STUDIA METODOLOGICZNE
NR 36 • 2016, 315–331
DOI: 10.14746/sm.2016.36.14

PAWEŁ STACEWICZ

O algorytmach i algorytmicznej dostępności wiedzy

ABSTRACT. On Algorithms and Algorithmic Accessibility of Knowledge
The paper presents a methodological (connected with methods of knowledge acquisition) interpretation of such computer science concepts like: algorithm, time complexity of algorithms, computability and uncomputability. Two concepts of algorithm are distinguished: a) general – expressed precisely by means of universal Turing machine formalism, and b) narrow – described within different models of hypercomputations. The issue of knowledge accessibility (i.e. solvability of specified classes of problems) is discussed with regard to both concepts.

KEY WORDS: algorithm, knowledge, Turing machine, computability, hypercomputations

Czy wszelkie dobrze określone problemy naukowe mogą zostać rozwiązane za pomocą algorytmów? Jakich algorytmów? Czy istnieje jedno pojęcie algorytmu, czy też jest ich więcej i zależnie od typu pojęcia inaczej przedstawia się ocena możliwości metody algorytmicznej?

Oto pytania, których tropem będę podążał w niniejszym tekście. Głównym celem artykułu jest omówienie zagadnienia niepełności metody algorytmicznej, z uwzględnieniem dwóch pojęć algorytmu: wąskiego i szerokiego. Realizując ten cel, będę odwoływał się do takich dyscyplin jak informatyka, metodologia nauk i filozofia. Zdecydowanie najwięcej miejsca poświęcę informatyce – jako tej dziedzinie, w której operuje się jasnym pojęciem algorytmu i bada się jego formalne własności. Z analizy tychże badań wyłonią się dwie konkluzje. Pierwsza, dotycząca dostępu do wiedzy za pośrednictwem algorytmów cyfrowych, będzie jednoznaczna

i negatywna. Druga, osadzona w teoretycznych poszukiwaniach różnych modeli hiperobliczeń, będzie nie tyle negatywna, co sceptyczna. Jej sceptycyzm zaś będzie uzasadniany faktem naszej aktualnej niewiedzy co do własności świata, które warunkują praktyczną realizowalność hiperobliczeń.

Redagując tekst, przyjąłem quasi-encyklopedyczną konwencję podziału prezentowanych treści na krótkie, zwarte, kolejno numerowane punkty; mam nadzieję, że taka metoda przyczyni się do dobrego zrozumienia zaproponowanego wywodu.

I. Informatyczne pojęcie algorytmu

1. Mimo szerokiej obecności we wszystkich niemal naukach ścisłych pojęcie **algorytmu** przynależy przede wszystkim do **informatyki**, a oznacza schemat maszynowej realizacji zadań określonego typu, możliwy do zakodowania w postaci **programu** komputerowego. Pojęcie to należy rozpatrywać w ramach **triady** (*informacja*, *algorytm*, *automat*) – która mieści w sobie trzy kluczowe pojęcia informatyki, a wyraża skrótowo fakt, że dyscyplina ta zajmuje się *algorytmicznym* przetwarzaniem odpowiednio zakodowanych *informacji* (danych) za pomocą określonego rodzaju *automatów* (komputerów)[1].

Warto zauważyć, że centralny element wyżej wymienionej triady, tj. algorytm, wiąże informatykę z **matematyką** – po pierwsze z tego względu, że duża część algorytmów służy do rozwiązywania problemów matematycznych (np. równań i ich układów), po drugie zaś dlatego, że różne własności algorytmów (np. złożoność czasowa) są badane za pomocą narzędzi matematycznych (np. miarą złożoności czasowej są funkcje jednej zmiennej).

[1] Natomiast element trzeci, tj. *automat*, wiąże informatykę z elektroniką – to zaś dlatego, że dopiero fizyczne automaty do przetwarzania danych (współcześnie są to automaty elektroniczne) umożliwiają realne zastosowania informatyki.

2. Informatyczne pojęcie algorytmu ma **rodowód matematyczno-logiczny**, a jego historia łączy się ściśle z długotrwałym procesem coraz szerszego formalizowania i mechanizowania matematycznych rozumowań.

Historię tę można podzielić na trzy nierównomiernej długości **etapy** [por. Pawlak, 1970 i Harel, 2000].

ETAP I. *Od starożytności po początek wieku XX*

Algorytmy są używane spontanicznie (bez ogólnej definicji algorytmu) do rozwiązywania różnorodnych problemów matematycznych: geometrycznych, arytmetycznych i algebraicznych (typowy problem to rozwiązywanie równań i ich układów). Sama nazwa „algorytm" pojawiła się w wieku IX, w wyniku fonetycznej przeróbki imienia uczonego arabskiego (al-Chwarizmiego), który w jednym ze swoich dzieł podał mechaniczne metody działań na liczbach zapisywanych dziesiętnie[2].

ETAP II. *Od roku 1936 do lat 50. XX wieku*

W wyniku prac takich matematyków jak Alan Turing, Alonzo Church i Emil Post pojęcie algorytmu – rozumianego jeszcze jako mechaniczna procedura rozwiązywania zagadnień matematyczno-logicznych – zostaje sprecyzowane, a następnie wykorzystane w badaniach nad rozstrzygalnością systemów formalnych. Największą bodaj rolę odegrała tu matematyczna konstrukcja Turinga, nazywana dziś uniwersalną maszyną Turinga. W tym samym czasie zostały wykryte istotne ograniczenia metody algorytmicznej (np. problem stopu).

ETAP III. *Czasy najnowsze, od drugiej połowy XX wieku*

Pojęcie algorytmu przenika do informatyki, stając się jej pojęciem naczelnym. Opracowuje się coraz to nowe algorytmy dla coraz bardziej zaawansowanych sprzętowo i programistycznie systemów; stosuje się je głównie poza matematyką, do problemów kodowanych liczbowo, ale ma-

[2] Na zasadzie fonetycznego podobieństwa „metody al-Chwarizmiego" zyskały miano „metod algorytmicznych" [por. Kordos 1994].

jących sens nieliczbowy (np. do problemów sterowania, przeszukiwania baz danych, modelowania czynności umysłowych etc). Bada się różne typy i własności algorytmów – mając na uwadze przede wszystkim efektywność odpowiednich programów komputerowych.

3. Wnikając głębiej w specyfikę etapu III, trzeba powiedzieć, że współczesna praktyka tworzenia algorytmów (czy też programów komputerowych) coraz częściej **wykracza** poza klasyczne podejście Turinga, dostosowane do zasad działania komputerów cyfrowych[3]. Powstają coraz to nowe sieci neuropodobne, wracają do łask używane niegdyś bardzo intensywnie techniki analogowe, na szeroką skalę są stosowane maszyny interaktywne i właściwe im algorytmy uczenia się. Ze względu na tenże proces, to jest coraz bardziej intensywne poszukiwania sprzętowo-programistycznej alternatywy dla maszyn cyfrowych, we współczesnej informatyce współistnieją ze sobą **dwa pojęcia algorytmu**.

3a. W ujęciu **wąskim**, a jednocześnie najbardziej precyzyjnym, algorytmem nazywa się każdy ogólny schemat procedury możliwej do wykonania przez **uniwersalną maszynę Turinga** (UMT)[4]. Z uwagi na obliczeniową równoważność UMT i komputerów cyfrowych[5] jest to pojęcie

---

[3] Warto dodać, że jest to praktyka spontaniczna. Nowe algorytmy i programy powstają w odpowiedzi na realne potrzeby. W wielu przypadkach (dotyczy to np. algorytmów uczących się oraz interakcyjnych) nie wiadomo, na ile dany schemat pozostaje zgodny z turingowskim rozumieniem algorytmu (zob. 3a).

[4] Uniwersalna maszyna Turinga jest określona przez taki uniwersalny program, który pozwala symulować działanie każdej konkretnej maszyny Turinga. Na wejście tegoż programu wprowadza się: a) kod maszyny symulowanej, oraz b) dane wejściowe maszyny symulowanej; na wyjściu zaś uzyskuje się taki sam wynik, jaki wygenerowałaby dla danych b) maszyna symulowana [zob. Turing, 1936].

[5] Ściśle rzecz biorąc, komputery cyfrowe nie są obliczeniowo równoważne uniwersalnej maszynie Turinga (UMT), ponieważ w odróżnieniu od UMT, której element stanowi nieskończona taśma, dysponują one skończoną pamięcią. Są one zatem obliczeniowo słabsze od UMT (dodajmy jednak, że „w granicy", przy założeniu ciągłego wzrostu zasobów pamięciowych maszyn, równoważność zachodzi).

Dla przedstawionego dalej wywodu istotne jest, że ograniczenia obliczeń zdefiniowanych w modelu UMT (silnym) muszą stosować się także do (słabszych obliczeniowo) realnych komputerów cyfrowych.

algorytmu dla maszyn cyfrowych (najpowszechniej dziś wykorzystywanych).

3b. W ujęciu **szerszym**, a precyzowanym cząstkowo w ramach różnych pozaturingowskich modeli obliczeń, algorytmem nazywa się ogólny schemat procedury możliwej do wykonania przez **pewną maszynę**, niekoniecznie cyfrową i niekoniecznie deterministyczną (np. analogową, kwantową, ewolucyjną – lista nie jest zamknięta).

Pytaniem otwartym pozostaje, w jakim sensie drugie (ogólniejsze) pojęcie algorytmu jest **sprowadzalne** do pierwszego (węższego). Jest to w istocie pytanie o zasadność tezy Churcha-Turringa (TC), która głosi, że formalizm UMT (a także inne, jemu równoważne) wystarczająco dobrze opisuje pojęcie **efektywnej obliczalności**, a więc algorytmicznej rozwiązywalności problemów [por. Mycka, Olszewski, 2015].

Innymi słowy: jeśli ktoś uznaje tezę TC za zasadną, to musi zgodzić się z tym, że każdy algorytm w szerokim sensie jest w istocie opisem procedury, którą w dostatecznie dobrym przybliżeniu można przedstawić jako program dla uniwersalnej **maszyny Turinga**. Jeśli chodzi natomiast o takie techniki obliczeniowe, które pozwalają rozwiązywać problemy nierozwiązywalne za pomocą maszyn Turinga (jak słynny problem stopu), to zwolennik tezy TC musi uznać te techniki za czysto teoretyczne, a tym samym **nierealizowalne w praktyce**. Musi przyjąć nadto, że pewne nierozwiązywalne cyfrowo przypadki szczególne wspomnianych problemów (nawet gdyby było ich nieskończenie wiele) nie są istotne z praktycznego punktu widzenia.

4. Niezależnie od ogólnego rozróżnienia między algorytmami rozumianymi wąsko i szeroko, z informatycznej praktyki wyłania się bardzo bogata **typologia algorytmów** (zarówno tych faktycznie stosowanych, jak i postulowanych z myślą o maszynach przyszłości).

Pewien jej fragment obrazują następujące pary[6] **przeciwieństw**:

[6] Są one przykładowe i wyróżnione ze względu na różne kryteria.

a) algorytmy cyfrowe vs analogowe,
b) algorytmy deterministyczne vs niedeterministyczne,
c) algorytmy szeregowe vs równoległe,
d) algorytmy sekwencyjne vs rekurencyjne,
e) algorytmy linearne vs populacyjne,
f) algorytmy działania vs algorytmy uczenia się[7].

Chociaż w większości przypadków w/w algorytmy są realizowane na maszynach cyfrowych (produkowanych masowo, a przez to najszerzej dostępnych), to w sferze czystej teorii niektóre z nich sytuują się poza (turingowskim) modelem obliczeń cyfrowych. Dotyczy to z pewnością obliczeń/algorytmów **analogowych**, które przy pewnym rozumieniu analogowości pozwalają przetwarzać dane istotnie różne od reprezentowanych cyfrowo dyskretnych symboli, a mianowicie dane ciągłe [por. Mycka, Piekarz, 2004]. Do tematu analogowości wrócę pod koniec tekstu.

5. Ponieważ siłą napędową informatyki są zastosowania, to będące ich podstawą algorytmy bada się przede wszystkim ze względu na **skuteczność.** Spośród warunkujących tę skuteczność własności na szczególną uwagę zasługują: zakres stosowalności algorytmu, własność stopu, stabilność numeryczna oraz różne rodzaje złożoności (odnoszące się zarówno do szybkości wykonywania algorytmów, jak i rozmiaru wykorzystywanej pamięci komputera; [por. Wirth 1989]).

Ze względu na tytułowe zagadnienie artykułu, tj. algorytmiczny dostęp do wiedzy, w dalszej części tekstu skupimy się na dwóch własnościach:

i) **własności stopu** – przysługującej algorytmowi wtedy, gdy dla wszelkich swoich danych wejściowych przewidziane przezeń sekwencje operacji kończą się (mówiąc nieco żargonowo: nie zapętlają się w nieskończoność),

---

[7] Z każdą ze wskazanych opozycji wiąże się szereg ważnych i ciekawych zagadnień filozoficznych.

Na przykład: a) pod jakimi warunkami co do struktury świata/materii techniki analogowe są realizowalne w praktyce (z punktu widzenia teorii są to techniki istotnie różne od technik cyfrowych)?; albo b) w jakim sensie i pod jakimi warunkami algorytmy uczenia się mogą skutkować inwencją maszyn? [Por. Stacewicz, 2010].

ii) **złożoności czasowej** – określającej szybkość realizacji algorytmu zależnie od rozmiaru danych wejściowych.

Własność pierwsza ma znaczenie zupełnie podstawowe, ponieważ od każdego efektywnego programu komputerowego oczekuje się przede wszystkim tego, by rozwiązując jakiś problem, zakończył pracę. Niestety, jak się okaże dalej, istnieją procedury, które dla pewnych typów danych wejściowych mogą trwać w **nieskończoność**, a co gorsza, wykrywanie tak kłopotliwych sytuacji w sposób algorytmiczny jest niewykonalne.

Własność druga dotyczy „czasowej" dostępności rozwiązań określonych problemów. Złożoność czasowa algorytmu określa bowiem, w jaki sposób dla coraz większych rozmiarów danych początkowych rośnie **czas rozwiązywania** problemu. Jeśli rośnie on zgodnie z pewną funkcją liniową (czyli wolno), mówi się o złożoności liniowej, jeśli rośnie zgodnie z pewną funkcją wykładniczą (czyli bardzo szybko), mówi się o złożoności wykładniczej etc[8].

Cechę złożoności czasowej przypisuje się również samym **problemom**: złożoność czasowa problemu P jest to złożoność najbardziej efektywnego (czyli najmniej złożonego) spośród wszystkich algorytmów rozwiązujących P. Na przykład: problem o złożoności wielomianowej to taki problem, dla którego istnieją rozwiązujące go algorytmy o złożoności wielomianowej (czyli niskiej), natomiast nie istnieją algorytmy o złożoności niższej.

II. Algorytmiczna metoda zdobywania wiedzy

6. Omówione wyżej informatyczne pojęcie algorytmu ma swój ważny odpowiednik w metodologii nauk, a jest nim **algorytmiczna metoda zdobywania wiedzy** (nie będąca, rzecz jasna, jedyną formą wiedzotwórczej aktywności naukowców). W ujęciu bardzo ogólnym metoda ta polega, po pierwsze, na symboliczno-regułowym zapisie wiedzy w postaci dogodnej

[8] Przykładowo: w przypadku algorytmu sortowania listy złożonej z n elementów rozmiarem danych wejściowych jest n, zaś funkcja złożoności czasowej $f(n)$ określa, ile elementarnych operacji komputera (a zatem: i czasu) potrzeba, by posortować listę o długości n.

do automatycznych inferencji, po drugie, na rozwiązywaniu właściwych danej nauce problemów poprzez konsekwentne stosowanie reguł symbolicznych, po trzecie zaś, na zapisywaniu szczególnie efektywnych schematów w postaci możliwych do dalszego wykorzystywania algorytmów[9].

Tak pojęta metoda ma swoje liczne zastosowania w naukach formalnych (jak matematyka) oraz silnie sformalizowanych (jak fizyka czy chemia)[10]. Można pokusić się nawet o stwierdzenie, że dana dyscyplina uzyskuje postać dojrzałą wówczas, gdy powstaje w jej obrębie pewien schematyczny **rachunek** pozwalający z powodzeniem stosować metodę algorytmiczną[11].

7. Ponieważ **formalną podstawą** metody algorytmicznej w nauce są różnego rodzaju algorytmy (albo: precyzyjne schematy, które można zapisać w postaci algorytmów), to zarówno siła, jak i słabość, tejże metody ma swoje źródło w ogólnych **własnościach algorytmów**[12].

O sile metody stanowią następujące cechy:

---

[9] Po raz pierwszy bodaj stosowanie tego rodzaju metody postulował żyjący w XVII i XVIII wieku G.W. Leibniz. Kluczowymi elementami jego propozycji były: i) *lingua characteristica* – czyli uniwersalny język symboliczny do wyrażania wszelkich myśli, oraz ii) *calculus ratiocinator* – niezawodny rachunek przekształcania symboli wspomnianego języka w celu dokonywania rozumowań. Ze wspomnianym pomysłem łączył Leibniz śmiałą ideę umaszynowienia rozumowań za pomocą maszyn liczących [zob. np. Marciszewski, Murawski, 1995].

[10] W przypadku tych ostatnich metoda algorytmiczna polega przede wszystkim na teoretycznym rozwiązywaniu problemów w ramach odpowiedniej teorii formalnej, zinterpretowanej w określonej dziedzinie problemowej. Dotyczy to np. fizyki, w obrębie której wiele problemów rozwiązuje się w drodze czysto formalnych przekształceń odpowiednich obiektów matematycznych.

[11] Kwestia ta ujawnia się najpełniej w matematyce, gdzie buduje się różne szczegółowe rachunki, czyli zbiory reguł/algorytmów pozwalających efektywnie przekształcać obiekty z danego działu matematyki (np. granice czy całki). Wymieńmy dla przykładu: rachunek granic, rachunek różniczkowo-całkowy, rachunek prawdopodobieństwa, rachunek zdań (logicznych).

[12] Poniższe punkt zostały sformułowane pierwotnie (w nieco uproszczonej postaci) w akademickim blogu dyskusyjnym Cafe Aleph. Zachęcam do dyskusji nad nimi w ramach wpisów dostępnych pod adresami: http://marciszewski.eu/?p=4032 oraz http://marciszewski.eu/?p=8234.

a) *ekonomia poznawcza*: jeden uniwersalny schemat (algorytm właśnie) reprezentuje nieskończoną liczbę rozwiązań problemów określonego typu; rozwiązania te uzyskujemy, stosując algorytm do odpowiednich danych;

b) *intersubiektywność i powtarzalność*: każda osoba, niezależnie od swoich prywatnych poglądów, przyzwyczajeń, nastrojów etc, stosując ten sam algorytm do tych samych danych, musi dojść do tego samego wyniku;

c) *możliwość automatyzacji*: każdy algorytm daje się zakodować w sposób możliwy do realizacji na niezawodnej i szybszej od człowieka maszynie;

d) *wiedzotwórczość*, która przejawia się na dwóch poziomach: d_1) każde zastosowanie algorytmu do nowych danych skutkuje nową wiedzą (rozwiązaniem nowego problemu), d_2) trafnie dobrany zbiór algorytmów ułatwia penetrację danej dziedziny na nowym jakościowo poziomie (na niższym poziomie dokonała się już automatyzacja).

O słabości metody algorytmicznej przesądzają cechy inne:

e) *mechaniczność*: realizacja algorytmu może przebiegać bezświadomie i bezrozumnie – nie wymaga ani rozumienia rozwiązywanego problemu, ani rozumienia istoty wykonywanych czynności;

f) *wtórność*: system działający algorytmicznie nie jest w pełni autonomiczny, realizuje skrupulatnie to, co zaplanował jego twórca; jeśli zaś system działa niedetermistycznie (z wykorzystaniem elementów losowych), to od użytkownika algorytmu zależy ocena generowanych rozwiązań i/lub określanie pewnych parametrów kierunkujących działanie algorytmu;

g) *niepełność*: teoretyczne badania nad algorytmami dowodzą, że nawet w dziedzinie zagadnień dobrze określonych istnieją problemy algorytmicznie nierozwiązywalne (zasadniczo lub praktycznie; zob. pkt. 10) [13].

[13] Powyższe punkty (zwłaszcza te, które opisują wady) pobudzają do dyskusji na temat stopnia przydatności metody algorytmicznej w nauce (zob. przypis 12). Rozwinięcie każdego z nich wymagałoby osobnego studium. W dalszej części tekstu skupię się na wadzie ostatniej, tj. niepełności, uwypuklając jej aspekty formalne (niezależne od zastosowań metody algorytmicznej w konkretnej dziedzinie problemowej).

8. Wskazane wyżej kwestie poznawczej ekonomii, intersubiektywności i niezawodności algorytmów prowadzą wprost do ciekawego pytania, czy w nauce obowiązuje **algorytmiczny wzorzec wiedzy** – wzorzec, który powinien motywować naukowców do zapisu wiedzy w postaci algorytmicznej?

Za odpowiedzią pozytywną przemawiają zarówno dotychczasowe ustalenia o charakterze historycznym, jak i metodologicznym. Po pierwsze bowiem, w rozwoju nauk ścisłych zawsze występowała żywa tendencja do formalizacji sprzęgniętej z jakimś rodzajem rachunku (zob. przypis 9). Po drugie zaś, wiedza zapisana w postaci algorytmicznej (i rozbudowywana w oparciu o algorytmy) spełnia szerokie wymagania racjonalizmu, a mianowicie wymóg intersubiektywnej komunikowalności i sprawdzalności [por. Ajdukiewicz 1965]. To zaś sprawia, że wzorzec algorytmizacji pozostaje zgodny z kluczową dla nauki ideą racjonalności (zgodność nie oznacza przy tym, że innego rodzaju podejście niż algorytmiczne nie jest w nauce racjonalne [por. Marciszewski 2013].

III. Algorytmiczna niedostępność wiedzy

9. Najpoważniejsze ograniczenia metody algorytmicznej – którym poświęcę dalszą część tekstu – wiążą się z jej **niepełnością** (zob. pkt 7g). Wyraża się ona w istnieniu problemów, które mają wprawdzie jakieś rozwiązania, ale nie sposób do nich dotrzeć za pomocą określonego typu algorytmów. W perspektywie algorytmicznej zatem tożsama z tymi rozwiązaniami wiedza okazuje się **niedostępna**.

Wspomniane ograniczenia trzeba nazwać **formalnymi**, ponieważ wynikają one z czysto informatycznych własności algorytmów i jako takie są niezależne od interpretacji (czyli zastosowania) algorytmu w danej dziedzinie problemowej. Ograniczenia te są ponadto zrelatywizowane do modelu obliczeń, w ramach którego poszukuje się takich czy innych algorytmów. Znaczy to, że ograniczenia występujące w pewnym modelu (np. cyfrowym) mogą nie obowiązywać w modelu innym (np. analogowym).

10. W obejmującym większość współcześnie stosowanych algorytmów **modelu cyfrowym** występują dwa typy problemów, których dotyczą opisane wyżej ograniczenia formalne.

Są to:

a) problemy nierozwiązywalne (nieobliczalne) **zasadniczo** – to znaczy takie, których nie można rozwiązać we wszystkich przypadkach szczególnych, za pomocą jednego uniwersalnego algorytmu. Kanonicznym przykładem zagadnienia tego typu jest problem stopu maszyny Turinga[14] [zob. Turing, 1936].

b) problemy nierozwiązywalne (nieobliczalne) **praktycznie** – to znaczy takie, dla których nie istnieją algorytmy o niższej złożoności czasowej niż wykładnicza. Dla coraz większych danych problemy te wymagają rosnącej lawinowo ilości czasu (co przesądza o braku algorytmicznego dostępu do ich rozwiązań). Jako typowe zagadnienia tego typu wskazuje się: problem komiwojażera oraz problem spełnialności formuł klasycznego rachunku zdań[15] [zob. np. Harel, 2000].

Ze względu na niemożność efektywnego rozwiązania wyżej wymienionych problemów w modelu cyfrowym wiedza o ich rozwiązaniach pozostaje algorytmicznie niedostępna (nawet wówczas, gdy rozwiązania takie istnieją).

Warto podkreślić, że w przypadku zagadnień pierwszego typu jest ona taka, ponieważ odpowiednim algorytmom nie przysługuje **własność stopu** (zob. pkt 5). Znaczy to, że we wszystkich możliwych do zastosowania algorytmach muszą wystąpić potencjalnie nieskończone pętle – takie pętle,

---

[14] Problem ten wyraża się pytaniem „Czy istnieje pewna maszyna Turinga MT (inaczej: pewien algorytm cyfrowy), która otrzymując na wejściu kod innej dowolnej maszyny Turinga MT_i oraz jej dowolnych danych wejściowych D_j, jest w stanie jednoznacznie odpowiedzieć na pytanie o to, czy MT_i kończy pracę dla D_j?" Można go uznać za kanoniczny, ponieważ nierozwiązywalności innych problemów nieobliczalnych dowodzi się poprzez sprowadzenie tychże problemów do zagadnienia stopu.

[15] Pierwszy ma charakterystykę następującą: „Dla danej sieci *n* miast oraz łączących je dróg o określonych długościach znajdź najkrótszą trasę objazdu wszystkich miast". Drugi można opisać tak: „Dla danej *n*-zdaniowej formuły klasycznego rachunku zdań sprawdź, czy istnieje takie wartościowanie zmiennych zdaniowych, przy którym formuła jest prawdziwa".

co do których nie sposób algorytmicznie stwierdzić, czy faktycznie biegną one w nieskończoność.

Wiedza dotycząca problemów drugiego rodzaju pozostaje niedostępna (tym razem: w praktyce) z powodu innej własności algorytmów: hipotetycznym programom, które mogłyby rozwiązać dany problem, nie przysługuje odpowiednio niska **złożoność czasowa**.

11. Wobec istnienia problemów **nierozwiązywalnych cyfrowo** zachodzi ważkie pytanie o strategie informatyczne[16], które – mimo wszystko – pozwalają się z tymi problemami mierzyć.

Ich różne możliwe warianty sprowadzają się w istocie do dwóch typów czynności: a) do prób takiego **przeformułowania** oryginalnego problemu, by stał się on efektywnie rozwiązywalny w modelu cyfrowym, albo b) do poszukiwania **alternatywnych** modeli obliczeń (względem cyfrowego), które poszerzyłyby klasę zadań efektywnie rozwiązywalnych.

11a. Wybierając strategię pierwszą, uznaje się wstępnie, że w praktyce, tj. dla potrzeb realnych zastosowań, wystarczy znać rozwiązania pewnych „łatwiejszych" odpowiedników danego problemu nieobliczalnego. Poszukiwanie owych odpowiedników przebiega różnie, zależnie od typu wchodzącej w grę nieobliczalności.

Jeśli chodzi o zagadnienia nieobliczalne zasadniczo (zob. pkt 10a), to godząc się na ich algorytmiczną nierozwiązywalność, usiłuje się wydzielać z nich takie **podproblemy** (zazwyczaj chodzi o problemy praktycznie istotne), dla których istnieją wystarczająco efektywne **algorytmy lokalne**. Na przykład: próbując rozwiązywać równania diofantyczne (mamy tu do czynienia z typowym zagadnieniem nieobliczalnym), identyfikuje się pewne ich użyteczne podzbiory (jak równania liniowe), a następnie szuka się dla nich wyspecjalizowanych, odpowiednio efektywnych, algorytmów cyfrowych[17].

---

[16] Mówiąc o strategii informatycznej, pozostajemy w kręgu metod algorytmicznych – niekoniecznie jednak cyfrowych.

[17] Podkreślmy jednak, że opisana strategia podziału problemu nieobliczalnego na podproblemy musi pozostać „nieskończenie ułomna". Ewentualnych podproblemów, które mogłyby zostać rozwiązane za pomocą efektywnych algorytmów lokalnych, musi być nie-

Jeśli chodzi o problemy nieobliczalne praktycznie (zob. pkt 10b), to najczęściej „osłabia się je", formułując takie warunki rozwiązywalności, które po pierwsze, wystarczają do **praktycznej realizacji** pewnych zadań, a po drugie, zapewniają efektywny algorytm znajdowania wyniku. Na przykład: w problemie komiwojażera poszukuje się nie najkrótszej możliwej drogi objazdu *n* miast (drogi *s*), lecz drogi co najwyżej dwa razy dłuższej niż najkrótsza ($\leq 2s$).

Podsumowując: strategie typu 11a, choć wydają się pragmatycznie interesujące, nie uwalniają nas od kłopotów z algorytmicznym dostępem do wiedzy. W każdym z rozważanych wyżej wariantów oryginalny problem **pozostaje algorytmicznie nierozwiązywalny** – efektywne rozwiązanie zyskuje bowiem problem inny, a mianowicie pewien zmieniony lub okrojony odpowiednik zagadnienia pierwotnego.

11b. Wybierając strategię drugą, polegającą na zmianie modelu obliczeń, nie zakłada się wstępnie konieczności przeformułowania oryginalnego problemu. Zagadnienia cyfrowo nieobliczalne traktuje się jako naukowe **wyzwanie** – wyzwanie, które może doprowadzić do takiej (niecyfrowej) teorii przetwarzania danych, która będzie miała efektywne „przełożenie" na faktyczne procedury realizowane przez pewnego typu maszyny[18]. Kwestię tę rozwinę w kolejnym punkcie.

12. Alternatywne modele operacji algorytmicznych, do poszukiwania których skłania fakt istnienia problemów nieobliczalnych cyfrowo, określa się coraz częściej zbiorczą nazwą **hiperobliczeń** (ang. *hypercomputations*). Nazwa ta ma zwrócić uwagę na fakt, że chodzi o wszelkie techniki obli-

---

skończenie wiele – gdyby bowiem było ich skończenie wiele, to dostosowane do nich algorytmy udałoby się połączyć, nawet na zasadzie prostego zsumowania, w jeden algorytm uniwersalny (a taki przecież nie istnieje ze względu na zasadniczą nierozwiązywalność problemu pierwotnego).

[18] Zauważmy, że tak właśnie stało się z teorią Turinga, która wyprzedziła o kilka lat realne rozwiązania w dziedzinie elektronicznych maszyn cyfrowych z wymiennym oprogramowaniem.

czeniowe, które, teoretycznie rzecz biorąc, pozwalają rozwiązywać problemy nieosiągalne dla maszyn Turinga[19].

Choć teoria technik alternatywnych jest już dość dobrze rozwinięta, to z metodologicznego punktu widzenia wciąż żywe są **wątpliwości**: a) co do samego pojęcia hiperobliczeń – a więc typów technik, które wolno określić mianem pozaturingowskich, oraz b) co do praktycznej realizowalności różnych typów hiperobliczeń.

12a. Jeśli chodzi o kwestię pierwszą, to wydaje się, że najbardziej ogólne pojęcie hiperobliczeń uzyskuje się poprzez takie **poszerzanie** modelu turingowskiego, które polega na modyfikowaniu co najmniej jednej z kluczowych cech tego modelu, a zatem:

i) *dyskretności* (cyfrowości), ii) *skończoności* (skończona liczba operacji wykonywanych w skończonym czasie), oraz iii) *determinizmu* (ściśle określony schemat przetwarzania danych).

I tak:

i') zastępując cechę dyskretności szerszą od niej własnością ciągłości danych, definiuje się wstępnie hiperobliczenia **analogowe** – czyli takie, które umożliwiają przetwarzanie sygnałów ciągłych, opisywanych matematycznie za pomocą liczb rzeczywistych;

ii') rezygnując z cechy skończoności obliczeń (na rzecz ich nieskończoności), postuluje się istnienie hiperobliczeń **infinitystycznych** – to znaczy takich, które pozwalają realizować w skończonym czasie nieskończoną liczbę operacji;

iii') porzucając wymóg determinizmu kolejnych kroków obliczeń, uzyskuje się różnego rodzaju hiperobliczenia **indeterministyczne** – zależne w swoim przebiegu od występowania pewnych zdarzeń losowych[20].

---

[19] Przy okazji jest to nazwa przemawiająca do wyobraźni.

[20] Osadzając powyższe rozróżnienia ogólne w realnych badaniach, warto wymienić konkretne przykłady technik każdego rodzaju. I tak: do technik analogowych zalicza się pewne uogólnienia tradycyjnego modelu Shannona [zob. Shannon, 1941 oraz Pour-El, 1974], a także schematy przetwarzania właściwe rekurencyjnym sieciom neuronowym [zob. Siegelmann, 1998]; do technik infinitystycznych należą hipotetyczne obliczenia tzw. przyspieszających maszyn Turinga [zob. Shagrir, 2004] i relatywistycznych maszyn Turinga [zob. Hogarth, 1994], z pewnością obliczenia kwantowe, a także ewolucyjne [zob. Michalewicz, 1992].

Podkreślić trzeba, że każdy ze wzmiankowanych typów hiperobliczeń, podzielonych dalej na różne podtypy, stanowi przedmiot osobnej **teorii przetwarzania danych**[21]. Każdą z nich można postrzegać jako szeroką ramę pojęciową, która wyznacza jeden z kierunków poszerzania tradycyjnego pojęcia algorytmu (zob. pkt 3b). Co ważne jednak, mnogość wzmiankowanych teorii, a także brak jakiejś ich jednolitej podstawy (w rodzaju uniwersalnej maszyny Turinga), znacznie utrudnia dyskusję nad algorytmiczną dostępnością wiedzy w ramach szerokiego pojęcia algorytmu. Sprawę komplikuje jeszcze bardziej kwestia praktycznej (również: fizycznej) realizowalności hiperobliczeń.

12b. Trudności z praktyczną realizacją technik pozaturingowskich wynikają z faktu, że teoretyczne pojęcia ciągłości (podstawa technik analogowych), nieskończoności (podstawa technik infinitystycznych) oraz indeterminizmu, niekoniecznie muszą przystawać do **świata**, z którego pochodzą przecież realne nośniki danych przetwarzanych przez fizyczne automaty. Przykładowo: gdyby świat był dyskretny (skwantowany) i skończony, a przy takim założeniu są konstruowane wszelkie współczesne automaty, to żadnych obliczeń analogowych ani infinitystycznych nie moglibyśmy nigdy przeprowadzić[22].

Wydaje się, że kluczowa jest tu kwestia **nieskończoności**. Nawet bowiem w przypadku technik analogowych poznanie rozwiązań pewnych problemów nieobliczalnych (a więc analogowy dostęp do pewnego rodzaju wiedzy) wymaga nieskończonej dokładności odczytu wygenerowanego wyniku. Ta zaś wydaje się leżeć poza zasięgiem ludzkich władz poznaw-

---

[21] Oczywiście niecnie stoi na przeszkodzie, by rozważać teorię różnych kombinacji powyższych typów obliczeń, na przykład indeterministycznych i analogowych zarazem, albo infinitystycznych i zarazem indeterministycznych.

[22] Co do cechy indeterminizmu, to w świetle współczesnych teorii fizycznych (zwłaszcza fizyki kwantowej) wydaje się ona faktyczną własnością mikroświata. Mimo to trwają dyskusje, czy czyste techniki indeterministyczne (bez wykorzystywania ciągłości i/lub nieskończoności) są faktycznie hiperobliczeniami, to znaczy, czy zapewniają pokonanie bariery nieobliczalności zasadniczej. Na przykład: obliczenia kwantowe zwiększają jedynie szybkość operacji, czyniąc praktycznie rozwiązywalnymi problemy o złożoności wykładniczej lub silniowej.

czych i konstruowanych przez człowieka urządzeń pomiarowych [por. Stacewicz, 2015].

13. W formie krótkiego podsumowania poczynionych wyżej uwag chciałbym uwypuklić dwie kwestie.

Po pierwsze, problem algorytmicznego dostępu do wiedzy jest najlepiej zbadany w przypadku **algorytmów cyfrowych**: mamy tu do czynienia ze ściśle określonymi zagadnieniami (jak problem stopu czy komiwojażera), o rozwiązaniach których nie możemy uzyskać wiedzy w sposób efektywnie algorytmiczny; a dzieje się tak mimo popularnej wśród informatyków, stricte pragmatycznej, strategii zastępowania wymienionych zagadnień problemami podobnymi, choć algorytmicznie dostępnymi.

Po drugie, z teoretycznych badań nad różnymi **alternatywnymi modelami obliczeń** (hiperobliczeń) wyłania się stopniowo szersze od cyfrowego pojęcie algorytmu; w jego ramach można mówić o szerszym algorytmicznym dostępie do rozwiązań problemów cyfrowo nieobliczalnych. Jak na razie jednak, owo szersze podejście wydaje się zbyt teoretyczne (chodzi bardziej o modele obliczeń niż ich realne zastosowania), zbyt pluralistyczne (proponuje się wiele, luźno powiązanych ze sobą teorii), a także mocno wątpliwe co do perspektyw praktycznej realizacji w sposób zakładany przez taką czy inną teorię.

Bibliografia

Ajdukiewicz K., (1965), *Logika pragmatyczna*, Warszawa, PWN.

Cafe Aleph (http://blog.marciszewski.eu/), akademicki blog dyskusyjny W. Marciszewskiego i P. Stacewicza.

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

Hogarth M., (1994), "Non-Turing computers and non-Turing computability", PSA vol. 1, s. 126–138.

Kordos M., (1994), *Wykłady z historii matematyki*, Warszawa, Wydawnictwa Szkolne i Pedagogiczne.

Marciszewski W., (2013), „Empiryzm, racjonalizm, irracjonalizm po przełomach naukowych XX wieku", [w:] *Przewodnik po epistemologii*, red. R. Ziemińska, Kraków, WAM.

Marciszewski W., Murawski R., (1995), *Mechanisation of Reasoning in a Historical Perspective*, Amsterdam–Atlanta, Rodopi.
Marciszewski W., Stacewicz P., (2011), *Umysł – komputer – świat. O zagadce umysłu z informatycznego punktu widzenia*, Warszawa, Akademicka Oficyna Wydawnicza EXIT.
Michalewicz Z., (1992), *Genetic Alghorithms + Data Structures = Evolution Programs*, Berlin, Springer.
Mostowski A.W., Pawlak Z., (1970), *Logika dla inżynierów*, PWN, Warszawa.
Mycka J., Olszewski A., (2015), „Czy teza Churcha ma jeszcze jakieś znaczenie dla informatyki?", [w:] *Informatyka a filozofia. Od informatyki i jej zastosowań do światopoglądu informatycznego*, red. P. Stacewicz, Warszawa, Oficyna Wydawnicza Politechniki Warszawskiej.
Mycka J., Piekarz M., (2004), „Przegląd zagadnień obliczalności analogowej", [w:] *Algorytmy, metody i programy naukowe*, red. S. Grzegórski, M. Miłosz, P. Muryjas, Lublin, Polskie Towarzystwo Informatyczne, s. 125–132.
Pour-El M.B., (1974), "Abstract Computability and its Relations to the General Purpose Analog Computer", *Transactions of the American Mathematical Society*, nr 199, s. 1–28.
Shagrir O., (2004), "Super-tasks, accelerating Turing machines and uncomputability", *Theoretical Computer Science*, 317, s. 105–114.
Shannon C., (1941), "Mathematical theory of differential analyzer, *Journal of Mathematical Physics*, nr 20, s. 337–354.
Siegelmann H.T., (1998), *Neural Networks and Analog Computation: Beyond the Turing Limit*, Boston, Birkhauser.
Stacewicz P., (2010), *Umysł a modele maszyn uczących się. Współczesne badania informatyczne w oczach filozofa*, Warszawa, Akademicka Oficyna Wydawnicza EXIT.
Stacewicz P., (2015), *Informatyczne kłopoty z nieskończonością?*, [w:] *Filozofia matematyki i informatyki*, red. R. Murawski, Kraków, Copernicus Center Press, s. 310––327.
Turing A.M., (1936), *On Computable Numbers, with an Application to the Entscheidungsproblem*, „Proceedings of The London Mathematical Society", 42, s. 230––265.
Wirth N., (1989), *Algorytmy + struktury danych = programy*, Wydawnictwa Naukowo-Techniczne, Warszawa 1989.

Paweł Stacewicz,
Zakład Filozofii Nauki, Socjologii i Podstaw Techniki,
Wydział Administracji i Nauk Społecznych,
Politechnika Warszawska,
Plac Politechniki 1,
00-661 Warszawa.
Telefon: (22) 234 15 51
e-mail: p.stacewicz@ans.pw.edu.pl