Ks. Adam Olszewski
Papieska Akademia Teologiczna, Kraków

MASZYNA POSTA

Nazwisko Emila Posta związane jest z różnymi zagadnieniami logiki formalnej. Dla przeciętnego studenta kojarzy się z badaniami metalogicznymi dotyczącymi rachunków zdaniowych, w szczególności z powstaniem logik wielowartościowych, czy też z twierdzeniem o pełności klasycznego rachunku zdań.

Znana anegdota opowiada, iż Tarski w rozmowie z Postem stwierdził, że Post jest amerykańskim logikiem, jedynym nie-Polakiem, który tak wiele dokonał w dziedzinie badań logicznych nad rachunkami zdaniowymi. Tarski relacjonował, iż Post miał odpowiedzieć, że urodził się w Polsce. Rzeczywiście, urodził się w żydowskiej rodzinie 11 II 1897 roku w Augustowie. Wraz z rodzicami wyemigrował do Nowego Yorku w 1904 roku.[1] Zasługi Posta dotyczą również zagadnień związanych z rozstrzygalnością. Znane są tzw. języki Posta: języki kanoniczne, języki normalne oraz tag-system. Post, jak pokazują to badania historyczne, wyprzedził idee Gödla i Churcha w badaniach nad rozstrzygalnością i niezupełnością.[2] Kiedy jednak mówi się, w kontekście zagadnień związanych z rozstrzygalnością, o abstrakcyjnych maszynach matematycznych, to zazwyczaj w pierwszym rzędzie kojarzone są maszyny Turinga. Nie wszyscy wiedzą, że Post stworzył koncepcję abstrakcyjnej maszyny, zwanej na jego cześć maszyną Posta, i to niezależnie od Turinga. Koncepcja jego maszyny opisana jest w artykule *Skończone procesy kombinatoryczne – sformułowanie 1*, który ukazał się w numerze kwartalnika „The Journal of Symbolic Logic" (vol. 1, nr 3, s. 103-105,

---

[1] Por. W M a r c i s z e w s k i, *Can Turing-Post Machine Produce Perequisites for Automated Reasoning?*, w: *Emil L. Post and the Problem of Mechanical Provability*, Białystok 1998, s. 8.

[2] Por. R. M u r a w s k i, *E.L. Post and the development of mathematical logic and recursion theory*, w: *Emil Post and the Problem of Mechanical Provability*, Białystok 1998, s. 21-22. Tam też cytowany jest fragment listu Posta do Gödla, gdzie Post pisze: „...perhaps the best I can say is that I would have proved Gödel's Theorems in 1921 – had I been Gödel." s. 22.

z końca 1936 roku).[3] Ten numer zaopatrzony jest w następującą notkę edytorską:

> Otrzymane 7 października 1936. Czytelnik powinien porównać artykuł napisany przez A.M. Turinga *O liczbach obliczalnych*, który ma się niebawem ukazać w „Proceedings of the London Mathematical Society" Jednak niniejszy artykuł, pomimo tego, że nosi późniejszą datę, został napisany całkowicie niezależnie od artykułu Turinga.[4]

Maszyna Posta (w skrócie MP) jest, podobnie jak choćby maszyna Turinga, maszyną abstrakcyjną. Wyjaśnienia wymaga słowo „abstrakcyjna" Znaczy to, że taka maszyna nie istnieje w rzeczywistości fizycznej, ponieważ w idei takiej maszyny usunięto ograniczenia nałożone przez prawa fizyki, odnoszące się do czasu i przestrzeni. Maszyna bowiem może działać dowolnie długo i dysponować nieskończoną taśmą podzieloną na jednakowe kratki. Taśma ta jest nieskończona w lewo i w prawo, podobnie jak prosta na płaszczyźnie, z tym że kratek jest tyle samo ile jest liczb całkowitych i są one ponumerowane za pomocą liczb całkowitych zgodnie z ich zwykłym uporządkowaniem.[5] Taka taśma jest pierwszą częścią tworzącą MP (sam Post nazywa taśmę *symbol space*, Post [1936] s. 103). Drugą częścią tworzącą MP jest urządzenie czytające – lub czytnik (wg Posta *worker*, albo *problem solver*, Post [1936] s. 103). Niemal dokładnie tak samo jest w konstrukcji maszyny Turinga. Każda z kratek taśmy jest pusta albo nie jest pusta. Dla obu sytuacji dysponujemy symbolami, które MP rozpoznaje. Mamy symbol 0, oznaczający, że kratka jest pusta; symbol I oznacza, że kratka nie jest pusta.[6] Informacja o tym, które kratki są oznaczone symbolem 0, a które symbolem I, określa *stan taśmy*. Czytnik porusza się wzdłuż taśmy w dyskretnym czasie, to znaczy nieciągle. W dowolnym momencie czasowym czytnik czyta jakąś konkretną kratkę. Każdy ruch czytnika zwiemy *krokiem*. Stan taśmy wraz z informacją o tym, nad którą z kratek znajduje się aktualnie czytnik, zwiemy *wewnętrznym stanem MP* [7]

Do tego podstawowego artykułu będziemy się w dalszym ciągu pracy odnosić według następującej konwencji: Post [1936] s. ...; podamy tu numer strony.

[4] Dane artykułu Turinga: A.M. T u r i n g. *On computable numbers, with an application to the Entscheidungsproblem*. „Proceedings of the London Mathematical Society" ser. 2, 42, nr 3, 4, (1936) s. 230-265; poprawka – tamże, 43, nr 7, (1937) s. 544-546. Z powodu poprawki artykuł Turinga niesłusznie przytaczany jest z datą 1936/1937.

[5] Liczby całkowite tworzą w pewnym sensie układ współrzędnych, dzięki któremu możemy identyfikować poszczególne kratki. Taśma jest przez to jakby 'przestrzenią jednowymiarową'

[6] W ogólnym przypadku MP może rozpoznawać (dysponować) większą, skończoną ilością symboli. Post w swoim artykule rozważał maszyny z dwoma symbolami: symbolem I oraz symbolem 0 oznaczającym pole puste.

[7] W idei maszyny Turinga występuje pojęcie stanu wewnętrznego, ale znaczy zupełnie co innego, choć pełni podobną rolę. Czytnik maszyny Turinga posiada skończoną liczbę stanów wewnętrznych.

MP może wykonywać operacje składające się z ruchów maszyny w lewo lub prawo, wpisywania lub wymazywania symboli. Operacja przebiega w zgodzie z *programem*, który z kolei składa się z *instrukcji* wykonywanych przez maszynę. Dowolna instrukcja, czyli elementarny krok maszyny, musi podpadać pod jeden z następujących schematów:

(1) rusz w prawo do następnej[8] kratki – symbolicznie: $i. \rightarrow j.$,
(2) rusz w lewo do następnej kratki – symbolicznie: $i. \leftarrow j.$,
(3) wpisz do obserwowanej kratki symbol I – symbolicznie: $i.\ \mathrm{I}\, j.$,
(4) wymaż symbol wpisany do obserwowanej kratki – symbolicznie: $i.\ 0\, j.$,
(5) zrób ruch warunkowy – symbolicznie: $i.\ ?\, j. / k.$,
(6) zrób ruch stop – symbolicznie: $i.\ \#$,

gdzie i, j, k, przebiegają zbiór liczb naturalnych bez zera.

Niektóre z powyższych schematów instrukcji MP wymagają komentarza. I tak, numer na początku każdej instrukcji (w naszym przypadku „*i.*") jest numerem instrukcji, która jest wykonywana. Wszystkie cztery pierwsze schematy instrukcji na końcu mają „*j.*" – jest to liczba oznaczająca numer instrukcji, którą maszyna ma wykonać jako następną.

W schemacie (4) maszyna ma wymazać symbol z obserwowanej kratki, czyli kratka ma być pusta, a to znaczy, że ma się w niej znaleźć symbol 0 i tego typu instrukcja może być wykonana, jeżeli obserwowana kratka jest niepusta. Podobna uwaga dotyczy schematu (3): taka instrukcja może być wykonana, gdy kratka jest pusta. W przeciwnym przypadku instrukcje są traktowane jako niewykonalne.

Schemat (5) należy rozumieć następująco: w *i*-tej instrukcji wykonasz ruch w zależności od tego, co rozpoznasz[9] w obserwowanej kratce; jeśli obserwowana kratka zawiera symbol 0, to przejdź do instrukcji o numerze *j*, a jeśli w obserwowanej kratce rozpoznasz symbol I, to przejdź do instrukcji o numerze *k*. W przypadku schematu (6) maszyna się zatrzyma. Instrukcje podpadające pod schematy (5) oraz (6) nie zmieniają stanu wewnętrznego MP

MP może: albo się zatrzymać, wypełniając instrukcję typu (6) – tzw. *poprawny stop*, albo się zatrzymać, nie mogąc wykonać niewykonalnej instrukcji – tzw. *niepoprawny stop*, albo nigdy się nie zatrzymać.

Programem dla MP nazwiemy dowolny skończony, niepusty ciąg instrukcji podpadających pod któryś z sześciu wymienionych schematów.

W poprawnym programie MP instrukcje muszą być ułożone według kolejności numerów instrukcji (tzn. *i*-ta instrukcja na *i*-tym miejscu w kolejności pojawienia się programie). Drugi wymóg poprawności programu

---

[8] Określenie 'następnej' znaczy: kolejnej zgodnie z porządkiem określonym w zbiorze liczb całkowitych.

[9] Maszyna bowiem umie rozpoznać, czy obserwowana kratka jest pusta czy też nie jest pusta. Post [1936] s. 103.

jest taki, że jeśli maszyna jakąś instrukcję wykona i na mocy tej wykonanej instrukcji ma przejść do instrukcji o jakimś numerze, to w programie musi być instrukcja o tym wskazanym numerze.

Aby program dla MP mógł zacząć działać, należy go zapisać i ustawić maszynę nad polem o współrzędnej zero (rodzaj standardowej pozycji) z określonym stanem taśmy.

Przykład programu dla MP:

1. I 4.
2. 0 3.
3. ← 2.
4. → 5.
5. ? 4./3.

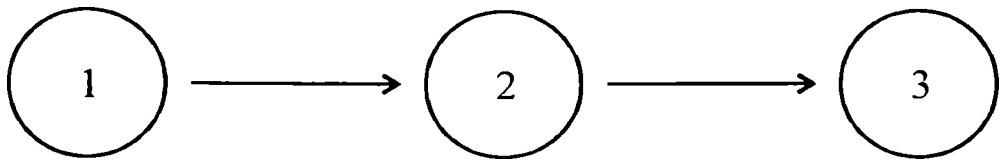
Aby ten program mógł zadziałać, należy dodatkowo określić stan taśmy. Otóż czytnik jest nad kratką o współrzędnej zero. Wszystkie kratki mają wpisany symbol 0, oprócz kratki o współrzędnej dwa, gdzie wpisany jest symbol I.[10] Program ten przy zadanym stanie taśmy, jak Czytelnik może sam sprawdzić, doprowadzi do niepoprawnego stopu maszyny. Oczywiście inny stan taśmy dla tego samego programu może dać inny rezultat, jak i inny program dla tego samego stanu taśmy może dać inny rezultat.

Jedną z najprostszych operacji obliczalnych, określonych w zbiorze liczb naturalnych, jest z pewnością funkcja następnika. Ogólnie rzecz biorąc, w naszej stylizacji maszyny Posta chodzi o to, by na taśmie była zapisana zadana z góry liczba naturalna n, którą na mocy przyjętej obecnie konwencji przedstawiać będziemy w postaci nieprzerwanego ciągu $n+1$ symboli I. Tak więc zero reprezentuje jeden symbol I, prócz niego taśma jest pusta. Konwencja ta ułatwia pisanie programów. Następujący program wykonuje operację następnika dla stanu wewnętrznego MP, gdy na taśmie jest zapisany skończony, nieprzerwany ciąg symboli I, zaś czytnik znajduje się nad pierwszym symbolem I z lewej strony. Oprócz tego pozostałe kratki taśmy są puste.

1. ← 2.
2. I 3.
3. #

Program ten wypisze jeden symbol I bezpośrednio w polu na lewo od nieprzerwanego ciągu I-ek. Można ten prosty program przedstawić w postaci graficznej, za pomocą tzw. *diagramu*.

[10] Przykład ten pochodzi z pracy V.A. Uspensky, *Post's Machine*, Moscow 1983: s. 16. Książeczka Uspienskiego w dużym stopniu oddziałała na niniejszą prezentację.



Opis tego diagramu jest bardzo prosty. Do okręgów wpisane są numery reprezentujące kolejne wiersze programu. Strzałki skierowane są do tych okręgów, które reprezentują wiersze programu, do którego ma przejeść MP po wykonaniu zadanej w bieżącym wierszu operacji. Naturalnie, dla złożonych programów diagram jest bardziej skomplikowany. Możliwe są oczywiście diagramy, gdzie z jednego okręgu wychodzą dwie strzałki, co odzwierciedla sytuację polecenia z instrukcji podpadającej pod schemat (5). Rysunek powyższy (i inne) pełnić może rolę heurystycznego argumentu za równoważnością maszyn Posta i maszyn Turinga. Tego typu diagramy dla maszyny Turinga zwane są często z angielska *flow graph*.[11] Dla tych maszyn (Turinga) numery w okręgach oznaczają stany wewnętrzne.[12]

Kompletny program dla MP realizujący operację następnika dla najbardziej ogólnego stanu taśmy wyglądać może tak:[13]

1. ? 2./3.
2. ← 4.
3. → 4.
4. ? 5./3.
5. I 6.
6. ← 7.
7. ? 8./15.
8. → 9.
9. ? 10./8.
10. I 11
11 → 12.
12. ? 13./19.
13. ← 14.
14. ? 5./13.
15. → 16.
16. → 17
17 ? 23./18.
18. 0 16.
19. ← 20.

[11] Por. na przykład znakomitą książkę: G. Boolos, R. Jeffrey, *Computability and Logic*, 3. ed., Cambridge 1989, oraz A.W. Mostowski, Z. Pawlak, *Logika dla inżynierów*, Warszawa 1970, s. 236-242.

[12] Por. przyp. 7.

[13] Por. V.A. Uspensky, *Post's Machine*, Moscow 1983, s. 39-48.

20. ← 21.
21. ? 23./22.
22. 0 20.
23. #

Dla tego programu można naturalnie zbudować diagram. Gdy nad strzałkami diagramu zapiszemy operacje, które wykonuje MP, to uzyskamy niemal jednoznaczny *flow graph* maszyny Turinga.

Jak się wydaje, Emil Post był postacią nieprzeciętną. Na przykładzie jego i innych wybitnych logików widać, jak niepoślednią rolę w badaniach logicznych odgrywa baza filozoficzna. Post również filozofował przy okazji swoich badań logicznych.[14] W artykule Post [1936] s. 104-105, Post pisze: 'Piszący (the writer; tzn. sam Post – przyp. A.O.) spodziewa się, że niniejsze sformułowanie okaże się być logicznie równoważne rekurencyjności w sensie rozwiniętym przez Gödla-Churcha.' (tłum. A.O.). Choć tego nie dowodzi, to jednak jego przypuszczenie okazało się słuszne. Poza tym uważa, że wartość tego sformułowania (w postaci maszyn Posta) służyć ma 'psychologicznej wierności' (*psychological fidelity*) dla hipotezy roboczej (*working hypothesis*), zatem wymagającej jeszcze pracy, na rzecz tego, co sformułował Church – mianowicie identyfikacji efektywnej obliczalności z rekurencyjnością, czyli Tezy Churcha. Sam Church napisał na temat artykułu Posta krytyczną i 'kwaśną' recenzję.[15]

Post uważał, podobnie zresztą jak Gödel, że matematyczne pojęcie obliczalności posiada charakter absolutny, niezależny od przyjętego formalizmu. Proponował jednak, aby nie traktować powyższej 'hipotezy roboczej' ani jako definicji (jak tego chciał Church), ani jako aksjomatu, ale jako rodzaj *prawa natury*. Sądzę więc, że hipoteza ta, wraz z niezupełnością arytmetyki Peano oraz wynikami Churcha o nierozstrzygalności pewnych problemów, wypowiada *prawdy*, odnoszące się do formalizmu jako takiego.

POST'S MACHINE

Summary

The aim of the article is to present the main idea of the so called Post's Machine. Contrary to Turing's Machine, Post's Machine is not so widely known. But some find it much simpler than the former. We begin by giving a short history of Post's publication, then we describe the definition of the Machine and give some easy examples of it. And finally we make some general philosophical remarks concerning Church's Thesis in this context.

[14] Przedstawienie interesujących filozoficznie koncepcji Posta wymagałoby odrębnego artykułu.

[15] A. C h u r c h, *Reviev of Post [1936]*, „The Journal of Symbolic Logic" 2: 1937 nr 1, s. 43.