Týden 10

Přednáška-pondělí

V pondělí přednáška nebyla (velikonoce).
Zde jen připomeneme pojmy z přednášek v předchozím týdnu, které se minulý týden neobjevily v písemných podkladech.

Převeditelnost mezi problémy

Důkladně jsme si ujasnili (i využitím tabulek problémů), co to znamená, když se řekne, že

> problém P_1 je *algoritmicky převeditelný* (stručněji *převeditelný*) na problém P_2; označujeme $P_1 \rightsquigarrow P_2$.

Ilustrovali jsme si na případu DHP$\rightsquigarrow$HP. Vyvodili jsme, že HP je také nerozhodnutelný (využitím tvrzení 6.11. v části 6.5.).

Částečná rozhodnutelnost, Postova věta

Definovali jsme *částečnou rozhodnutelnost* problémů. Přitom byla podstatná následující definice, kterou zde formulujeme v řeči „tabulek“:

> Turingův stroj M *částečně rozhoduje* problém P (typu ANO/NE), jestliže u každého vstupu, pro nějž je v T_P ANO, je v tabulce T_M výstup ANO a pro každý vstup, pro nějž je v T_P NE, je v T_M výstup NE nebo znak $\perp$ (nedefinováno).

(Pro vstupy, kterým problém P přiřazuje odpověď NE, nemusí stroj M svůj výpočet skončit.)
A samozřejmě: problém je částečně rozhodnutelný, jestliže existuje algoritmus (Turingův stroj), který jej částečně rozhoduje.
Speciálně jsme si uvědomili Postovu větu (Věta 7.2.)
Uvědomili jsme si také, že problém HP je částečně rozhodnutelný (viz Univerzální TS), a že tedy $\overline{\text{HP}}$ (doplňkový problém k problému HP) není (ani) částečně rozhodnutelný.

Cvičení

Prezentace referátů

Referát č. 17 (Simulace mezi různými variantami Turingových strojů 2)
Představte si Turingův stroj pracující na “čtverečkované rovině” (místo lineární pásky). Vstupní slovo je zapsáno na začátku v jednom řádku, čtecí hlava stojí na jeho začátku (ostatní buňky=čverečky obsahují prázdný znak). Obor hodnot přechodové funkce je nyní rozšířen tak, že možné pohyby hlavy jsou Left, Right, Up, Down.

Stručně a srozumitelně popište, jak je možné simulovat tento "rovinný" stroj klasickým "lineárním" strojem. (Nápověda. Musíte tedy popsat, jak bude mít lineární stroj uložen na pásce obsah oné roviny; stačí mít v každém okamžiku zachycen jen obdélník obsahující všechna políčka roviny, která simulovaný stroj dosud navštívil. Pak musíte popsat, jak bude simulující stroj provádět analogii konkrétních instrukcí simulovaného.)

Referát č. 18 (Další otázka týkající se Turingových strojů)
Představme si, že navrhujeme Turingův stroj M, přičemž chceme používat již hotový M_1 jako proceduru, které lze předat vstup (parametr), tj. řetězec symbolů, a obdržet od ní příslušný výstup.
K tomu se hodí chápat M jako dvoupáskový (víme, že vícepáskový stroj lze simulovat jednopáskovým, je-li potřeba). Když M potřebuje výstup M_1 odpovídající vstupu w, neboli potřebuje zjistit řetězec $M_1(w)$, napíše w na druhou pásku, na ní pak nechá běžet M_1 a až M_1 skončí, překopíruje si M výsledný řetězec z druhé pásky na „základní" pásku a udělá si s ním, co potřebuje.
Popište teď, jak byste řešili případ, kdy M potřebuje takto „volat" stroje $M_1, M_2, \ldots, M_k$, které se ovšem mohou také *rekurzivně volat navzájem.*

Příklady

Příklad 10.1

Zjistěte, co dělají dva níže uvedené fragmenty programů pro stroje RAM. Připomeňme, že paměťová buňka s adresou 0 je pracovní registr a buňka s adresou 1 je indexregistr. Hodnota operandu $*i$ (i je zápis celého čísla, např. 281) je číslo uložené v buňce s adresou $i + j$, kde j je aktuální obsah indexregistru. Oproti základní definici zde užíváme také symbolické názvy paměťových buněk (vyhrazených pro příslušné proměnné) a symbolická návěští.

```
        READ
        STORE   N
        LOAD    =2
cykl:   STORE   temp
        LOAD    N
        JGTZ    body
        LOAD    temp
        WRITE
        HALT
body:   SUB     =1
        STORE   N
        LOAD    temp
        MUL     temp
        JUMP    cykl
```

```
        LOAD    N
        STORE   1
zmena   LOAD    *A
        STORE   X
cykl    LOAD    1
        SUB     =1
        JZERO   konec
        STORE   1
        LOAD    *A
        SUB     X
        JGTZ    zmena
        JUMP    cykl
konec   HALT
```

Příklad 10.2

Navrhněte Turingův stroj M s jednostranně nekonečnou páskou, který pro dané vstupní slovo $w \in \{a,b\}^*$ sestrojí (výstupní slovo) $w(w)^R$. Stroj M tedy realizuje příslušné (vstupně/výstupní) zobrazení $f_M : \{a,b\}^* \to \{a,b\}^*$ (např. $f_M(abb) = abbbba$).
Pak zvolte vhodné kódování slov v abecedě $\{a,b\}$ tak, aby analogické vstupně/výstupní zobrazení mohl realizovat RAM. Navrhněte konkrétní RAM M', který toto zobrazení realizuje; přitom postupujte tak, že RAM M' přímočaře simuluje Turingův stroj M. (Nejde o co nejjednodušší RAM pro daný úkol, ale o to, abyste aplikovali obecný postup prokazující, že každý TS je možné simulovat RAMem.)

Příklad 10.3

Vysvětlete, co to je doplňkový problém k problému P (typu ANO/NE). Pak konkrétně definujte doplňkový problém Non-Eq-CFG k problému Eq-CFG (ekvivalence bezkontextových gramatik).

Příklad 10.4

Vysvětlete podrobně, co to znamená, když řekneme, že HP$\rightsquigarrow$Non-Eq-CFG (tj. že Halting Problem je převeditelný na problém Non-Eq-CFG). (Převeditelnost HP$\rightsquigarrow$Non-Eq-CFG zde nedokazujeme, ale dále ji bereme jako fakt.)

Příklad 10.5

Je možné, že oba problémy Eq-CFG a Non-Eq-CFG jsou částečně rozhodnutelné? Umíte prokázat částečnou rozhodnutelnost alespoň jednoho z nich?

Přednáška-čtvrtek

Problém zastavení
Podrobně jsme si probrali důkaz tvrzení, že „diagonální problém zastavení“ (Diagonal Halting Problem, DHP) není algoritmicky rozhodnutelný. Použili jsme důkaz sporem — tak, jak je uvedeno ve studijním textu (část 6.5.).
Věta o rekurzi
Omezíme se na Turingovy stroje M, které definují zobrazení f_M typu

$$\{0,1\}^* \to \{0,1\}^* \cup \{\perp\},$$

jak bylo diskutováno výše.
Připomeňme si, že každý Turingův stroj M lze přirozeně zakódovat slovem $KOD(M) \in \{0,1\}^*$. I když zobrazení KOD nemusí být surjektivní (tedy *na* množinu $\{0,1\}^*$), je technicky užitečné, že

> každý řetězec $u \in \{0,1\}^*$ lze chápat jako kód nějakého Turingova stroje, označeného M_u.

(Pro $u = KOD(N)$ je $M_u = N$; když $u \neq KOD(N)$ pro žádný stroj N, tak jako M_u bereme např. nějaký fixní stroj M', pro nějž je $\forall w \in \{0,1\}^* : f_{M'}(w) = \perp$.)

> *Věta o rekurzi.*
> Mějme TS K, pro nějž je f_K totální, tedy $f_K : \{0,1\}^* \to \{0,1\}^*$. Pak existuje nějaký řetězec $u \in \{0,1\}^*$ takový, že $f_{M_u} = f_{M_{f_K(u)}}$.

Na stroj K lze pohlížet jako na „transformátor programů (tj. Turingových strojů)“. Ke každému stroji (resp. jeho kódu) vydá K nějaký (obecně jiný) stroj (resp.jeho kód). Věta o rekurzi tedy říká, že pro každý takový „transformátor strojů“ existuje stroj, který má stejnou I/O tabulku jako stroj vzniklý jeho transformací pomocí K.

Důkaz.

Promysleme si toto:

Když dostaneme $x \in \{0,1\}^*$, umíme jistě zkonstruovat stroj, označme jej $N(x)$, který předepisuje tento výpočet:

- Nejprve (se ignoruje vstup y a vpravo od něj) je simulován stroj M_x na vstupu x (s výhodou lze použít univerzální TS jako podprocedura),
- v případě, že výpočet M_x na x skončí a vydá nějaké $w \in \{0,1\}^*$, tak se spustí K na w, čímž se vypočte $f_K(w)$,
- simuluje se $M_{f_K(w)}$ na (původním) vstupu y,
- když $M_{f_K(w)}$ na y skončí, je zanechán na pásce jen jeho výstup a výpočet skončí.

Při hlubším promyšlení ovšem vidíme, že postupujeme algoritmicky, tedy, že vyrobení (kódu) stroje $N(x)$ k zadanému x můžeme naprogramovat ...

Takže vlastně umíme sestrojit TS N, který k libovolnému vstupu $x \in \{0,1\}^*$ zkonstruuje kód stroje $N(x)$. (Tedy stroj $N(x)$ je $M_{f_N(x)}$; všimněmě si také, že f_N je totální funkce.)

Podívejme se na řetězec $u = f_N(KOD(N))$, což je kód stroje M_u. Prozkoumejme, co dělá stroj M_u, když má na vstupu y:

- Nejprve (ignoruje svůj vstup y a vpravo od něj) simuluje stroj $M_{KOD(N)} = N$ na vstupu $KOD(N)$,
- výpočet N na $KOD(N)$ určitě skončí a vydá $f_N(KOD(N)) = u$; spustí se K na u, čímž se vypočte $f_K(u)$,
- simuluje se $M_{f_K(u)}$ na (původním) vstupu y,
- když $M_{f_K(u)}$ na y skončí, zanechá se na pásce jen jeho výstup a výpočet skončí.

Čili M_u má očividně stejné I/O chování (stejnou I/O tabulku) jako $M_{f_K(u)}$.

———-

Podívejme se na následující drobnou aplikaci věty o rekurzi:
Existuje program (Turingův stroj), který ignoruje (smaže) vstup a vypíše svůj vlastní kód.
(Vezměme K, který ke každému u vyrobí kód stroje, jehož činností je „smaž vstup a vypiš u“. Musí tedy existovat konkrétní u, pro nějž $f_{M_u} = f_{M_{f_K(u)}}$, tedy M_u a $M_{f_K(u)}$ mají stejnou I/O tabulku. Jelikož $M_{f_K(u)}$ provádí činnost „smaž vstup a vypiš u“, tak pro jakýkoli vstup vydá výstup u. Stroj M_u tedy také pro jakýkoli vstup vydá výstup u, neboli vypíše svůj kód.)