



Spróbuj, Czytelniku, obejrzeć krzywą o równaniu

$$X^2 - Y^2 + Y^4 = 0.$$

Inne ciekawe rysunki można otrzymać rozpatrując równania postaci

$$(X + A_1 Y)^{n_1} \cdot (X + A_2 Y)^{n_2} \cdot \dots \cdot (X + A_k Y)^{n_k} + w(X, Y) = 0,$$

gdzie $w(X, Y)$ jest wielomianem jednorodnym stopnia większego niż $n_1 + n_2 + \dots + n_k$, tzn.

$$w(X, Y) = B_0 X^n + B_1 X^{n-1} Y + \dots + B_{n-1} X Y^{n-1} + B_n Y^n,$$

gdzie $n > n_1 + \dots + n_k$.

Tę właśnie prostą widzimy (w przybliżeniu) na rysunku. Wyznacza ona tak zwany punkt w nieskończoności albo punkt niewłaściwy krzywej K . Gdyby ekran ukazywał rozwiązania nierzeczywiste (zespolone), zobaczylibyśmy jeszcze cztery pozostałe punkty w nieskończoności, odpowiadające „prostym” $X = \epsilon^k \cdot Y$, gdzie $\epsilon = \cos \frac{2\pi}{10} + i \sin \frac{2\pi}{10}$, a $k = 1, 3, 7$ lub 9 (rys. 9).

Widzimy więc, że oglądając krzywą K w różnej skali możemy zbadać ją z lokalnego i z globalnego punktu widzenia.

Wróćmy do parametru C . Niech C będzie liczbą rzeczywistą różną od zera i niech K' oznacza krzywą powstałą z K przez C -krotne powiększenie. Mamy

$$(X, Y) \in K' \Leftrightarrow (X/C, Y/C) \in K,$$

a zatem krzywa K' opisana jest równaniem

$$10(X/C)^2(Y/C)(X/C - Y/C) + (X/C)^5 + (Y/C)^5 = 0.$$

Mnożąc obie strony przez C^5 mamy

$$10 X^2 Y (X - Y) \cdot C + X^5 + Y^5 = 0.$$

Język BASIC ma co prawda operację potęgowania, ale jest ona poprawna tylko dla dodatnich argumentów. Skomplikowałoby to program na tyle, że praktyczniej będzie w naszym przypadku napisać $X * X * X * X * X$ zamiast $X ** 5$ itd. Wyjaśnia to postać najważniejszej, 30 linii programu na rysunku 1.

Czego nie potrafi komputer

Mgr Jarosław DEMINET

Czy komputer potrafi wszystko?

Wiadomo, że obecnie wiele problemów przekracza możliwości współczesnych komputerów, np. gra w szachy (na poziomie mistrza świata) albo niezawodna prognoza pogody. Nie ma jednak podstaw do przypuszczeń, że tych akurat problemów nie da się rozwiązać kiedyś w przyszłości. Czy jednak istnieją problemy, których żaden komputer nigdy nie rozwiąże? Okazuje się, że tak. Niektóre z nich są zresztą bardzo proste. Oto przykład.

Wiadomo, że każdy niebanalny program komputerowy zawiera pętle, tzn. ciągi instrukcji wykonywane wielokrotnie. Kontynuowanie obliczeń w pętli przebiega do momentu spełnienia określonego warunku, np. w algorytmie szybkiego sortowania (*Delta* 9/1985) zewnętrzna pętla wykonywała się, dopóki $i < j$, a zatem kończyła się, gdy $i \geq j$. Zarówno i , jak i j były zmieniane wewnątrz pętli, przy czym i było zwiększane, a j zmniejszane. Można dowiedzieć, że po skończonej liczbie wykonania pętli rzeczywiście obie zmienne przyjmą takie wartości, że $i \geq j$. Co jednak byłoby, gdybyśmy pisząc algorytm zrobili jakiś błąd i gdyby dla pewnych danych wejściowych warunek $i \geq j$ nigdy nie był spełniony? Powiedzielibyśmy wówczas, że program się zapętlił — pewien jego fragment wykonywałby się nieskończenie długo i wymagałby przerwania z zewnątrz. Oczywiście sytuacja taka jest przykra i każdy chce jej uniknąć. Własność programu polegająca na tym, że program zawsze zakończy swoją pracę, nazywamy własnością stopu. Na ogół potrafimy udowodnić „ręcznie” własność stopu dla swojego programu. Może wobec tego można napisać program komputerowy, który potrafiłby automatycznie dowodzić własność stopu lub jej brak dla dowolnego programu? Okazuje się, że nie, a najprościej tego dowiedzieć przez sprowadzenie do sprzeczności.

Przypuśćmy, że mamy procedurę WŁASNOŚĆ—STOPU (X), dla której parametrem X jest tekst dowolnej procedury, zapisanej w jakimś języku programowania. Procedura daje w wyniku

wartość logiczną *prawda*, gdy procedura X ma własność stopu, zaś *falsz*, gdy tej własności nie posiada. Oczywiście dysponujemy tekstem procedury WŁASNOŚĆ—STOPU. Teraz możemy napisać bezparametrową procedurę:

ZAGADKA:

dopóki WŁASNOŚĆ—STOPU (ZAGADKA) wykonuj
początek
 nic nie rób
koniec

Tekst tej prostej procedury przekazujemy jako parametr procedurze WŁASNOŚĆ—STOPU. Jaki będzie wynik?

Załóżmy, że WŁASNOŚĆ—STOPU (ZAGADKA) jest prawdą. Ale to znaczy, że warunek pętli w procedurze ZAGADKA będzie zawsze spełniony, a zatem pętla będzie się wykonywać nieskończenie i ZAGADKA nie ma własności stopu.

Załóżmy, że WŁASNOŚĆ—STOPU (ZAGADKA) jest fałszem. To znaczy, że pętla w procedurze ZAGADKA nie wykona się ani razu i procedura od razu zakończy się, a zatem ma ona własność stopu.

W obu przypadkach dochodzimy do sprzeczności. Oznacza to, że procedura WŁASNOŚĆ—STOPU o szukanych własnościach nie istnieje. Sprzeczność zniknie, gdy założymy, że procedura WŁASNOŚĆ—STOPU sama nie ma własności stopu, tzn. że dla niektórych wartości swojego parametru (dla niektórych procedur) nie daje w ogóle odpowiedzi w skończonym czasie. Ale to już nie to...

A więc nigdy nie da się zaprogramować komputera tak, aby określał, czy dowolny program zakończy swoją pracę. Mówimy, że własność stopu *nie jest rozstrzygalna*. Okazuje się zresztą, że wiele innych własności programów nie jest rozstrzygalnych. Nie istnieje np. algorytm znajdujący zbędne instrukcje i zmienne (zbędne — tzn. takie, które zawsze będą ominięte przez program). Można bowiem dowiedzieć, że gdyby taki algorytm istniał, to pozwoliłby on również na dowodzenie własności stopu.

A dla programistów wynika z tego morał: żaden komputer nie sprawdzi naszych programów całkowicie i do końca. Musimy błysnąć intuicją i sami znajdować dowody własności stopu naszych programów.