Mechanika, komputer, człowiek (III)

Prof. dr Dominik ROGULA

Algorytmy, obliczalność i rozstrzygalność

Przykładów algorytmów dostarcza w obfitości arytmetyka — wykonywanie działań podstawowych, wyszukiwanie liczb pierwszych itp. Jako inny przykład przytoczymy pewien algorytm rysowania na kartce papieru.
Niech operacjami podstawowymi będą:
up — podnieś rysik (tak by nie dotykał kartki),
down — opuść rysik (tak by dotykał kartki),
move (a, b) — przesuń (prostoliniowo) rysik do punktu o współrzędnych (a, b),
end — zakończ.
Wówczas ciąg instrukcji na marginesie przedstawia algorytm rysowania znaku × o środku w punkcie (0, 0) i długości ramienia *a*.

up
move (a, a)
down
move (−a, −a)
up
move (−a, a)
down
move (a, −a)
up
end

Intuicyjne pojęcie algorytmu jest dość jasne: algorytm to zupełnie jednoznacznie określona procedura, czyli sposób postępowania. Algorytm nie może — w żadnej sytuacji, której dotyczy — pozostawiać wątpliwości, czy dalej postąpić tak, czy inaczej. Musi też jednoznacznie określać swe zakończenie.
Poszczególne „kroki" algorytmu muszą być efektywnie wykonalne, dając zawsze ten sam wynik, przynajmniej w zakresie istotnym dla zadania, którego algorytm dotyczy.
Mimo to, pojęcie algorytmu jest na tyle pierwotne w ludzkim pojmowaniu spraw tego świata, że nie oczekujemy, iż jakaś uznana gałąź nauki dostarczy nam jego ścisłej definicji. Zamiast tego mamy szereg formalnych odpowiedników intuicyjnego pojęcia algorytmu.
Mówiąc o algorytmie, abstrahujemy od materialnej strony jego realizacji (liczydło, palce, napisy). Dlatego zamiast mówić o obiektach, których dotyczy algorytm, i o sytuacjach, w jakich obiekty te uczestniczą w kolejnych stadiach algorytmu, możemy mówić o znakach, które w jakikolwiek umowny sposób oznaczają te obiekty i sytuacje. Algorytm widziany od tej formalno-lingwistycznej strony, to dokładnie określony przepis na dokonywanie zmian w ciągu znaków. Przyjmuje się przy tym, że zbiór znaków dopuszczalnych w tych napisach (czyli tzw. *alfabet*) jest skończony, same napisy są skończone, no i oczywiście opis reguł określających algorytm też jest skończony. Rodzaj znaków alfabetu, ani nawet ich liczba, nie są przy tym istotne, a to wskutek możliwości jednoznacznych i prostych przekodowań napisów.
Możliwa jest także „arytmetyzacja" algorytmów. Mając dowolny alfabet

$$A = \{a_1, a_2, \ldots, a_n\}$$

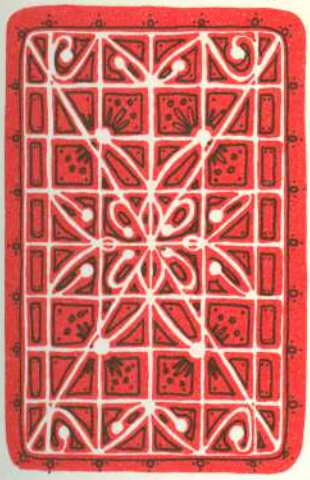
możemy rozpatrywać (nieskończony) zbiór *słów* nad tym alfabetem, to znaczy zbiór wszystkich skończonych ciągów utworzonych ze znaków a_j należących do A. Zbiór słów nad alfabetem A oznaczymy przez A^*. Wprowadzając do alfabetu A dowolne uporządkowanie, np. takie że

$$a_i < a_j \equiv i \leqslant j,$$

możemy już jednoznacznie uporządkować zbiór A^*. Słowa o tej samej ilości znaków porządkujemy w sposób leksykograficzny i umawiamy się, że słowo krótsze jest zawsze wcześniejsze od dłuższego. Każde słowo nad A ma wówczas dokładnie określony numer $k > 0$, a każdy taki numer wyznacza dokładnie jedno słowo nad A. Nadając słowu pustemu numer 0, otrzymujemy wzajemnie jednoznaczną relację między ciągami znaków a liczbami naturalnymi. Działanie dowolnego algorytmu można więc również jednoznacznie przedstawić jako przekształcenie liczb naturalnych.
Z pojęciem algorytmu wiąże się cały szereg pojęć pochodnych. Należy do nich przede wszystkim pojęcie funkcji obliczalnej. Niech A, B będą alfabetami i niech

$$f\colon A^* \dot{\to} B^*$$

będzie funkcją częściową (tzn. określoną na ogół tylko dla niektórych słów nad A, nie koniecznie na całym zbiorze A^*). Funkcję tę nazywamy *obliczalną*, jeżeli istnieje taki algorytm, że dla każdego $s \in A^*$, dla którego funkcja f jest określona, wynikiem działania tego algorytmu jest $f(s)$.
Podobnie określamy pojęcie obliczalności dla funkcji arytmetycznych $f\colon N \dot{\to} N$, tzn. określonych w zbiorze liczb naturalnych i przybierających wartości naturalne. Z uwagi na możliwość wzajemnego „przekładu" klasa formalno-lingwistycznych funkcji obliczalnych wyznacza klasę arytmetycznych funkcji obliczalnych



i vice versa. Możemy więc uznać, że mamy w gruncie rzeczy jedno ogólne pojęcie obliczalności funkcji. Dotyczy to także funkcji wielu zmiennych, a to dzięki istnieniu efektywnych wzajemnie jednoznacznych odwzorowań postaci

$$N \times N \leftrightarrow N, \qquad N \times N \times N \leftrightarrow N$$

etc., pozwalających wzajemnie jednoznacznie odwzorować zbiory par, trójek, etc. liczb naturalnych na zbiór liczb naturalnych. Innym ważnym pojęciem jest pojęcie rozstrzygalności predykatów. Predykat to nazwa własności; zdanie orzekające, że: „przedmiot x ma własność P", zapisujemy w skrócie jako $P(x)$.

Predykat P o dziedzinie D nazywamy *rozstrzygalnym*, jeżeli istnieje algorytm, który dla każdego $x \in D$ pozwala rozstrzygnąć czy $P(x)$, czy nie $P(x)$.

O rozstrzygalnym predykacie możemy mówić, że wyraża on własność lub relację rozstrzygalną.

Mając zdefiniowane pojęcie rozstrzygalności predykatu możemy natychmiast zdefiniować pojęcie zbioru rozstrzygalnego. Mianowicie mówimy, że zbiór $X \subset D$ jest rozstrzygalny (względem D), jeżeli predykat P_X zdefiniowany przez

$$P_X(x) \overset{\text{df}}{\equiv} x \in X$$

jest rozstrzygalny. Równoważną definicję otrzymujemy korzystając wprost z pojęcia obliczalności: zbiór jest rozstrzygalny wtedy i tylko wtedy, gdy jego funkcja charakterystyczna (równa 1 dla elementów należących do zbioru i 0 dla pozostałych) jest obliczalna.

Pojęcie rozstrzygalności jest więc szczególnym przypadkiem pojęcia obliczalności, sprowadza się ono bowiem do obliczalności funkcji o wartościach zero-jedynkowych (logicznych, bądź arytmetycznych).

Wszystkie powyższe pojęcia mają charakter nieformalny. Jednakże wszystkie one mogą być uściślone. Oczywiście uściślone pojęcia zawsze różnią się od ich pierwowzorów intuicyjnych. Za triumf teorii algorytmów można jednakże uznać fakt, że dokonane na wiele różnych sposobów uściślenia okazują się formalnie wzajemnie równoważne. Takimi są np. pojęcia obliczalności w sensie Turinga, λ-definiowalności Churcha, μ-rekursywności Kleene'go, ogólnej rekursywności i wiele innych.

Algorytm jest zawsze algorytmem czegoś, przepisem na wykonywanie pewnego zadania, jak rozwiązywanie równania kwadratowego, tłumaczenie programu z języka FORTRAN na kody wewnętrzne maszyny cyfrowej, wykonywanie pewnych rysunków. W trakcie realizacji algorytmu następuje przekształcenie pewnej sytuacji wejściowej w wyjściową, czyli „danych" w „wyniki". W sytuacji, kiedy algorytm jest narzędziem działania, a nie obiektem studiów, sama struktura algorytmu jest mniej ważna niż wykonywane zadanie, czyli funkcja

dane → wyniki.

Każda taka funkcja określona przez wyniki realizacji pewnego algorytmu jest, zgodnie z podanymi wyżej definicjami, funkcją obliczalną.

Takie okoliczności powodują, że oprócz tego rygorystycznego pojęcia algorytmu, które omówiliśmy powyżej, celowe może być wprowadzanie pojęć opartych na złagodzonych wymaganiach co do zdefiniowania postępowania w każdej sytuacji, która może się zdarzyć w trakcie wykonywania algorytmu. W pewnych sytuacjach możliwe są bowiem różne warianty postępowania, które jednak różnią się w sposób nieistotny w tym sensie, że prowadzą do tego samego wyniku. Zdarza się to już w najprostszych działaniach arytmetycznych: np. instrukcja „oblicz sumę liczb a, b, c" jest całkiem jednoznaczna, jeśli chodzi o wynik, lecz z punktu widzenia działań elementarnych algorytmem nie jest, bo nie określa kolejności poszczególnych operacji dodawania. Dopuszczenie tego rodzaju swobody wyboru postępowania w szczegółach nie mających wpływu na wynik prowadzi do rozszerzenia pojęcia algorytmu. Posługując się tak rozszerzonym pojęciem algorytmu należy jednak pamiętać, że jest ono różne od pojęcia rygorystycznego, i że ta różnica czasem może okazać się ważna. Jeżeli przez maszynę rozumieć układ, który nie jest w stanie dokonać wyboru — ani w sposób przypadkowy, ani metodyczny — to maszyna taka może realizować algorytmy tylko w sensie rygorystycznym.

Natomiast pojęcia takie jak „zadanie algorytmizowalne" (tzn. mające algorytm wykonywania), „funkcja obliczalna" itp. nie ulegają zmianie przy takim rozszerzeniu pojęcia algorytmu. Każdy bowiem algorytm rygorystyczny jest algorytmem w sensie rozszerzonym, a każdy algorytm w sensie rozszerzonym można „przerobić" na algorytm rygorystyczny, realizujący tę samą funkcję.

Przepisy obliczeniowe, zawierające działania na dowolnych liczbach rzeczywistych, nie są algorytmami w sensie określonym wyżej. Ściśle rozumiane działania na dowolnych liczbach rzeczywistych nie spełniają bowiem warunku efektywności, gdyż nie są wykonalne w skończonej liczbie kroków. Rachunki przybliżone są oczywiście algorytmizowalne, ale wymaga to sprecyzowania sposobu traktowania przybliżeń. Po takim zabiegu przepis obliczeniowy staje się algorytmem i wówczas daje się reprezentować przez pewne operacje na liczbach naturalnych.