Provozně ekonomická fakulta

Teoretická informatika
Tomáš Foltýnek
foltynek@pef.mendelu.cz

Rozhodnutelnost

Opakování z minulé přednášky

- Co je to Turingův stroj?
- Jak je Turingův stroj formálně definován?
- Co je to konfigurace, krok výpočtu a výpočet TS?
- Jaký je rozdíl mezi jazykem akceptovaným a rozhodovaným TS?
- Co je to rekursivní a rekursivně spočetný jazyk?
- Jak lze použít TS pro výpočet hodnot funkcí?
- Co je to rozhodnutelný, částečně rozhodnutelný a nerozhodnutelný problém?
- Jaké varianty TS znáte?
- Jaké jsou uzávěrové vlastnosti rekurzivních a rekurzivně spočetných jazyků?

Otázky pro dnešní přednášku

- Existují jazyky (problémy), které nejsou rekursivně spočetné (částečně rozhodnutelné)?
- Existují jazyky (problémy), které jsou rekursivně spočetné (částečně rozhodnutelné), ale nejsou rekursivní (rozhodnutelné)?
- Pokud ano, které to jsou?

Osnova přednášky

- Formalizace pojmu algoritmus
  - Churchova teze
  - Další formalizace algoritmu
  - Kódování strojů a slov
  - Univerzální TS
- Problém příslušnosti
  - Diagonalizace
- Problém zastavení
  - Redukce
- Důsledky nerozhodnutelnosti

Churchova teze

- Algoritmus = postup řešení určitého problému
  - Obecný
  - Konečný
  - Deterministický
  - Srozumitelný
  - Opakovatelný
- Jak pojem algoritmus formalizovat?
- Church-Turingova teze: **„Každý proces, který lze intuitivně nazvat algoritmem, lze ralizovat pomocí Turingova stroje“**
- Ztotožnění pojmů „algoritmicky řešitelný“ a „řešitelný pomocí TS“

Další formalizace algoritmu

- Postovy systémy
- Minského stroje
- μ-rekursivní funkce
- λ-kalkul
- while programy

- Všechny jsou výpočetně ekvivalentní Turingově stroji

Kódování Turingova stroje

- Je dán TS $M = (Q, \Sigma, \Gamma, \triangleright, \perp, \delta, q_0, q_A, q_R)$.
- Předpokládejme seřazení a očíslování prvků množin Q, Σ, Γ takové, že $Q = \{q_0, q_A, q_R, \ldots\}$, $\Gamma = \{\triangleright, \perp, \ldots\}$
- Dále označme $s_1 = L$ a $s_2 = R$
- Pak hodnotu přechodové funkce $\delta(q_i,x_j) = (q_k,x_l,s_m)$ lze jednoznačně zakódovat binárním řetězcem $0^i10^j10^k10^l10^m$
- Binárním kódem TS M je pak řetězec 111<KÓD1>11 <KÓD2>11…11 <KÓDr>111
- Analogicky lze kódvat i vstupní slova stroje M
- Kód stroje M označíme <M>, kód slova w označíme <w>

Univerzální Turingův stroj

- Díky kódování TS a vstupního slova lze sestrojit Univerzální TS U takový, že
- L(U) = {<M>#<w> | M akceptuje slovo w}
- Výpočet stroje U:
  - Ověří, že je slovo požadovaného tvaru (pokud ne, tak zamítá)
  - Simuluje krok po kroku výpočet stroje M nad slovem w.
  - U má 3 pásky: kód stroje M, aktuální obsah pásky stroje M, aktuální stav stroje M
  - U akceptuje (zamítá) právě tehdy, když M nad w akceptuje (zamítá). Pokud M nad w cyklí, pak i U nad <M>#<w> cyklí.

Problém příslušnosti pro TS

- Problém určit, zda stroj M slovo w akceptuje, nebo neakceptuje (tj. zamítne, nebo bude cyklit).
- Tedy problém určit, zda $w \in L(M)$
- Definujeme jazyk odpovídající problému:
  PP = {<M>#<w> | stroj M akceptuje slovo w}
- Tedy PP = {<M>#<w> | $w \in L(M)$}

Rozhodnutelnost PP

- Snadno lze ověřit, že PP je částečně rozhodnutelný
  - L(U) = PP
  - U není úplný, proto zatím nevíme, zda je PP rozhodnutelný
    - Může existovat i jiná metoda, než je UTS?
- Odpověď na otázku za chvíli…

Diagonalizace I.

- Existuje jazyk, který není rekursivně spočetný
- Každý řetězec nad abecedou {0,1} lze chápat jako kód nějakého TS, resp. jako kód nějakého slova.
- Pro dané slovo $x \in \{0,1\}^*$ označme M_x Turingův stroj s kódem x a w_x slovo s kódem x.
- Všechna slova nad abecedou lze snadno uspořádat a sestrojit dvourozměrnou tabulku

Diagonalizace II.

	w_ε	w_0	w_1	w_{00}	w_{01}	w_{10}	...
M_ε	**1**	0	1	1	0	1	
M_0	0	**1**	1	1	0	1	
M_1	1	1	**1**	0	1	1	...
M_{00}	0	0	0	**0**	0	1	
M_{01}	1	0	0	1	**0**	0	
M_{10}	0	0	0	1	1	**0**	
...			...				...

Diagonalizace III.

- Zkonstruujeme jazyk D nad abecedou {0,1}.
- Zařadíme do něj právě taková slova w_d, která nejsou akceptována strojem M_d.
- Sestrojili jsme jazyk, který není akceptován žádným Turingovým strojem.
- Kdyby byl akceptován nějakým strojem M_m, pak by nesměl obsahovat slovo w_m právě tehdy, když jej obsahuje, a naopak.

Důkaz ne-rozhodnutelnosti PP

- Sporem. Předpokládejme existenci úplného TS T akceptujícího jazyk PP
- Pro vstup <M>#<w> stroj T pracuje takto:
  - T zastaví a akceptuje právě tehdy, když M zastaví a akceptuje
  - T zastaví a zamítne právě tehdy, když M zastaví a zamítne, anebo když cyklí na w.
- Zkonstruujeme stroj N, který pro vstup x pracuje takto:
  - Na pásku zapíše řetěz x#x
  - Simuluje výpočet stroje T na vstupu x#x
  - N akceptuje právě tehdy, když T zamítne vstup x#x
  - N zamítne právě tehdy, když T akceptuje vstup x#x
- Stroj N nad vstupem <N> pak nechá stroj T simulovat sebe sama a zamítne právě tehdy, když akceptuje, a akceptuje právě tehdy, když tamítne
- Dostáváme tedy spor, tudíž stroj T nemůže existovat
- Jazyk PP tedy není rekursivní

Důsledek ne-rozhodnutelnosti PP

- Jazyk co-PP = {<M>#<w> | M neakceptuje w} není rekursivně spočetný
- PP je rekursivně spočetný. Kdyby byl i co-PP rekursivně spočetný, pak by byl PP i co-PP rekursivní
- Víme, že PP není rekursivní, tudíž co-PP nemůže být ani rekurisvně spočetný
- Neexistuje tedy (ani neúplný) TS akceptující jazyk co-PP

Problém zastavení I.

- Problém rozhodnout, zda daný TS M nad daným vstupem w zastaví, není rozhodnutelný
- Jazyk PZ = {<M>#<w> | výpočet M na w je konečný} není rekursivní
- Důkaz provedeme sporem. Předpokládáme rozhodnutelnost PZ, tedy existenci TS T, L(T) = PZ
- Ukážeme, že pak by existoval úplný TS akceptující PP
- Ten však existovat nemůže, tudíž nemůže existovat ani T.

Problém zastavení II.

- Sestrojíme stroj N akceptující jazyk PP. Stroj bude pracovat takto:
- Simuluje výpočet stroje T (PZ) na vstupu <M>#<w>
- Pokud T akceptuje, pak to znamená, že výpočet M nad w je konečný. N může tedy simulovat výpočet M na w
- Pokud T zamítne, pak to znamená, že M nad w cyklí. N tedy vstup zamítne.
- Protože PP není rekursivní, stroj N nemůže existovat. Nemůže tedy existovat stroj T, na němž je výpočet stroje N založen.

Redukce

- Problém příslušnosti jsme převedli (redukovali) na problém zastavení.
- Kdybychom měli k dispozici algoritmus PZ (stroj T), pak bychom měli i algoritmus PP (stroj N).
- Víme-li že neexistuje algoritmus pro PP (stroj N), pak nemůže existovat ani algoritmus PZ (stroj T)

Formální definice redukce

- Redukce jazyka $A \subseteq \Sigma^*$ na jazyk $B \subseteq \Psi^*$ (značíme $A \leq B$) je rekursivní funkce $r: \Sigma^* \rightarrow \Psi^*$ taková, že $w \in A \Leftrightarrow r(w) \in B$
- Jedná se tedy o funkci realizovatelnou TS (rekursivní fce) zachovávající příslušnost do jazyka
- Zobrazení, které všechna slova z jazyka A zobrazí na slova z jazyka B a všechna slova napatřící do jazyka A zobrazí na slova nepatřící do jazyka B.

Využití redukce

- Nechť $A \leq B$. Pak platí:
  - Není-li jazyk A rekursivně spočetný, pak není ani jazyk B rekursivně spočetný
  - Není-li jazyk A rekursivní, pak není ani jazyk B rekursivní
  - Je-li jazyk B rekursivně spočetný, pak je i jazyk A rekursivně spočetný
  - Je-li jazyk B rekursivní, pak je i jazyk A rekursivní

Důsledky pro informatiku

- Existují problémy, které nelze řešit pomocí počítače
  - Problém zastavení
  - Postův korespondenční problém
  - Problém rozhodnout, zda jazyk generovaný danou CFG je konečný
  - Problém rozhodnout, zda dvě CFG generují stejný jazyk
  - Každá netriviální vlastnost rekursivně spočetných jazyků

Rozhodnutelné problémy

- Je-li problém rozhodnutelný, ještě to neznamená, že je rozhodnutelný „v rozumném čase“.
- Za rozumný čas považujeme takový čas, kdy je pro nás výsledek výpočtu ještě využitelný.
- Rozhodnutelností se zabývá teorie vyčíslitelnosti
- Časovou (a prostorovou) náročností se zabývá teorie složitosti
  - Můžete se těšit na příští přednášku