Vyčíslitelnost a transvyčíslitelnost

Vilém Vychodil

3. dubna 2001

Abstrakt

Tato stať shrnuje mé zajímavé postřehy na téma vyčíslitelnost, teorie složitosti a možnost realizace algoritmického řešení problémů. Text je psán, zcela záměrně, neformálním způsobem. Neklade si za cíl jednotlivá tvrzení dokazovat, spíš ukazovat souvislosti mezi problematikami.

Pod pojmem „vyčíslitelnost“ si intuitivně představujeme *algoritmizovatelnost* daného problému. Intuitivní pojem algoritmu jakožto konečného počtu kroků po jejichž aplikaci je konečný vstup transformován na konečný výstup, sám o sobě neříká nic o tom, zda-li existují i problémy nealgoritmizovatelné.

Alonzo Church vyslovit tezi o realizaci programů na Turingově stroji. Turingův stroj má spoustu modifikací. Lze si jej představit jako soustavu sestávající z konečného *automatu* a *nekonečné pásky*. Tato páska má jeden konec (zde se typicky uvažuje značka „začátek pásky“) a je možné z ní číst a/nebo zapisovat pomocí *hlavy*. Přechodovou funkci automatu můžeme rozšířit následovně:

$$\delta : Q \times T \to Q \times T \times \{-1, 1\},$$

kde Q je množina stavů automatu, T je symbol na pásce na pozici hlavy, množina $\{-1, 1\}$ udává, zda-li se hlava při přechodu přesune doleva (hodnota -1) nebo doprava (hodnota 1). Přesuny hlavy si lze představovat jako inkrementy indexu absolutní pozice hlavy (počet předcházejících polí) na pásce.

Turingův stroj si také můžeme představit jako jednoduchý počítač, disponující jednoduchými instrukcemi pro přesun hlavy doleva a doprava, zápis symbolu na pásku a podmíněný skok do stavu $q \in Q$ při symbolu t na aktuální pozici hlavy. I když tyto definice nejsou úplně ekvivalentní, výpočetní síla obou strojů je stejná. K tomuto závěru lze jednoduše dojít právě z Churcovy teorie.

Church předpokládá, a zatím tomu vše nasvědčuje, že pro libovolný „výpočetní postup“, který lze nazvat algoritmem lze sestavit instanci Turingova stroje, která jej realizuje. Vstupem do algoritmu je zřejmě počáteční slovo na pásce $w \in T^*$, spolu se sestaveným automatem tvoří *počáteční konfiguraci* Turingova stroje. Jelikož je jedním je základních vlastností algoritmu i jeho konečnost, automat Turingova stroje má dva speciální stavy. Prvním stavem je *accept* (automat zde ukončuje činnost), druhý je *reject* (rovněž je činnost ukončena, tentokrát ale s chybou). Pokud automat skončí ve stavu *accept*, budeme slovo na pásce chápat jako výstup algoritmu. Tato teze je podpořena jednak tím, že všechny formalismy ekvivalentních s Turingovými stroji mají stejnou výpočetní sílu a jednak tím, že doposud nikdo nepřišel s „algoritmem“, který by na Turingově stroji realizovat nešel.

Pozornému čtěnáři jistě neunikla souvislost s teorií automatů. Jakou množinu jazyků je schopen generovat Turingův stroj? Na úvod podotkněme, že *akceptováním vstupní věty* chápeme zastavení automatu ve stavu *accept*. Pro Turingovy stroje definujeme dvě základní třídy jazyků. Jazyk $L \subseteq T^*$ je **rekursivně spočetný**, pokud existuje instance Turingova stroje, která akceptuje právě všechny věty tohoto jazyka. Tato definice je poněkud benevolentní k řetězcům, které nejsou větami jazyka. Nijak totiž nespecifikuje, zda-li analýza skončí odmítnutím řetězce (Turingův stroj se zastaví se stavu *reject*) nebo se nezastaví vůbec — Turingův stroj *cyklí*. Z tohoto důvodu se uvažuje druhá třída — třída **rekursivních** jazyků, která jsou generovány *úplnými Turingovými stroji*, tj. stroji, které se pro libovolný vstup zastaví. Tím je zajištěno, že automat ze vždy ocitne ve stavu *accept* nebo *reject*.

Obecně je známo, že volný monoid $(T^*, \cdot, \varepsilon)$ je spočetný. Jelikož uvažujeme konečnou abecedu $|T| = n$, všech řetězců délky i je právě n^i (variace s opakováním). Sjednocením spočetně mnoha konečných množin obdržíme spočetnou množinu (v tomto případě nekonečnou spočetnou množinu), tj. $|T^*| = \aleph_0$. Formální jazyk L je podmnožina $L \subseteq T^*$, tj. množina všech formálních jazyků je rovna systému 2^{T^*} s mohutností kontinua $\aleph$.

Gramatiky, jakožto prostředek k definici jazyků, jsou v podstatě velmi omezený nástroj. Uvažujmě jednoprvkovou abecedu $A = \{a\}$ a všechny jazyky nad A generované gramatikami typu 0. Jelikož se jednotlivé gramatiky liší v podstatě jen v množinách neterminálních symbolů a odvozovacích pravidel, můžeme je upravit tak, že jejich neterminální symboly budou symboly z pevně dané spočetné množiny N. Tato úprava je velmi jednoduchá, prostě

přejmenujeme neterminální symboly u jednotlivých gramatik tak, aby odpovídaly vždy některému symbolu z N. Nyní můžeme zapsat odvozovací pravidla každé gramatiky jako větu nad (nekonečnou) abecedou $N \cup \{\underline{,}, \rightarrow, \underline{\{}, \underline{\}}\}$ (symboly jsou zvýrazněny podtržením). Z toho plyne, že všechny gramatiky lze charakterizovat všemi větami některého jazyka nad takto definovanou nekonečnou abecedou. Takový jazyk je však spočetný, protože vět délky i je spočetně mnoho (ale ne konečně mnoho, protože N musíme z důvodu obecnosti chápat jako nekonečnou množinu), a sjednocení spočetně mnoha spočetných množin je spočetná množina.

To je významné zjištění. Jednak ukazuje, že gramatiky jsou velmi omezené, lze jimi popsat jen spočetně mnoho jazyků nad abecedou. Zároveň je patrné, že musejí existovat jazyky, které nejsou typu 0, a kterých je vlastně „většina". Toto tvrzení je o to významnější, pokud uvážíme, jaký má vztah množina rekursivně spočetných jazyků L_T k množině jazyků L_0. Je zřejmé, že problém analýzy jazyka typu 0 je algoritmizovatelný.

Pokud bychom si představili dvoupáskový Turingů stroj, který pracuje nedeterministicky, je možné sestavit automat turingova stroje tak, aby prováděl nedeterministicky odvození zdola — nahoru. Na první pásce stroje by zřejmě byla vstupní věta, stroj akceptuje řetězec právě když na druhé pásce bude jediný symbol S (startovní sumbol gramatiky) a automat skončí ve stavu *accept*. Uvědomme si, že nedeterministický Turingův stroj je ekvivalentní jeho deterministické variantě, protože simulace Turingova stroje je opět algoritmizovatelný problém[1]. Rovněž dvoupáskový stroj není těžké simulovat (nebo jej prostě transformovat na jednopáskový, jehož políčka se postupně střídají).

Předešlá úvaha vede ke tvrzení (které je rigorosně dokazatelné), že platí $L_0 \subseteq L_T$. Opačná inkluse platí rovněž, její důkaz však není již tak průhledný. V podstatě jde ale o to, že jsme schopni zachytit pohyb hlavy na pásce a její činnost a popsat ji gramatikou bez omezení. Z předešlého plyne důležitý závěr. Turingovy stroje generují jazyky třídy 0. Existují nealgoritmizovatelné problémy a je jich nespočetně mnoho. Vyčíslitelných problémů je jen „velmi málo".

Důkaz o existenci nealgoritmizovatelného prohlému je možné provést i přímo na Turingově stroji. Vychází z Cantorova diagonalizačního principu. Nejprve si představme, že automat Turingova stroje (nějak) zakódujeme pomocí binární soustavy. Jište by nebylo těžké vymyslet spoustu způsobů, kterak popsat přechodovou funkci pomocí jedniček a nul. Stejně tak můžeme zakódovat všechna slova, vlastně vytvoříme zástupné symboly pro prvky symbolů na pásce, se kterými Turingův stroj pracuje. Takto zkódovaný Turingův stroj se nazývá *universální Turingův stroj*. Nyní můžeme sestavit tabulku na jejích řádcích budou Turingovy stroje a sloupce budou řetězce na abecedou vstupních symbolů. Vnitřní pole tabulky obsahují 1 právě když je odpovídající řetězec akceptován příslušným strojem, 0 jinak.

Uvažujme jazyk L takový, že $w \in L$ právě když automat M_l neakceptuje l, tj. na diagonálním poli je symbol 0. Předpokládejme, že existuje automat M_x, který akceptuje jazyk L. Pokud je na diagonálním poli příslušném x hodnota 0, pak M_x neakceptuje x a tím pádem $x \notin L$, to je ale spor, protože L je jazyk právě všech řetězců s touto vlastností. Pokud je na diagonálním poli příslušném x hodnota 1, pak automat akceptuje tento řetězec a $x \in L$, to je ale spor, protože L obsahuje pouze řetězce, které nemají tuto vlastnost. Z toho je zřejmé, že existuje jazyk, který není rekursivně spočetný.

Jakkoliv to možná zní nepovzbudivě, existují problémy, které jsou sice vyčíslitelné, ale pro jejich rozsah nemá smysl se do jejich řešení pouštět. Pod pojmem „*zpracování N bitů*" budeme chápat přenos N bitů po jednom nebo několika kanálech výpočetního systému. Bremermann odvodil teoretickou fyzikální mez tohoto čísla N. Následující stať jsem doslovně převzal ze skripta Algebraická teorie systémů od Ivana Chajdy. Jedinou změnou je odstranění jednoho nadbytečného „s" ve jménu Heisenberg.

Informaci v počítači kódujeme pomocí některé fyzikální veličiny a to tak, že je zakódována jako energetická hladina definovaného typu energie v intervalu $(0, E)$, kde E je množství energie, potřebné (nebo možné) pro náš záměr. Předpokládejme, že energetické hladiny lze rozlišit s přesností ΔE, tedy interval $(0, E)$ lze rozdělit na

$$N = \frac{E}{\Delta E}$$

podintervalů, každému odpovídá množství energie ΔE. Odpovídá-li každé hladině právě jeden bit, lze pomocí energie E zakódovat právě N bitů. Abychom vyjádřili maximum informace, je tedy nutné co nejvíce zmenšit ΔE. To však má svou fyzikální mez, danou Heisenbergovým principem.

$$\Delta E \cdot \Delta t \ \geq \ h,$$

kde Δt je doba trvání (měření) množství energie ΔE a h je Planckova konstanta, tj. $h = 6.623 \cdot 10^{-27}\ erg \cdot s^{-1}$. Z předchozí nerovnosti dostaneme $N \leq E \cdot \Delta t \cdot h^{-1}$. Vyjádříme-li množství energie E v daném množství hmoty dle Einsteinova vztahu $E = m \cdot c^2$, kde $c = 3.10^{10}\ cm \cdot s^{-1} = 3 \cdot 10^8\ m \cdot s^{-1}$ je rychlost světla ve vakuu, dostaneme maximální hodnotu N,

$$N \ = \ m \cdot c^2 \cdot \Delta t \cdot h^{-1},$$

což je 1 g hmoty za 1 s dává $N = 1,36 \cdot 10^{47}$ bitů.

Předpokládejme nyní, že hypotetický počítač by měl k dispozici hmotnost celé zeměkoule, tj. asi $6 \cdot 10^{27}\ g$, a pracoval by v čase Δt rovném celé době existence Země,

[1] průměrný programátor by jej jistě zvládl napsát

tj. asi okolo 10^{10} let, což je přibližně $3.15 \cdot 10^{17}$ let, což je přibližně $3.15 \cdot 10^{17}$ s. Pak by mohl zpracovat maximálně $2.56 \cdot 10^{92}$ bitů, zaokrouhleno na celé řády tedy 10^{93} bitů. Toto číslo $N = 10^{93}$ se nazývá **Bremermannova mez**. Je-li pro zpracování úlohy zapotřebí více než 10^{93} bitů, pak je tato úloha fyzicky neřešitelná — *transvyčíslitelná*.

Toto číslo se zdá asi dost velké, je však třeba si uvědomit, že každý skutečný počítač nemůže mít hmotnost celé Země a pracovat miliardy let, dále nelze kódovat v hladinách s maximální teoretickou fyzikální rozlišitelností, tj. skutečný počítač může zpracovávat jen daleko menší počet bitů. Přesto lze ukázat, že i poměrně snadno představitelná úloha může tuto mez překročit.

Nechť systém S má n prvků, každý nabývá právě k^n stavů. Tedy všech podmnožin stavů systému S. je 2^{k^n}. Předpokládejme, že máme pomocí počítače klasifikovat systém z množiny všech systémů tohoto typu. Známe-li efektivní způsob vyhledávání, při kterém každý bit informace umožňuje rozdělit zbývající množinu variant, je nutné zpracovat $log_2 2^{k^n} = k^n$ bitů. Tato úloha bude transvyčíslitelnou pro $k^n \geq 10^{93}$, což nastane např. pro následující dvojice hodnot k, n

k	2	3	4	5	6	7	8	9
n	308	194	154	133	119	110	102	97

Např. v úloze rozpoznání obrazců budeme zpracovávat obraz, skládající se z $n = q \cdot q$ bodů, každý může nabývat k stavů (odstíny, barvy atd.). Tedy lze získat právě $k^n = k^{q \cdot q}$ různých obrazců. Bremermannova mez je překročena již při počtu bodů 18×18, nabývají-li body jen dvou hodnot (černá, bílá) nebo 10×10 bodů, nabývají-li 9 barev. Sítnice lidského oka obsahuje asi milion buněk citlivých na světlo, nabývá-li každá jen dvou stavů, pak lze na sítnici znázornit až $2^{1000000}$ obrazců, což je asi 10^{300000} bitů informace. Toto číslo je tak nepředstavitelně daleko za Bremermannovou mezí, že lidský mozek musí rozpoznávat obrazce nikoliv jednoduchým výčtem, ale musí využívat daleko významnějších funkcí intelektu.

Dalším případem, kdy lze velmi snadno překročit Bremermannovu mez je testování stavů mikročipů. Pokud každá součástka mikročipu může nabývat hodnoty 0 nebo 1, pak množina všech stavů je pro n součástem rovna 2^n. Bremermannova mez je v tomto případě dle předešlé tabulky překročena pro $n = 308$.