in N$)[18].

---

[16] Przypomnijmy, że zagadnienie to wyraża się pytaniem o istnienie takiej (diagnostycznej) maszyny Turinga, która dla każdej innej maszyny Turinga i każdych jej danych wejściowych byłaby w stanie jednoznacznie rozstrzygać, czy ta właśnie maszyna, dla tych właśnie danych wejściowych zakończy pracę, czy też będzie pracować w nieskończoność.

[17] Jest ona nieskończona, ponieważ ze względu na nieskończoną długość taśmy maszyny uniwersalnej istnieje nieskończenie wiele danych wejściowych, które można na niej umieścić (mimo skończonej liczby symboli alfabetu i skończonej liczby stanów symulowanych maszyn konkretnych).

[18] Jak zauważa Chaitin, kwestia odpowiedniego doboru listy, tj. sposobu uporządkowania zbioru programów, jest niezwykle ważna. Należy podkreślić, że jest ona ważna nie tylko w przypadku definiowania liczb Omega (w ich przypadku Chaitin podał szczególny sposób określenia listy), ale również w przypadku definiowania innego typu liczb nieobliczalnych (Chaitin, 1993). Jeden z anonimowych recenzentów niniejszej pracy stwierdził słusznie, że typ określonej w głównym tekście liczby L (obliczalna czy nieobliczalna), zależy od sposobu uporządkowania zbioru programów p_i (czyli sposobu sporządzenia ich listy). W szczególności: dla pewnych porządków można uzyskać liczby obliczalne (jak np. 2/3). Aby problem ten rozwiązać, można przyjąć wzmiankowaną wyżej definicję listy Chaitina. Niezależnie od powyższych wyjaśnień trzeba podkreślić jednak, że liczba L jest zdefiniowana w taki sposób, że nawet jeśli występująca w jej definicji lista powoduje jej obliczalność, to sama ta definicja nie pozwala stwierdzić tejże obliczalności za pomocą jakichkolwiek operacji realizowalnych przez maszyny Turinga. Dzieje się tak, ponieważ podstawą definicji jest problem stopu, a jego nierozstrzygalność powoduje, że nie można stwierdzić (zawczasu), które programy na liście zatrzymują

Zauważmy, że liczba L jest ściśle określona – ponieważ programy p_i definiujące jej kolejne bity albo zatrzymują się, albo nie. Nie jest natomiast obliczalna, to znaczy algorytmicznie wyznaczalna – ponieważ określająca wartości kolejnych bitów kwestia stopu nie daje się algorytmicznie rozstrzygnąć w skończonym czasie. Przykład ten ponownie pokazuje, że nieobliczalność w sensie Turinga jest silnie powiązana z nieskończonością. Liczba L ma bowiem nieskończone i niewyznaczalne za pomocą skończonego programu rozwinięcie (niewyznaczalne w tym sensie, że obliczenie niektórych jego cyfr musiałoby trwać nieskończenie długo).

Rozwijając „wątek nieskończoności" w kontekście bardziej ogólnym, trzeba stwierdzić, że wszelkie liczby nieobliczalne w sensie Turinga cechuje nieskończoność aktualna (a nie potencjalna[19]). Każda ich reprezentacja symboliczna (np. dziesiętna) zawiera bowiem nieskończoną liczbę cyfr, która musi być rozumiana jako nieskończona całość, niemożliwa do stopniowego generowania, cyfra po cyfrze, za pomocą jakiegokolwiek skończonego programu (dla maszyny cyfrowej)[20].

Wyjaśnijmy na koniec, że klasa liczb nieobliczalnych w sensie Turinga jest niezwykle obszerna, ponieważ ma moc *continuum*, a więc jest równoliczna ze zbiorem liczb rzeczywistych. W odróżnieniu od niej klasa liczb obliczalnych, a więc takich, które są algorytmicznie wyznaczalne za pomocą maszyn Turinga, ma moc *aleph zero*, a więc jest równoliczna ze zbiorem liczb naturalnych[21]. Owa dysproporcja między

się, a które nie. Mówiąc krótko: być może dla pewnej listy programów liczba L jest obliczalna, ale my, posługując się wyłącznie obliczeniami Turingowskimi, nie jesteśmy w stanie tego określić.

[19] W sprawie rozróżnienia między nieskończonością potencjalną i aktualną zob. Murawski (2014). Na uwagę zasługuje również tekst Witolda Marciszewskiego o nieskończoności (Marciszewski, 2012).

[20] Aktualną nieskończoność liczby nieobliczalnej obrazuje dobrze następująca metafora: gdyby jakiś ponadalgorytmiczny Boski Umysł zechciał podzielić się z nami wiedzą o pewnej liczbie nieobliczalnej X, to musiałby wyjawić ją nam w całości, całości nieskończonej, natomiast nie byłby w stanie dostarczyć zwięzłej algorytmicznej reguły opisującej ją w skończony sposób. Jest to swobodna parafraza uwag Chaitina (Chaitin, 1998, s. 54–55). O różnicy między typami nieskończoności przysługujących zapisom liczb obliczalnych (nieskończoność potencjalna) i zapisom liczb nieobliczalnych (nieskończoność aktualna) wypowiadam się szerzej w innej pracy (Stacewicz, 2018a, s. 180–181).

[21] Wynika to z faktu, że wszystkie maszyny generujące unikatowe ciągi symboli składających się na symboliczne przedstawienia liczb obliczalnych można ponu-

nieskończonościami przysługującymi zbiorom liczb obliczalnych i nieobliczalnych wydaje się zaskakująca: wszystko to, co mogą wygenerować maszyny Turinga, okazuje się być „kroplą w oceanie nieobliczalności".

3. Minimalne ograniczenia realnych kodów cyfrowych

Przez *realne kody cyfrowe* rozumiem tutaj liczbowe kody faktycznych, możliwych do fizycznej implementacji, programów, które to programy reprezentują w skończony sposób funkcje wiążące ze sobą dane wejściowe i wyniki obliczeń. Ze względu na obliczeniową równoważność (wyidealizowanych) komputerów cyfrowych i maszyn Turinga[22] wyniki tychże obliczeń są zawsze cyfrowymi reprezentacjami jakichś liczb obliczalnych w sensie Turinga (ewentualnie ich fragmentów, o ile dana liczba ma nieskończone rozwinięcie).

Z uwagi na wspomnianą równoważność ogólne ograniczenia realnych kodów cyfrowych – ograniczenia, którym muszą podlegać wszelkie programy, dla wszelkich maszyn cyfrowych – daje się wyznaczać w ramach Turingowskiego modelu obliczeń, który ma postać abstrakcyjnej maszyny uniwersalnej, zwanej uniwersalną maszyną Turinga (UMT)[23]. Mówiąc dokładniej: jeśli rozwiązania pewnego problemu nie da się zakodować w postaci programu dla UMT, to nie da się go zapisać również jako procedury wykonywalnej przez pewną maszynę cyfrową[24]. Co nie oznacza – dodajmy to koniecznie – że

merować i ustawić w nieskończony ciąg. Zbiór liczb nieobliczalnych musi mieć natomiast moc *continuum*, ponieważ jest określony jako różnica zbioru $\mathbb{R}$ (o mocy *continuum*) i zbioru liczb obliczalnych (o mocy *aleph zero*).

[22] Mówiąc dokładniej, każdy program pewnej maszyny cyfrowej (niezależnie od technicznych szczegółów jej konstrukcji) można przełożyć na program maszyny Turinga, w szczególności zaś na program maszyny uniwersalnej. Mimo to ze względu na czysto fizyczne ograniczenia realnych maszyn cyfrowych (nie idealnych, lecz realnych), nie wszystkie zadania „wykonalne" dla UMT, są wykonalne dla nich. Temat ten zostanie rozwinięty dalej, w głównym tekście tego rozdziału.

[23] Dopowiedzmy, że uniwersalna maszyna Turinga jest to taka maszyna, która za sprawą specjalnie dobranego, definiującego ją programu jest w stanie symulować działanie każdej konkretnej maszyny Turinga (Harel, 2000, s. 252).

[24] Szeroką gamę problemów nieobliczalnych w modelu Turinga opisuje np. Harel (2000, s. 201–224). O pewnych istotnych problemach metamatematycznych tego typu wspomina również Gödel (1995/2018, s. 13, w niniejszym tomie).

nie da się jej określić w postaci procedury dla maszyny innego typu, np. analogowej.

Z punktu widzenia niniejszych rozważań kluczową rolę odgrywa tutaj zagadnienie wyznaczania liczb nieobliczalnych, a dokładniej ich kolejnych cyfr, składających się na ich symboliczne reprezentacje. Liczby takie mają poprawne definicje, ich kolejne cyfry (np. 0 i 1) są dokładnie określone, a mimo to nie istnieje program dla maszyny Turinga, który pozwalałby w skończonym czasie, z dowolną zadaną dokładnością, liczby takie wyznaczać. A zatem funkcje odpowiadające poszczególnym liczbom nieobliczalnym – funkcje wiążące zadawaną dokładność (np. numer ostatniej żądanej cyfry dziesiętnego rozwinięcia liczby) z odpowiednim fragmentem liczby – określają granice kodowania cyfrowego. Jeśli ogólne rozwiązanie danego problemu jest sprowadzalne do tego rodzaju funkcji, to rozwiązania tego nie da się zakodować cyfrowo. Mówiąc jeszcze inaczej: jeśli dla pewnego problemu *P* każdy liczbowy kod funkcji wiążącej jego dane wejściowe i wyniki jest zapisem pewnej liczby nieobliczalnej, to problem ów leży (wtedy i tylko wtedy) poza granicami możliwości kodowania cyfrowego. W ten sposób, tj. objaśniając kwestię nierozwiązywalności problemów za pomocą pewnego typu maszyn w kategoriach liczbowych, zyskujemy pewien nowy wgląd zarówno w powody, jak i w hipotetyczne możliwości przezwyciężenia Turingowskiej nieobliczalności.

Ograniczenia wyznaczone przez liczby nieobliczalne, a dokładniej przez skojarzone z nimi funkcje generujące ich symboliczne reprezentacje, należy traktować jako ograniczenia minimalne, niezależne od fizycznej charakterystyki maszyn cyfrowych. Stwierdzenie to wynika z faktu, że maszyna UMT jest obliczeniowo równoważna nie fizycznym maszynom cyfrowym, lecz komputerom teoretycznym, o nieskończonych zasobach pamięciowych i dowolnie długim, choć skończonym, czasie działania[25]. Znaczy, to że maszyna UMT jest w stanie „wykonać" więcej zadań niż fizyczne maszyny cyfrowe pewnego typu (np. maszyny o maksymalnej pamięci RAM 8 MB). Stąd wniosek, że ograniczenia realnych fizycznych komputerów i sterujących nimi kodów cyfrowych są tak naprawdę większe niż ograniczenia maszyn

[25] Za potencjalnie nieskończone zasoby pamięciowe oraz potencjalnie nieskończony czas działania odpowiada w modelu UMT nieskończona taśma (Stacewicz, 2018a).

wyidealizowanych, czyli maszyn Turinga. Ograniczenia tych ostatnich stanowią zatem „matematyczne minimum", obejmujące swoim zasięgiem wszelkie komputery cyfrowe.

Powróćmy jednak do własności liczb nieobliczalnych. Przypomnijmy za rozdziałem 2., że wszelkie zapisy takich liczb cechuje nieskończoność aktualna. Zapisy te stanowią bowiem nieskończone całości – to znaczy, nieskończone sekwencje symboli, których nie wyznacza żadna skończona reguła, mająca postać skończonego programu dla maszyny Turinga. Patrząc z takiej perspektywy, za matematyczną „przyczynę" Turingowskiej nieobliczalności problemów trzeba uznać nieskończoność aktualną – nieskończoność przysługującą zapisom liczb, które musiałyby kodować rozwiązania tychże problemów.

Z uwagi na wskazane wcześniej odpowiedniości między konkretnymi liczbami tego typu a problemami cyfrowo nieobliczalnymi (np. określona wcześniej liczba L odpowiada problemowi stopu), a także fakt, że zbiór liczb nieobliczalnych ma moc *continuum*, nasuwa się wniosek, że problemów nieobliczalnych w sensie Turinga jest nieskończenie wiele, a ponadto, że jest ich znacznie więcej niż obliczalnych (których zbiór, podobnie jak zbiór liczb obliczalnych, ma moc *aleph zero*). Jest to wniosek, a nie przypuszczenie, ponieważ każdej liczbie nieobliczalnej odpowiada co najmniej jeden problem nierozwiązywalny, polegający na wyznaczaniu dowolnie długiego fragmentu jej cyfrowego przedstawienia.

Można oczywiście utrzymywać, że nieskończone continuum problemów cyfrowo nieobliczalnych mieści w sobie stosunkowo niewielką liczbę zagadnień praktycznie istotnych. Na przykład, nawet problem stopu – jako dotyczący wszelkich maszyn Turinga, a nie tylko jakiegoś ich wyróżnionego podzbioru – można uznawać za zbyt szeroki, a tym samym za mało znaczący z praktycznego punktu widzenia. Skrajnie praktyczny punkt widzenia wydaje się jednak złudny. Trudno bowiem o pewność, że rozwiązania problemów niemających bezpośredniego przełożenia na zastosowania nie kryją w sobie doniosłych praktycznie konsekwencji (których na danym etapie rozwoju nauki i techniki nie znamy)[26].

[26] Aby uzasadnić przekonanie o praktycznej doniosłości wszelkich problemów nieobliczalnych, można powołać się na nieco karkołomne, ale jednak sugestywne rozumowanie przez analogię. Otóż podobnie jak w zbiorze liczb rzeczywistych nie można pominąć (bez uszczerbku dla ich matematycznej użyteczności) liczb nieobliczalnych (bo ich istnienie zapewnia zbiorowi R własność ciągłości), tak w zbiorze wszelkich problemów nie można pominąć zbioru problemów nieobliczalnych. Wy-

Przed przejściem do kolejnego rozdziału, poświęconego technikom alternatywnym względem obliczeń Turingowskich, warto zwrócić uwagę na jedną jeszcze cechę liczb nieobliczalnych. Otóż w stosunku do zbioru liczb osiągalnych dla maszyn Turinga, czyli obliczalnych, są one wielkościami, które wykraczając poza ten zbiór, pozwalają „rozbudować" go do postaci zbioru liczb rzeczywistych. To zaś nasuwa myśl, że mogą istnieć takie techniki informatyczne, które odwołują się do teorii liczb rzeczywistych (a idąc dalej: do pewnych wyników analizy matematycznej), ponadto zaś, w warstwie implementacyjnej pozwalają, operować na fizycznych odpowiednikach niektórych lub wszystkich liczb rzeczywistych. Możliwości istnienia takich technik przyjrzymy się w rozdziale kolejnym.

4. Czy mogą istnieć efektywnie realizowalne kody niecyfrowe?

Ze względu na właściwości komputerów cyfrowych[27] ogół kodów reprezentujących dane, programy i wyniki działania tych urządzeń podlega pewnym minimalnym ograniczeniom, wyznaczanym w ramach Turingowskiego modelu obliczeń. W gruncie rzeczy ograniczenia te polegają na niemożności „wyjścia" poza zbiór liczb obliczalnych w sensie Turinga.

W związku z powyższym zachodzi pytanie o to, czy istnieją jakiekolwiek maszyny informatyczne, różne od cyfrowych, które byłyby w stanie operować na realnych *kodach nieobliczalnych*, tj. pewnych fizycznych reprezentacjach liczb nieobliczalnych w sensie Turinga. Gdyby maszyny takie faktycznie istniały, mogłyby, po pierwsze, rozwiązywać problemy, których jedyne dostępne rozwiązania ogólne są kodowane za pomocą liczb nieobliczalnych, po drugie zaś, mogłyby generować wyniki będące takimi liczbami (lub reprezentowane za ich pomocą). Moc obliczeniowa tego rodzaju automatów byłaby zatem większa od mocy urządzeń cyfrowych.

Z punktu widzenia czystej teorii maszyny takie istnieją, a ogólne zasady ich działania są określone przez różne modele *hiperobliczeń*

wód ten wymagałby dalszego rozwinięcia, dlatego sygnalizujemy go tylko w przypisie.

[27] Przypomnijmy, że chodzi tutaj o obliczeniową równoważność (wyidealizowanych) komputerów cyfrowych i maszyn Turinga.

– nazywanych tak ze względu na właściwy im potencjał poszerzania możliwości maszyny UMT (Copeland, 2002). Należą do nich między innymi: modele infinitystyczne – dopuszczające wykonywanie nieskończonej liczby operacji (obliczeń) w skończonym czasie (Shagrir, 2004); modele niedeterministyczne – opisujące obliczenia inicjowane i/lub kontrolowane losowo (Deutsch, 1985); oraz *analogowe* – pozwalające przetwarzać sygnały ciągłe, opisywane matematycznie za pomocą liczb rzeczywistych z określonego przedziału (Mycka, Piekarz, 2004). Warto podkreślić, że idea kodowania niecyfrowego przejawia się najpełniej w przypadku obliczeń ostatniego typu, tj. analogowych, ponieważ ich teoria daje możliwość operowania na wielkościach (kodach) z całego *continuum* (a nie na konkretnych liczbach nieobliczalnych czy pewnym ich skończonym lub przeliczalnym podzbiorze)[28].

Teoretyczne propozycje obliczeń takiego czy innego typu nie przesądzają oczywiście kwestii ich fizycznej realizowalności. Kwestię tę rozstrzyga negatywnie *hipoteza Churcha-Turinga*, która w jednej z wersji stwierdza, że „funkcja jest efektywnie obliczalna wtedy i tylko wtedy, gdy jest obliczalna za pomocą uniwersalnej maszyny Turinga" (Harel, 2000, s. 240)[29]. W kontekście kodowania sformułowanie to można interpretować tak, że jedynymi efektywnie przetwarzalnymi kodami są dane akceptowalne i możliwe do wygenerowania przez maszynę UMT, a więc kody cyfrowe (dyskretne). Z tej perspektywy zatem wszelkie kody, niezależnie od ich opisu teoretycznego, są praktycznie redukowalne do kodów cyfrowych – co polega między innymi na tym, że zawsze istnieje możliwość ich dowolnie dokładnego przybliżenia za pomocą odpowiedników cyfrowych. Zważywszy na fakt, że model UMT ma charakter teoretyczny i określa bardziej ograniczenia obliczeń niż ich realne możliwości, konkluzję przywołanej hipotezy można ująć jeszcze inaczej. Otóż model UMT wyznacza absolutnie minimalne ograniczenia kodowania w informatyce[30]. Innymi słowy: wszel-

[28] Warto dodać również, że techniki analogowe pozostają najbliższe informatycznej praktyce – tak ze względów historycznych (bo maszyny analogowe konstruowano już w latach 30. XX wieku), jak i z perspektywy współczesnych badań (Mycka, Piekarz, 2004; Shannon, 1941).

[29] Przytoczone sformułowanie traktuję jako hipotezę, ponieważ nie przesądzam tego, czy efektywnie realizowalne fizycznie są wyłącznie obliczenia turingowskie (realizowane w praktyce przez maszyny cyfrowe).

[30] W rozdziale poprzednim, w czwartym akapicie, wyjaśniłem ponadto, że są to minimalne ograniczenia teoretyczne technik cyfrowych.

kie realne obliczenia – niezależnie od teoretycznego modelu, który je opisuje – muszą podlegać ograniczeniom określonym w tym właśnie, maksymalnie bliskim praktyce modelu (tj. UMT). Ograniczenia konstrukcji alternatywnych, np. modeli obliczeń analogowych, są po prostu szersze.

Najpoważniejsze argumenty za prawdziwością tezy Churcha-Turinga, a więc także za istnieniem powyższych ograniczeń, odnoszą do pojęcia nieskończoności. Kwestia podstawowa to fakt, że liczby nieobliczalne – odpowiadające rozwiązaniom pewnych problemów – cechuje nieskończoność aktualna. Przypomnijmy, że chodzi tutaj o ich niekończące się, nieregularne i niemożliwe do stopniowego generowania rozwinięcia, które jako nieskończone całości reprezentują (cyfrowo) daną liczbę.

Wyznaczanie takich reprezentacji, a zatem i rozwiązywanie odpowiadających im problemów, musi wymagać posługiwania się istniejącymi w przyrodzie, fizycznymi wielkościami nieobliczalnymi. Osadzenie takich naturalnych nośników nieobliczalności w maszynie jest konieczne, ponieważ wiadomo, że całościowych reprezentacji liczb nieobliczalnych nie daje się kodować ani wyznaczać w sposób tradycyjny, tj. za pomocą minimalnie „angażujących" naturę kodów i operacji binarnych[31]. W szczególności wszelkie efektywne implementacje wspomnianych wyżej technik analogowych wymagają wykorzystywania fizycznych wielkości nieobliczalnych. Wynika to z faktu, że zarówno specyfika, jak i siła tych technik (to znaczy: ich większa moc obliczeniowa od technik cyfrowych) polegają na możliwości przetwarzania i generowania wielkości z pewnego ciągłego *continuum* (Mycka, Piekarz, 2004). To zaś nie byłoby ciągłe, gdyby nie wypełniające je wielkości nieobliczalne[32].

[31] Kody i operacje binarne również muszą być fizycznie zaimplementowane za pomocą takich czy innych wielkości naturalnych (np. impulsów elektrycznych); rzecz jednak w tym, że w ich przypadku wystarczy posługiwać się dowolnymi właściwie wielkościami fizycznymi, które są łatwo rozróżnialne (albo nawet jedną rozpoznawalną wielkością i jej brakiem). Stopień „zaangażowania" natury jest więc w ich przypadku minimalny.

[32] Ten sam fakt można wyrazić, odnosząc się do własności liczb rzeczywistych, które stanowią matematyczny odpowiednik przetwarzanych analogowo sygnałów ciągłych. Otóż bez liczb nieobliczalnych każdy przedział liczb rzeczywistych (odpowiednik fizycznej dziedziny sygnałów analogowych) ma moc *alef zero*, a więc jest równoliczny z dyskretnym zbiorem liczb naturalnych.

Powstaje zatem realny problem istnienia nośników nieobliczalności w przyrodzie. Przypomnijmy, że ich najbardziej problematyczna cecha to mająca im przysługiwać fizycznie, zgodnie jednak z teoretycznymi własnościami liczb nieobliczalnych, nieskończoność aktualna[33]. Gdyby nośniki takie istniały, zbiór możliwych do praktycznego wykorzystania informatycznych kodów wykraczałby poza zbiór kodów cyfrowych. Obejmowałby on takie kody, które mają bezpośrednie umocowanie w naturze. Ich niektóre przynajmniej składniki byłyby po prostu „wywołaniami" zjawisk naturalnych, które zwracałyby wprost i całościowo pewne wielkości nieobliczalne. W szczególności zainicjowana przez Claude'a Shannona teoria obliczeń analogowych-ciągłych (Shannon, 1941) przewiduje, że w skład złożonego kodu analogowego mogą wchodzić elementarne operacje całkowania, których ciągłe wyniki (realizowane w czasie rzeczywistym) muszą być uzyskiwane w drodze pomiaru zjawisk zachodzących w specjalnych układach fizycznych (np. elektronicznych integratorach). A jak już pisałem wyżej, dla ciągłości zbioru wyników jest niezbędne, aby zawierały się w nim wielkości nieobliczalne.

Istnienie zjawisk przyrodniczych o charakterze *nieobliczalnym* – to znaczy takich, które nie dają się opisać w kategoriach liczb obliczalnych i funkcji realizowalnych przez maszyny Turinga – postulują pewne teorie fizyczne. Jeden ze szczególnie często cytowanych przykładów pochodzi z pracy Pour-El i Richardsa (1989). Zgodnie z nim opisana pewnym równaniem różniczkowym fala trójwymiarowa może uzyskiwać stany wyrażalne tylko za pomocą liczb nieobliczalnych. Do tej samej kategorii należą propozycje Johna Doyle'a, które wskazują na niemożność opisu występujących w przyrodzie procesów osiągania równowagi (np. termodynamicznej) za pomocą funkcji obliczalnych (Copeland, 2002, s. 470). Te i inne przykłady zdają się wskazywać na realne istnienie zjawisk, które moglibyśmy traktować jako naturalne nośniki nieobliczalności. Pamiętajmy jednak, że za zgodność teorii fizycznych z rzeczywistością odpowiadają testy empiryczne, których żadna skończona liczba (znowu problem z nieskończonością!) nigdy ze stuprocentową pewnością nie pozwoli stwierdzić.

[33] Doniosłą filozoficznie argumentację za istnieniem wielkości aktualnie nieskończonych w przyrodzie zawiera opracowanie *Amor Infiniti. Jakie doń prowadza intuicje filozoficzne?* (Marciszewski, 2012).

Załóżmy jednak, niezależnie od powyższej obiekcji natury epistemologicznej, że fizyczne nośniki kodów nieobliczalnych istnieją i można ich używać w ramach takich czy innych obliczeń naturalnych[34]. Mimo takiego założenia pojawia się kolejny problem, dotyczący możliwości *odczytu*, a więc poznania uzyskanego wyniku. Problem polega na tym, że dla poznania wyniku konieczna jest nieskończona dokładność odczytu całej wielkości nieobliczalnej[35]. Jest ona niezbędna, ponieważ dokładność skończona, która charakteryzuje przecież wszelkie realne instrumenty pomiarowe, sprowadziłaby pożądaną w danej sytuacji liczbę nieobliczalną do poziomu skończonej wielkości obliczalnej. Utracilibyśmy zatem oczekiwany efekt przezwyciężenia ograniczeń obliczeń cyfrowych. Można wprawdzie argumentować, że w przypadku niektórych problemów wystarczy, aby wielkości nieobliczalne były po prostu przetwarzane, a nie odczytywane – bo rozwiązaniem problemu jest jakaś konkretna skończona wartość, którą można odczytać (Stannett, 2003, s. 121–123). W rozważanym tu ujęciu chodzi jednak o znajomość ogólnego rozwiązania problemu (funkcji wiążącej wszelkie możliwe dane wejściowe z odpowiadającymi im wynikami), a tego typu rozwiązanie koduje cała liczba nieobliczalna o nieskończonym aktualnie rozwinięciu. Problem epistemiczny zatem pozostaje: bez nieskończonej dokładności odczytu rozwiązania takiego nie możemy poznać.

Konkludując: przysługująca liczbom nieobliczalnym nieskończoność aktualna sprawia, że sugerowane przez tezę Churcha-Turinga ograniczenia technik obliczeniowych – technik, które wymagają fizycznej implementacji określonych informatycznych kodów – można przezwyciężyć pod dwoma co najmniej warunkami: 1) występowanie w przyrodzie wielkości nieskończonych aktualnie, które ponadto możemy rejestrować i przetwarzać, 2) istnienie umysłowej dys-

[34] Mam tutaj na myśli obliczenia projektowane przez człowieka, lecz angażujące w istotny sposób substraty i/lub procesy naturalne (np. obliczenia kwantowe lub dokonywane za pomocą molekuł DNA). Do klasy obliczeń naturalnych w szerszym sensie zalicza się ponadto: 1) obliczenia inspirowane obserwacją natury (np. realizowane przez sztuczne sieci neuronowe) oraz 2) procesy występujące w przyrodzie, opisywane w kategoriach obliczeniowych (np. procesy wewnątrz-mózgowe) (Kari, Rozenberg, 2008; Rozenberg, Back, Kok, 2012).

[35] Dokładność taka jest konieczna w przypadku technik analogowych, które z definicji operują na wielkościach ciągłych (dwie wielkości z dziedziny ciągłej mogą różnić się od siebie dowolnie mało).

pozycji do wglądu w obiekty nieskończone aktualnie oraz dotyczące ich relacje i metody (np. metody definiowania). Warunek drugi trzeba uznać za spełniony – o czym świadczą tworzone przez ludzi teorie nieskończoności aktualnej, w tym teoretyczne modele obliczeń na wielkościach nieskończonych aktualnie. Możliwość spełnienia warunku pierwszego wydaje się co najmniej problematyczna.

BIBLIOGRAFIA

Angius, N., Turner, R. (2017). Philosophy of Computer Science. *Stanford Encyclopedia of Philosophy*, pobrane z: https://plato.stanford.edu/entries/computer-science/

Chaitin, G. J. (1993). Randomness in Arithmetic and the Decline and Fall of Reductionism in Pure Mathematics. *Bulletin of the European Association for Theoretical Computer Science*, *50*, 314–328.

Chaitin, G. J. (1998). *The Limits of Mathematics*. Singapore: Springer.

Chaitin, G. J. (2005). Omega and Why Maths Has No TOEs. Pobrane z: https://plus.maths.org/content/os/issue37/features/omega/

Colburn, T. R. (2000). *Philosophy and Computer Science*. Armonk, NY: M.E. Sharpe.

Copeland, J. (2002). Hypercomputation. *Mind and Machines*, *12*(4), 461–502.

Deutsch, D. (1985). Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. *Proceedings of The Royal Society of London A*, *400*, 97–117.

Etesi, G., Nemeti, I. (2002). Non-Turing Computations via Malament-Hogarth Space-Times. *International Journal of Theoretic Physics*, *41*(2), 341–370.

Gödel, K. (1995/2018). O pewnych zasadniczych twierdzeniach dotyczących podstaw matematyki i wnioskach z nich płynących. *Studia Semiotyczne*, *32*(2), 9–32.

Harel, D. (2000). *Rzecz o istocie informatyki. Algorytmika*. Warszawa: Wydawnictwa Naukowo-Techniczne.

Kari, L., Rozenberg, G. (2008). The Many Facets of Natural Computing, *Communications of the ACM*, *51*(10), 72–83.

Krajewski, S. (2014). Neopitagoreizm współczesny: uwagi o żywotności pitagoreizmu. W: M. Heller, S. Krajewski (red.), *Czy fizyka i matematyka to nauki humanistyczne?* (s. 348–366). Kraków: Copernicus Center Press.

Leibniz, G. W. (1890). *Philosophische Schriften* (t. VII). Berlin: Weidmann.

Marciszewski, W. (2012). Amor Infiniti. Jakie doń prowadzą intuicje filozoficzne? Pobrane z: http://marciszewski.eu/?p=2955

Marciszewski, W., Stacewicz, P. (2011). *Umysł-Komputer-Świat. O zagadce umysłu z informatycznego punktu widzenia*. Warszawa: Akademicka Oficyna Wydawnicza EXIT.

Moor, J. H. (1978). Three Myths of Computer Science, *The British Journal for the Philosophy of Science*, *29*(3), 213–222.

Murawski, R. (2014). Nieskończoność w matematyce. Zmagania z potrzebnym, acz kłopotliwym pojęciem. *Zagadnienia Filozoficzne w Nauce*, *55*(2), 5–42.

Mycka, J. M., Piekarz, M. (2004). Przegląd zagadnień obliczalności analogowej. W: S. Grzegórski, M. Miłosz, P. Muryjas (red.), *Algorytmy, metody i programy naukowe* (s. 125–132). Lublin: Polskie Towarzystwo Informatyczne.

Mycka, J. M., Olszewski A. (2015). Czy teza Churcha ma jeszcze jakieś znaczenie dla informatyki? W: P. Stacewicz (red.), *Informatyka a filozofia. Od informatyki i jej zastosowań do światopoglądu informatycznego* (s. 53–74). Warszawa: Oficyna Wydawnicza Politechniki Warszawskiej.

Ord, T. (2002). Hypercomputation: Computing More Than the Turing Machine. Pobrane z: https://arxiv.org/ftp/math/papers/0209/0209332.pdf

Ord, T. (2006). The many forms of hypercomputation. *Applied Mathematics and Computation*, *178*(1), 8–24.

Pour-El, M. B., Richards, J. I. (1989). *Computability in Analysis and Physics*. Berlin: Springer.

Rozenberg, G., Back, T., Kok, J. N. (2012). *Handbook of Natural Computing*. Berlin-Heidelberg: Springer.

Rubel, L. (1993). The Extended Analog Computer. *Advances in Applied Mathematics*, *14*(1), 39–50.

Shagrir, O. (2004). Super-Tasks, Accelerating Turing Machines and Uncomputability. *Theoretical Computer Science*, *317*(1-3), 105–114.

Shannon, C. (1941). Mathematical Theory of the Differential Analyzer. *Journal of Mathematics and Physics*. *20*(1-4), 337–354.

Stacewicz, P. (2012). Co łączy umysł z teorią liczb? *Filozofia Nauki*, *79*(3), 111–126.

Stacewicz. P. (2015). Informatyczne kłopoty z nieskończonością. W: R. Murawski (red.), *Filozofia matematyki i informatyki* (s. 310–327). Kraków: Copernicus Center Press.

Stacewicz, P. (2018a). Czy informatykom musi wystarczyć nieskończoność potencjalna? W: R. Murawski, J. Woleński (red.), *Problemy filozofii matematyki i informatyki* (s. 177–190). Poznań: Wydawnictwo Naukowe Uniwersytetu im. Adama Mickiewicza w Poznaniu.

Stacewicz, P. (2018b). O teoretycznej (nie)zbędności kategorii liczby w informatyce i jej metodologii. Pobrane z: http://marciszewski.eu/?p=9995

Stannett, M. (2003). Computation and Hypercomputation. *Minds and Machines*, *13*(1), 115–153.

Trzęsicki, K. (2006). From the Idea of Decidability to the Number Omega. *Studies in Logic, Grammar and Rhetoric*, *22*(1), 73–142.

Trzęsicki, K. (2006). Leibnizjańskie inspiracje informatyki. *Filozofia Nauki*, *55*(3), 21–48.

Turing, A. M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, *s2-42*(1), 230–265.

UNCOMPUTABLE NUMBERS AND THE LIMITS OF CODING IN COMPUTER SCIENCE

SUMMARY: The description of data and computer programs with the use of numbers is epistemologically valuable, because it allows the definition of the limits of different types of computations. This applies in particular to discrete (digital) computations, which can be described by means of computable numbers in the Turing sense. The mathematical fact that there are other types of real numbers, i.e. uncomputable numbers, determines the minimal limitations of digital techniques; on the other hand, however, it points to the possibility of theoretical development and physical implementation of computationally stronger techniques, such as analogue-continuous computations. Analyses presented in this article lead to the conclusion that physical implementations of unconventional (non-digital) computations require the occurrence of actually infinite entities in nature. Although some arguments of theoretical physics support the physical existence of such entities, they are not definitive.

Key words: numerical coding, computable numbers, uncomputable numbers, Turing machine, digital computations, analogue computations, infinity.