Paradoks Russella

W miejscowości M jest fryzjer, nazwijmy go *superfryzjerem*, który strzyże tych i tylko tych mieszkańców miejscowości, którzy nie strzygą siebie samych. Czy superfryzjer strzyże siebie samego? Chwila namysłu pokazuje, że obie możliwości są wykluczone: nie może on strzyc siebie samego, bo strzyże tylko tych, którzy siebie sami nie strzygą; gdyby zaś sam się nie strzygł, to musiałby się strzyc, bo strzyże wszystkich tych, którzy sami się nie strzygą. A zatem, superfryzjer nie może istnieć! Pokażemy jak z powyższego faktu otrzymać różne twierdzenia matematyczne, odpowiednio definiując mieszkańców miejscowości M oraz to, kto kogo strzyże. Część tych wyników była opisana w artykule Wojciecha Czerwińskiego (*Delta* 10/2016) poświęconym metodzie przekątniowej, bardzo blisko związanej z paradoksem Russella.

Paradoks Russella. Bertrand Russell używał historii o fryzjerze, by ilustrować następujący problem dotyczący podstaw matematyki, odkryty przez niego w 1901 roku, i mający wielki wpływ na rozwój tej dziedziny. Czym jest zbiór? Chcielibyśmy, by elementami zbiorów mogły być inne zbiory; na przykład, okrąg to zbiór punktów na płaszczyźnie, i można rozważać zbiór wszystkich tych zbiorów, które są na płaszczyźnie okręgami o promieniu 1. Ogólniej, można by oczekiwać, że dowolne określenie postaci „zbiór wszystkich zbiorów o własności P", gdzie P jest precyzyjnie określoną własnością zbiorów, definiuje pewien zbiór. Pokażemy, że tak być nie może. Mieszkańcami M niech będą wszystkie zbiory, i powiemy, że zbiór x strzyże zbiór y, jeżeli y należy do x. Brak superfryzjera oznacza tyle, że określenie „zbiór wszystkich zbiorów, które nie należą do siebie samych" nie może określać żadnego zbioru.

Twierdzenie Cantora. Paradoks Russella był zainspirowany dziesięć lat starszą metodą przekątniową Georga Cantora. Cantor zastanawiał się, które zbiory są *przeliczalne*, tzn. wszystkie ich elementy można wypisać w nieskończonym ciągu $x_1, x_2, x_3, \ldots$ Oczywiście, zbiór liczb naturalnych $\{1, 2, 3, \ldots\}$ jest przeliczalny, ale też zbiór liczb wymiernych w przedziale $(0, 1)$ też jest przeliczalny: $\frac{1}{2}, \frac{1}{3}, \frac{2}{3}, \frac{1}{4}, \frac{3}{4}, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, \frac{1}{6}, \frac{5}{6}, \frac{1}{7}, \frac{2}{7}, \frac{3}{7}, \frac{4}{7}, \frac{5}{7}, \frac{6}{7}, \frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8}, \ldots$, również zbiór wszystkich liczb wymiernych jest przeliczalny (ćwiczenie dla Czytelnika). Z kolei, zbiór wszystkich liczb rzeczywistych, nawet tych w przedziale $(0, 1)$, nie jest przeliczalny, co właśnie wykazał Cantor, i my teraz znowu wykażemy. Każda liczba rzeczywista z przedziału $(0, 1)$ jest określona przez nieskończony ciąg cyfr, np. $0{,}14451323\ldots$ (przy czym nie rozważamy zapisów, które od pewnego momentu mają same cyfry 9).

Niech $c_1, c_2, c_3, \ldots$ będzie ciągiem liczb rzeczywistych z przedziału $(0, 1)$. Zdefiniujmy liczbę $x \in (0, 1)$ następująco: jej n-ta cyfra to 1, jeśli n-ta cyfra liczby c_n jest równa 0, i 0 w przeciwnym przypadku. Wykażemy, że x nie pojawia się w ciągu: gdyby $x = c_k$ dla pewnego k, to k byłoby superfryzjerem w miejscowości M zamieszkałej przez liczby naturalne, w której m strzyże n wtedy, i tylko wtedy, gdy n-ta cyfra liczby c_m to 1. Sprzeczność.

Twierdzenie Turinga. Wybierzmy swój ulubiony język programowania, np. Java, C++, Pascal czy Python (w oryginalnym dowodzie z 1936 r. Alan Turing użył bardzo prostego „języka", zwanego dziś *maszyną Turinga*). *Kod źródłowy programu* jest to napis (używający znaków dostępnych na klawiaturze komputera), który jest zgodny ze składnią wybranego języka. Rozważmy programy, których kod źródłowy jest szczególnej postaci $(*)$: pierwszą wykonywaną instrukcją jest „wczytaj napis N wprowadzony przez użytkownika", potem następuje ciąg instrukcji bez interakcji z użytkownikiem, a ostatnią instrukcją programu jest wypisanie słowa „koniec". Z punktu widzenia nieśmiertelnego użytkownika, który uruchamia taki program i wprowadza napis N, są dwa możliwe scenariusze: albo program po pewnym czasie napisze „koniec" – mówimy wtedy, że program *akceptuje* napis N – albo nigdy nic nie napisze, tzn. „zawiesi się". Dla wielu programów, wynik działania na danym napisie N bardzo łatwo przewidzieć. Można by pokusić się o napisanie programu Q, który dostaje na wejściu kod źródłowy dowolnego programu P postaci $(*)$ oraz napis N, po czym napisze „wiesza się", jeżeli program P się wiesza dostawszy na wejściu N, oraz napisze „akceptuje" w przeciwnym przypadku. Twierdzenie Turinga, które teraz udowodnimy, mówi, że taki program Q nie może istnieć. Gdyby istniał, to skonstruowalibyśmy program F który, dostawszy kod dowolnego programu P, akceptuje go wtedy, i tylko wtedy, gdy program P uruchomiony na wejściu P się wiesza, co można stwierdzić, uruchamiając program Q na parze P, P. Wtedy F byłby superfryzjerem w miejscowości M zamieszkałej przez kody źródłowe postaci $(*)$, w której P strzyże K jeśli program P akceptuje K. Sprzeczność.

Twierdzenie Gödla o niezupełności. W uproszczeniu, ten wynik z 1931 r. mówi, że istnieje zdanie, którego ani nie da się dowieść, ani dowieść jego zaprzeczenia. Wywnioskujemy to z twierdzenia Turinga. Przez *zdanie* rozumiemy napis spełniający pewne proste wymagania składniowe. O *dowodach* zakładamy jedynie tyle, że istnieje program komputerowy, który dostawszy napis K oraz napis Z odpowiada w skończonym czasie „poprawny" bądź „niepoprawny", w zależności od tego, czy K jest poprawnym dowodem zdania Z. Dodatkowo, każde zdanie, które ma dowód, jest prawdziwe.

Przypuśćmy, że dla każdego zdania Z albo istnieje jego dowód, albo dowód zdania „nieprawda, że Z", oznaczanego $\neg Z$. Napiszemy program Q, który działa następująco. Rozważmy zdanie Z „program P akceptuje napis N" oraz jego negację $\neg Z$. Wczytaj na wejściu kod źródłowy programu P oraz napis N. Jeżeli Z ma dowód, to napisz „akceptuje", a jeżeli $\neg Z$ ma dowód, to napisz „wiesza się". Żeby stwierdzić, który przypadek zachodzi, program Q przeszukuje wszystkie (coraz dłuższe) napisy, i dla każdego z nich sprawdza, czy jest dowodem Z bądź $\neg Z$. Z naszych założeń wynika, że w skończonym czasie znajdzie dowód albo Z, albo $\neg Z$, i że opisany program Q poprawnie przewiduje zachowanie programu P na wejściu N, przecząc twierdzeniu Turinga. Musi więc istnieć zdanie, które nie ma dowodu, ani którego zaprzeczenie nie posiada dowodu. Z kolei, z twierdzenia Gödla o *pełności* z 1929 roku, takie zdanie nie jest ani prawdziwe, ani nieprawdziwe – jest *niezależne* od przyjętych aksjomatów. Więcej o tym można znaleźć w moim artykule w *Delcie* 1/2017.

Inne zastosowanie w informatyce. W *złożoności obliczeniowej* mierzy się, jak „skomplikowana" jest własność liczb naturalnych, tj. ile kroków musi wykonać program, by stwierdzić, czy dana liczba ma tę własność. Przykładowo, własność „liczba parzysta" jest mniej skomplikowana niż własność „liczba pierwsza". Metoda przekątniowa pozwala wszystko znacznie bardziej skomplikować.

Szymon TORUŃCZYK