Týden 11

Přednáška-pondělí

Univerzální Turingův stroj

Algoritmus, kterým jsme prokazovali částečnou rozhodnutelnost problému zastavení (HP) byl tento: na zadaný stroj (program) M a vstup w spusť „interpret“, který provádí činnost (výpočet) M na vstupu w. Takový interpret je ovšem také program, tedy algoritmus, a šlo by jej proto realizovat (naprogramovat) ve formě konkrétního Turingova stroje U (viz Věta 7.3.).

Riceova věta

Uvědomili jsme si, že každá vlastnost V Turingových strojů (např. „stroj M má více než 100 stavů“, „výpočet stroje M je pro každý vstup konečný“, apod.) rozdělí množinu všech Turingových strojů na dvě disjunktní podmnožiny: jedna je množina strojů, které vlastnost V mají, a druhá je množina strojů, které vlastnost V nemají. Vlastnost V je *triviální*, jestliže je jedna z oněch dvou příslušných množin prázdná (tedy buď všechny stroje vlastnost V mají nebo ji nemá ani jeden). Vlastnost V je *netriviální*, jestliže ji alespoň jeden stroj má a alespoň jeden stroj nemá.

Důkladně jsme si promysleli, co to je *vstupně/výstupní vlastnost*, zkráceně též *I/O vlastnost*, Turingových strojů (či obecně „programů“).

Speciálně jsme si uvědomili, že vlastnost V není I/O vlastností právě tehdy, když existují dva stroje M_1, M_2, které mají stejnou I/O tabulku (tedy stejné vstupně/výstupní chování), ale jeden z nich vlastnost V má a druhý ji nemá.

Probrali jsme si pak vlastnosti v řešeném příkladu 7.1. a uvědomili si, které jsou I/O vlastnostmi. Speciálně jsme si všimli, že podle definice je každá triviální vlastnost I/O vlastností.

Pak jsme se zamysleli nad Riceovou větou:

> Každá netriviální vstupně/výstupní vlastnost programů je nerozhodnutelná.

(V přednášce pro hlubší zájemce ukážeme důkaz využitím věty o rekurzi.)

Složitost algoritmů

Připomněli jsme si, že složitostí algoritmů většinou nemyslíme složitost jejich struktury či obtížnost jejich návrhu, ale časovou (či paměťovou) náročnost jimi definovaných výpočtů.

Přirozeně jsme navrhli chápat

> *časovou složitost Turingova stroje* M jako funkci $T_M : \mathbb{N} \to \mathbb{N}$

tak, že $T_M(n)$ udává počet kroků výpočtu stroje M nad vstupem velikosti (tj. délky) n v *nejhorším případě* (worst-case complexity).

Poznámka. O časové složitosti má tedy rozumný smysl hovořit jen u strojů, jejichž (všechny) výpočty jsou konečné.

Pak jsme navrhli (rámcově) Turingův stroj M_1 přijímající (přechodem do stavu q_{accept}) právě ta slova v abecedě $\{0,1\}$, která jsou z jazyka $\{0^m1^m \mid m \geq 1\}$.

Jednalo se o standardní jednopáskový Turingův stroj (s jednostranně nekonečnou páskou).

Při analýze jeho časové složitosti jsme si připomněli

> asymptotickou notaci $f(n) \in O(g(n))$, $f(n) \in o(g(n))$, $f(n) \in \Theta(g(n))$ (včetně běžných funkcí, které se vyskytují při analýze složitosti algoritmů)

a přišli na to, že u našeho konkrétního stroje M_1 platí $T_{M_1}(n) \in \Theta(n^2)$.

Pak jsme se zamysleli a přišli na nápad, jak navrhnout (standardní) stroj M_2 řešící tentýž problém, pro nějž jsme odvodili $T_{M_2}(n) \in \Theta(n \log n)$.

Přitom jsme si uvědomili, proč při takové analýze nezáleží na základu logaritmu (který zde bereme jako 2, pokud není řečeno jinak).

Pak jsme si ještě všimli, že umíme navrhnout *dvoupáskový* (či jen dvouhlavý) Turingův stroj M_3, který řeší výše uvedený problém a pro nějž je $T_{M_3}(n) \in \Theta(n)$.

Jen letmo jsme zaznamenali jako fakt, že

> mezi rozumnými výpočetními modely existují (vzájemné) polynomiální simulace,

takže třída PTIME, tedy *třída problémů řešitelných polynomiálními algoritmy* (tedy algoritmy s časovou složitostí omezenou polynomy), je nezávislá na tom, ve kterém (rozumném) výpočetním modelu (tedy ve kterém „programovacím jazyku“) algoritmy realizujeme.

Dynamické programování

Uvažovali jsme o problému nejdelší společné podposloupnosti ze sekce 10.2.4. Nejprve jsme celkem přímočaře sestrojili rekurzivní proceduru $LCS(u, v)$. Analýzou jsme zjistili, že počet volání LCS při řešení instance u, v, kde $|u| = |v| = n$, může být větší než 2^n. Navržený algoritmus tedy jistě není polynomiální.

Kladli jsme si otázku, zda se nedá navrhnout polynomiální algoritmus řešící uvedený problém. Zlepšení nás napadlo, když jsme si uvědomili, že při rekurzivním řešení se mnohé podpřípady mohou zbytečně řešit vícekrát (nezávisle na sobě). Přitom každý podpřípad stačí vyřešit jednou a řešení poznamenat do vhodné „tabulky“ (v našem případě dvourozměrného pole). Přitom postupujeme od menších podpřípadů k větším. To je princip tzv. metody *dynamického programování*. Výsledný algoritmus (také ilustrovaný jednou z animací) už polynomiální očividně je.

Metoda „Rozděl a panuj“

Připomněli jsme si i tuto obecnou metodu návrhu (efektivních) algoritmů. Naznačili jsme si odvození řešení rekurentních rovnic užitečných při analýze složitosti rekurzivních algoritmů, jak je to probráno v části 10.2.2.

Přednáška-čtvrtek

Dvě aplikace věty o rekurzi

1. Důkaz Riceovy věty

Uvažujme (libovolnou) netriviální vlastnost V Turingových strojů, která je rozhodnutelná. Existuje tedy Turingův stroj D, který pro každé $u \in \{0,1\}^*$ určí, zda M_u má či nemá vlastnost V. Vezměme teď nějaký konkrétní M_1, který vlastnost V má a nějaký konkrétní M_2, který V nemá (takové stroje existují, protože V je netriviální).

Uvažujme stroj K, který pracuje následovně:

na vstupní $u \in \{0,1\}^*$ spustí D; když D zjistí, že M_u má vlastnost V, vydá K jako výstup $f_K(u) = Kod(M_2)$, a když D zjistí, že M_u nemá vlastnost V, vydá K jako výstup $f_K(u) = Kod(M_1)$.

Podle věty o rekurzi musí existovat nějaké u tak, že I/O tabulka strojů $M_u, M_{f_K(u)}$ musí být stejná; přitom z návrhu K je zřejmé, že jeden ze strojů $M_u, M_{f_K(u)}$ vlastnost V má a druhý ji nemá. Vlastnost V tedy není vstupně/výstupní (neboli I/O) vlastnost.

Ukázali jsme tak, že každá netriviální rozhodnutelná vlastnost V Turingových strojů není I/O vlastností. Jinými slovy: každá netriviální I/O vlastnost Turingových strojů je nerozhodnutelná.

2. Nerozhodnutelnost minimality Turingova stroje

Při dohodnutém kódování Turingových strojů (slovy v abecedě $\{0,1\}$) můžeme kódy strojů přirozeně porovnávat podle délky a v rámci stejné délky abecedně (kde $0 < 1$).

Řekneme, že *Turingův stroj* M je *minimální*, jestliže neexistuje Turingův stroj M', který je ekvivalentní s M (tj. má stejné I/O chování, tj. stejnou I/O tabulku, neboli $f_M = f_{M'}$) a má menší kód než M.

Všimněme si, že minimálních strojů je nutně nekonečně mnoho.

Uvažujme problém

NÁZEV: Min-TM

VSTUP: Turingův stroj M.

OTÁZKA: Je M minimální?

Ukážeme, že

Min-TM je nerozhodnutelný.

(Ve skutečnosti není problém ani částečně rozhodnutelný, ale spokojíme se teď jen s důkazem nerozhodnutelnosti. Všimněme si také, že vlastnost minimality není I/O vlastností, a proto nerozhodnutelnost neplyne z Riceovy věty.)

Zase na to půjdeme sporem. Předpokládejme tedy, že existuje stroj D, který pro každé $u \in \{0,1\}^*$ zjistí, zda M_u je/není minimální.

Pak můžeme sestrojit stroj K, který se chová následovně:

pro zadané u systematicky generuje slova, která jsou větší než u; každé takové slovo po jeho vygenerování nechá K prověřit strojem D a tento proces generování skončí, když narazí na slovo u', pro něž $M_{u'}$ je minimální. (K takovému u' musí při generování nutně dospět, protože minimálních strojů je nekonečně mnoho.) Příslušné u' je vydáno jako výstup stroje K.

Podle věty o rekurzi existuje u takové, že pro strojem K vydané $f_K(u) = u'$ platí, že M_u a $M_{u'}$ mají stejnou I/O tabulku (tedy jsou ekvivalentní). Přitom $u = Kod(M_u)$ je menší než $u' = Kod(M_{u'})$ a stroj $M_{u'}$ je minimální – spor.

Nerozhodnutelnost aritmetiky (se sčítáním a násobením)

Uvedli jsme si jen hlavní myšlenku nerozhodnutelnosti pravdivosti uzavřených formulí 1.řádu s predikáty $PLUS(x,y,z)$ (tj. $x + y = z$) a $MULT(x,y,z)$ (tj. $x \cdot y = z$) ve standardním modelu aritmetiky $(\mathbb{N}, +, \cdot)$.

(Využili jsme nerozhodnutelnosti existence akceptujícího výpočtu zadaného Turingova stroje M na zadaném slově w.)

Rozhodnutelnost Presburgerovy aritmetiky (jen sčítání)

Nastínili jsme důkaz rozhodnutelnosti Presburgerovy aritmetiky (využívající jen predikát $PLUS(x,y,z)$), tedy důkaz věty 9.3. z kapitoly 9.

Partie textu k prostudování

Univerzální Turingův stroj, Riceova věta (v části 7.). Složitost algoritmů (8.1., 8.2.). Dynamické programování (10.2.4.), část 10.2.2. (metoda „rozděl a panuj“).

Cvičení

Prezentace referátů

Referát č. 19

V definici modelu RAM v základním studijním materiálu je uvedena hodnota operandu $*i$ jako číslo uložené na adrese, jež je dána součtem čísla i a čísla uloženého v indexregistru.

Jiná užívaná možnost nepřímé adresace je, že hodnota $*i$ je chápána jako číslo uložené na adrese, která je uložena v buňce s adresou i.

Ukažte, jak lze RAM-program v jedné variantě simulovat RAM-programem v druhé variantě a naopak. (Ilustrujte na jednoduchém příkladu.)

Referát č. 20 (Rozhodnutelnost a nerozhodnutelnost)
Uvažujme problém

NÁZEV: UHP (*Uniform Halting Problem*)

VSTUP: Turingův stroj M.

OTÁZKA: Zastaví se M na každý vstup?

Zjistěte, zda tento problém je rozhodnutelný či nerozhodnutelný a své zjištění prokažte. (Můžete např. vyjít ze stručné zmínky v textu
`http://www.cs.vsb.cz/jancar/TEORET-INF/teoret-inf.pdf`.)

Příklady

(Případné dokončení dřívějších příkladů, či alespoň jejich stručná diskuse.)

Příklad 11.1

Vysvětlete, co dělá univerzální Turingův stroj U, když dostane jako vstup slovo tvaru uv, kde slovo u je kódem stroje U.

Příklad 11.2

Uveďte alespoň tři vlastnosti Turingových strojů, pro něž plyne nerozhodnutelnost z Riceovy věty, a alespoň tři vlastnosti, pro něž nerozhodnutelnost z Riceovy věty neplyne.

Příklad 11.3

Připomeňte si, jak se používá a co vyjadřuje značení O, o, Θ.
Pak seřaďte následující tři funkce podle rychlosti jejich růstu.
a/ $n/2005$, $\sqrt{n} \cdot 3n$, $n + n \cdot \log n$
b/ $(\log n)^n$, n^n, $2^{\sqrt{n}}$

Příklad 11.4

Nechť $a, b > 1$. Ukažte:

- $\exists c : \forall x : \log_a x = c \cdot \log_b x$ (tedy $\log_a n \in \Theta(\log_b n)$)
- $a^{\log_b n} = n^{\log_b a}$ (návod: aplikujte na obě strany funkci $\log_b$)

Příklad 11.5

Věta o řešení rekurentních rovnic se dá úvest i takto (mírně obecněji než je v učebním textu):
Nechť $a \geq 1$, $b > 1$ jsou konstanty, f je funkce (typu $\mathbb{N} \to \mathbb{N}$, či alespoň asymptoticky kladná) a pro funkci $T : \mathbb{N} \to \mathbb{N}$ platí rekurentní vztah

$$T(n) = aT(n/b) + f(n).$$

Pak platí:

1. Je-li $f(n) = O(n^c)$ a $c < \log_b a$, pak $T(n) = \Theta(n^{\log_b a})$.
2. Je-li $f(n) = \Theta(n^{\log_b a})$, pak $T(n) = \Theta(n^{\log_b a} \cdot \log n)$.
   Obecněji: je-li $f(n) = \Theta(n^{\log_b a} \log^k n)$, $k \geq 0$, pak $T(n) = \Theta(n^{\log_b a} \cdot \log^{k+1} n)$.
3. Je-li $f(n) = \Omega(n^c)$ a $c > \log_b a$
   a $a \cdot f(n/b) \leq d \cdot f(n)$ pro nějaké $d < 1$ a skoro všechna n (tedy až na konečně mnoho výjimek),
   pak $T(n) = \Theta(f(n))$.

Ve 3. případu lze vyvodit, že když $f(n) = \Theta(n^c)$, tak $T(n) = \Theta(n^c)$. (Můžete ověřit.)
Pak zjistěte u následujících příkladů, zda se na ně věta vztahuje, a v kladném případě odvoďte řešení.

- $T(n) = 3T(n/2) + n^2$
- $T(n) = 4T(n/2) + n^2$
- $T(n) = 8T(n/2) + n^2$
- $T(n) = 7T(n/2) + n^2$
- $T(n) = 2T(n/2) + n \log n$
- $T(n) = T(n/2) + 2^n$
- $T(n) = 2^n T(n/2) + n$