Týden 10

Přednáška

Rozhodnutelnost a nerozhodnutelnost problémů

Nejprve jsme si důkladně připomněli, co to je (v našem kontextu) *problém*. Uvědomili jsme si, že každému problému P je jednoznačně přiřazena (většinou nekonečná) „*tabulka*“ T_P s dvěma sloupci: v prvním sloupci jsou v jednotlivých řádcích vyjmenovány všechny (přípustné) vstupy (neboli instance) problému P

> (vstupy jsou zadány vhodně zvolenými kódy = slovy v určité abecedě [de facto stačí abeceda {0,1}] a jsou např. seřazeny podle délky a v rámci stejné délky lexikograficky])

a v druhém sloupci jsou uvedeny příslušné výstupy

> (v i-tém řádku je tedy v 1. sloupci i-tý vstup w a v 2. sloupci je příslušný výstup $p(w)$, kde p je zobrazení $p : IN \to OUT$ určené problémem P; tabulka T_P je prostě reprezentací zobrazení p).

V případě ANO/NE-problému se v druhém sloupci vyskytují jen výstupy Ano a Ne.

> *Poznámka.* S ohledem na budoucí úvahy si speciálně uvědomme, že pro každý vstup v tabulce T_P je definován příslušný výstup; v druhém sloupci se tedy nikde neobjeví „nedefinováno“ (znak $\perp$).

Detailně jsme si pak připomněli problém zdvojení slov v abecedě $\{0, 1\}$ (v tabulce je u každého slova u v 1. sloupci slovo uu v 2. sloupci), problém ekvivalence konečných automatů (Eq-FA), problém ekvivalence bezkontextových gramatik (Eq-CFG), problém zastavení Turingova stroje (HP, halting problem), diagonální problém zastavení (DHP).
Pak jsme si uvědomili, že každému *algoritmu* A odpovídá také jistá (vstupně/výstupní) tabulka T_A, která každému možnému vstupu w algoritmu A přiřazuje

- výstup vydaný algoritmem A – v případě, že běh algoritmu A pro vstup w je konečný,
- nebo speciální znak $\perp$ (nedefinováno) – v případě, že běh algoritmu A pro vstup w je nekonečný.

Jak víme, můžeme si (díky Churchově-Turingově tezi) pod pojmem algoritmus představit (např.) Turingův stroj. Každý Turingův stroj M tedy určuje příslušnou tabulku T_M. Můžeme jí říkat např.

vstupně/výstupní tabulka, zkráceně *I/O-tabulka*, stroje M.

Je to prostě (nekonečná) reprezentace vstupně/výstupního chování stroje M, tedy reprezentace příslušného I/O zobrazení $f_M : \Sigma^* \to \Gamma^* \cup \{\perp\}$, kde Σ je vstupní a Γ (celková) pásková abeceda stroje M.

> *Poznámka.* Víme, že se bez ztráty obecnosti můžeme omezit na stroje s abecedami $\Sigma = \{0,1\}$, $\Gamma = \{0,1,\square\}$, jejichž I/O zobrazení je typu $\{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ (což mj. znamená, že stroje jsou navrženy tak, aby při ukončení výpočtu zůstal na pásce jediný souvislý úsek nul a jedniček [nepřerušovaný znaky $\square$]).

Pomocí uvedených pojmů tabulky problému a I/O-tabulky algoritmu (Turingova stroje) jsme vyjádřili, kdy je problém P *algoritmicky řešitelný*; v případě ANO/NE-problému hovoříme o *algoritmické rozhodnutelnosti*, či zkráceně jen *rozhodnutelnosti*.

Jen nám tedy naprosto jasné, co je myšleno, když se řekne „problém Eq-FA je rozhodnutelný“ a „problémy Eq-CFG, HP, DHP jsou nerozhodnutelné“.

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.

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.).

Sekce pro hlubší zájemce – důkazy

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.)

Partie textu k prostudování

Rozhodnutelnost a nerozhodnutelnost problémů, převeditelnost mezi problémy (6.5.). Částečná rozhodnutelnost, Postova věta, univerzální Turingův stroj (7.).

Cvičení

Cvičení 1.5. se nekoná (státní svátek).