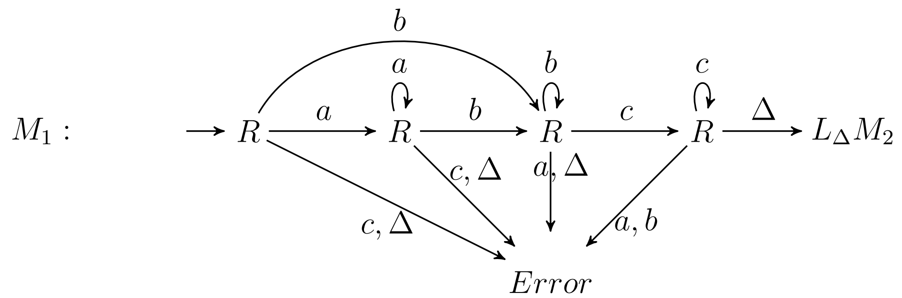
Úkol č. 1 do předmětu Složitost

Vojtěch Havlena (xhavle03)

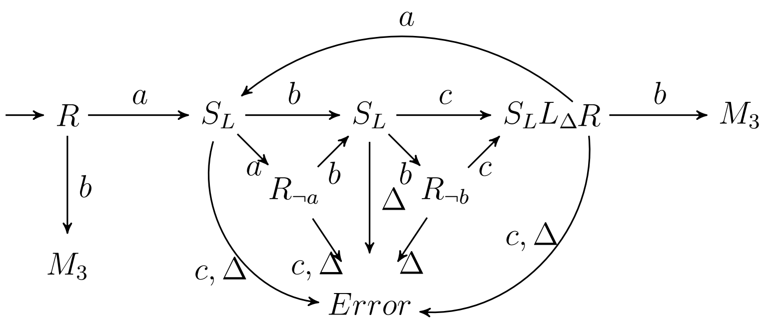
1. **Příklad** Výsledný TS, který rozhoduje jazyk $L = \{a^i b^j c^k | i < j < k\}$ (předpokládám, že $i, j, k \in \mathbb{N}_0$) jsem rozdělil do několika částí, které potom mohou být analyzovány samostatně. První částí je dílčí TS M_1 (který je zároveň výsledným TS, ale prozatím neuvažujme předání řízení stroji M_2 úplně na konci). Tento TS (M_1) provádí kontrolu, zda řetězec na vstupu je ve tvaru $a^i b^j c^k$, kde $i \in \mathbb{N}_0, j, k \in \mathbb{N}$. Tato kontrola se provádí postupným projitím vstupního řetězce. Je možné, aby řetězec na vstupu neobsahoval prefix tvořený symboly a, nicméně vzhledem k tomu, jak je zadaný jazyk L, v každém případě musí obsahovat sufix $b^j c^k$, kde $j, k > 0$.



Druhou částí je dílčí TS M_2 (opět neuvažujme předání řízení stroji M_3). Tento stroj postupně po jednom odstraňuje (pomocí posuvu vlevo, S_L) symboly a, b a c, dokud zbývá nějaký symbol a. V případě, že úplně na začátku první symbol je b, předáme řízení stroji M_3 (řetězec neobsahuje žádný symbol a). Pokud odstraníme symbol a a neexistuje další symbol b nebo c, který bychom mohli odstranit, řetězec odmítneme. Stejně tak odmítneme, pokud s odstraněním posledního symbolu a, zbydou na pásce pouze symboly c nebo pouze Δ (počet některých symbolů je stejný).

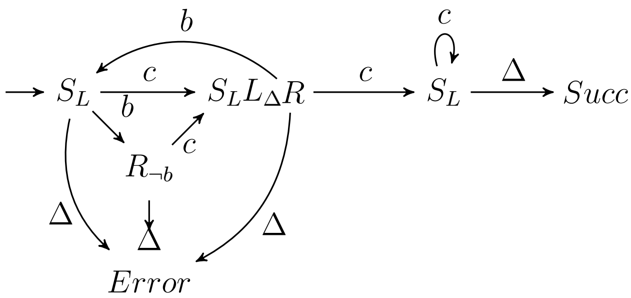
Více detailně: TS M_2, pokud je první vstupní symbol a, tento symbol odstraní posuvem užitečného obsahu pásky doleva. Dále se posunuje doprava, dokud nenarazí na jiný znak než a. V případě, že další znak je c nebo Δ, je řetězec odmítnut. Když narazíme na symbol b, opět pomocí posuvu už. obsahu pásky vlevo smažeme symbol b a opět přejdeme zbylé symboly b (pokud dalším symbolem je Δ a ne c, odmítneme). V případě, že narazíme na symbol c, odstraníme i tento symbo posuvem obsahu vlevo a posuneme se vpravo na první pozici. Pokud je první symbol modifikovaného řetězce a, celý tento postup opakujeme. Pokud prvním symbolem je c, Δ, počet nekterých symbolů je stejný, proto odmítneme. Jinak jsme odstranili všechny symboly a a předáme řízení stroji M_3.

M_2 :



V případě, když už modifikovaný řetězec neobsahuje žádné symboly a, řízení se předá stroji M_3. Tento stroj postupně odstraňuje symboly b a c posuvem obsahu pásky vlevo. Opět se kontroluje, zda je počet symbolů b větší než počet symbolů c. V případě, kdy se odstraní všechny symboly b a zbyly nějaké symboly c, tyto symboly se odstraní a řetězec je přijat.

M_3 :



Poznámky: Nepřijetí řetězce je v diagramu pro jednoduchost označeno jako $Error$, což může označovat posouvání hlavy doleva až do abnormálního zastavení. Dále v diagramu se označuje S_L jako posuv užitečného obsahu pásky, který se vyskytuje napravo od aktuální pozice hlavy, o jednu pozici vlevo. V případě, že se vpravo od aktuálního symbolu vyskytuje symbol Δ, je aktuální symbol přepsán na Δ.

Horní odhad časové složitosti Vzhledem k tomu, jak je výsledný TS rozdělen na dílčí části, časová složitost vzhledem ke vstupu w je dána jako

$$t_M(w) = t_{M_1}(w) + t_{M_2}(w) + t_{M_3}(w).$$

Opět uvažujme že jednotlivé dílčí stroje neobsahují předání řízení dalšímu stroji M_i a výsledný TS M je dán kompozicí jednotlivých strojů: $M :\rightarrow M_1 M_2 M_3$. Tedy

$$T_M(n) = \max\{t_{M_1}(w) + t_{M_2}(w) + t_{M_3}(w) | w \in \Sigma^n\}.$$

Uvažujme libovolný řetězec w délky nejvýše n. Potom $t_{M_1}(w) \leq 2n + 2$ – vstupní řetězec se celý projde a pak se provede návrat na 1. pozici pásky pomocí L_Δ. Tedy

$$t_{M_1}(w) \in O(n)$$

Složitost posuvu obsahu pásky vlevo S_L je $O(n)$ – postupně čteme symboly vlevo a zapisujeme je na pozici napravo, tedy v nejhorším případě projdeme celý řetězec (n) krát nějaká kladná konstanta kvůli posunování se na předchozí pozici a zapisování posouvaného symbolu. Složitost jednoho průchodu stroje M_2 (bez zpětné hrany a) je tedy $3t_{S_L} + t_{R_{\neg a}} + t_{R_{\neg b}} + t_{L_\Delta} + c$, kde $c \in \mathbb{N}$. Do konstanty c jsou schovány jednotlivé posuvy, např. $R, Error$. A tento jeden průchod se provede maximálně i-krát (počet symbolů a). Vzhledem k tomu, že stroj M_1 provede kontrolu tvaru vstupního řetězce, můžeme dále uvažovat vstupní řetězec ve tvaru $a^i b^j c^k$. Tedy v nejhorším případě máme

$$t_{M_2}(w) \leq i(3t_{S_L} + t_{R_{\neg a}} + t_{R_{\neg b}} + t_{L_\Delta} + c) \leq n(3t_{S_L} + 2n + c),$$

což spolu s horním odhadem S_L a tím, že $t_{R_{\neg a}} + t_{R_{\neg b}} \leq n$ a $t_{L_\Delta} \leq n + c$, dává $t_{M_2}(w) \in O(n^2)$.

Podobně složitost v nejhorším případě jednoho průchodu první částí stroje M_3 (bez zpětné hrany b) je $2t_{S_L} + t_{R_{\neg b}} + t_{L_\Delta} + c$ a tento průchod se provede celkem $(j - i)$-krát (v případě vykonávání stroje M_3 už víme, že $j > i$ a v případě vykonávání poslední části stroje M_2 víme také, že $k > j$). Tedy společně s druhou částí stroje M_3 dostáváme

$$t_{M_3}(w) \leq (j - i)(2t_{S_L} + t_{R_{\neg b}} + t_{L_\Delta} + c) + (k - j)t_{S_L} \leq n(3t_{S_L} + 2n + c).$$

Tím pádem opět se složitostí S_L dostáváme $t_{M_3}(w) \in O(n^2)$. Celková časová složitost v nejhorším případě je tedy $T_M(n) \in O(n^2)$.

Horní odhad prostorové složitosti Vzhledem k tomu, jak je Turingův stroj navržen se využívá pouze ta část pásky, kde je zapsán vstup. Symboly se pouze odstraňují posunem obsahu pásky vlevo. Hlava se dostane nejdál na pozici 1 za vstupním řetězcem w, tj. $n + 1$, kde $n = |w|$. Tedy celkový horní odhad prostorové složitosti je $S_M(n) \in O(n)$.

2. Příklad Výsledný RAM program je možné rozdělit do dvou částí. První část programu, $sqr(n)$, počítá druhou mocninu čísla n. Druhá část s využitím sqr počítá druhou odmocninu. První verze programu zmenšuje postupně číslo n a hledá největší i takové, že $i^2 \leq n$. Tedy algoritmus v pseudokódu vypadá následovně.

Algoritmus 1: ZÁKLADNÍ SCHÉMA ALGORITMU

Vstup: Číslo n takové, že $n \geq 0$
Výstup: $\lfloor \sqrt{n} \rfloor$

$i \leftarrow n$
$j \leftarrow sqr(i)$
while $j > n$ **do**
  $i \leftarrow i - 1$
  $j \leftarrow sqr(i)$
end
return i

Nevýhoda tohoto algoritmu je, že se v cyklu opakovaně počítá druhá mocnina čísla i. Proto je možné využít toho, že $(i-1)^2 = i^2 - 2i + 1$. Novou hodnotu j je tedy možné spočítat z předchozí hodnoty. Druhá mocnina se tedy spočítá pouze jednou na začátku programu. Navíc násobení $2i$ lze nahradit za sčítání $i + i$. V cyklu se tedy používají pouze operace $+, -$. Zápis výsledného algoritmu v pseudokódu je následující. Na řádcích 5 – 8 se postupným přičítáním čísla i počítá druhá mocnina čísla i. Pro ušetření počtu registrů se při výpočtu druhé mocniny využívají proměnné (registry), které se potom dále používají pro výpočet druhé odmocniny.

Algoritmus 2: VÝSLEDNÝ ALGORITMUS

Vstup: Číslo n takové, že $n \geq 0$
Výstup: $\lfloor \sqrt{n} \rfloor$

1: $i \leftarrow n$
2: $k \leftarrow 0$
3: $l \leftarrow 0$
4: $j \leftarrow 0$
5: **while** $k < i$ **do**
6:     $j \leftarrow j + i$
7:     $k \leftarrow k + 1$
8: **end**
9: **while** $j > n$ **do**
10:     $k \leftarrow i$
11:     $i \leftarrow i - 1$
12:     $l \leftarrow k + k$
13:     $j \leftarrow j - l$
14:     $j \leftarrow j + 1 \ldots (j \leftarrow j - 2k + 1)$
15: **end**
16: **return** i

Pro výpočet jsou tedy potřeba registry: i_1 (vstup), r_0 (akumulátor), r_1 (proměnná i), r_2 (proměnná k), r_3 (proměnná j), r_4 (proměnná l).

```
READ 1

.sqrstart
LOAD 1
SUB 2
JZERO .cstart
JNEG .cstart
LOAD 3
ADD 1
STORE 3
LOAD 2
ADD =1
STORE 2
JUMP .sqrstart

.cstart
READ 1
SUB 3
JZERO .halt
JNEG .halt
LOAD 1
STORE 2
SUB =1
STORE 1
LOAD 2
ADD 0
STORE 4
LOAD 3
SUB 4
ADD =1
STORE 3
JUMP .cstart
.halt
LOAD 1
HALT
```

Uniformní časová a prostorová složitost Předpokládejme vstupní vektor $I = (n)$. Kód v pseudokódu na řádcích 5 – 8 (v RAM programu to odpovídá řádkům 4 – 14) se provede celkem n-krát (n-krát se přičte číslo n). Tedy složitost tohoto úseku je celkem $11n \in O(n)$. Kód v pseudokódu na řádcích 9 – 15 (v RAM programu 17 – 32) se provádí jednou pro každé $i \in \{n, \dots \lfloor\sqrt{n}\rfloor\}$. Cyklus se tedy bude provádět celkem $(n - \sqrt{n} + 1)$-krát. Celková složitost tohoto cyklu je tedy $16(n - \sqrt{n} + 1) \leq 16n \in O(n)$. Celkově tedy pro vstup $I = (n)$ máme

$$t_{\Pi}^{uni}(I) \leq 27n + c \in O(n),$$

kde $c \in \mathbb{N}$ a tato konstanta schovává jednotlivé kroky (které nejsou v cyklu) jako počáteční čtení vstupu, zastavení, ... Délka vstupu $I = (n)$ je $len(I) = \lceil \log_2(n) \rceil$. Horní odhad uniformní celkové časové složitosti T_{Π} tedy bude:

$$T_{\Pi}(k) \leq \max\left\{t_{\Pi}^{uni}(I) | len(I) = k\right\} = \max\left\{t_{\Pi}^{uni}(I) | \lceil \log_2(n) \rceil = k\right\} \in O(2^k).$$

(délka vstupu je $n \leq 2^k$, složitost $t_{\Pi}^{uni}(I)$ je $O(n)$, tudíž horní odhad $T_{\Pi}(k)$ je $O(2^k)$). Co se týče uniformní prostorové složitosti pro vstup $I = (n)$, $s_{\Pi}^{uni}(I) = 6 \in O(1)$ (během výpočtu se použije max. 6 registrů). Horní odhad uniformní celkové prostorové složitosti S_{Π} tedy bude

$$S_{\Pi}(k) = 6 \in O(1).$$

Logaritmická časová a prostorová složitost V případě logaritmické časové složitosti nejprve spočítáme horní odhad cenové funkce pro vstup $I = (n)$. Největší číslo se kterým se v programu pracuje je n^2 (předpokládejme, že $n^2 > c_0$, kde c_0 je maximální konstanta použitá v programu. Pokud by platilo $n^2 < c_0$, tak horní odhad cenové funkce bude právě tato konstanta, nicméně na asymptotické složitosti to nic nemění). Tedy pro každý krok programu k:

$$c_{(\Pi,I)}^{log}(k) \leq len(n^2) = \lceil \log_2(n^2) \rceil \leq 2\lceil \log_2(n) \rceil$$

Potom tedy horní odhad log. časové složitosti pro vstup $I = (n)$ je

$$t_{\Pi}^{log}(I) \leq t_{\Pi}^{uni}(I) c_{(\Pi,I)}^{log} \in O(n \log_2(n))$$

a horní odhad logaritmické celkové časové složitosti T_{Π} tedy bude

$$T_{\Pi}(k) \leq \max\left\{t_{\Pi}^{log}(I) | \lceil \log_2(n) \rceil = k\right\} \in O(2^k k).$$

Co se týče logaritmické prostorové složitosti, největší uložená hodnota pro vstup $I = (n)$ je n^2, tedy horní odhad logaritmické prostorové složitosti je (celkem je 6 registrů)

$$s_{\Pi}^{log}(I) \leq 6 \cdot 2\lceil \log_2(n) \rceil \in O(\log_2(n)).$$

Horní odhad logaritmické prost. složitosti tedy bude

$$S_{\Pi}(k) \leq \max\left\{s_{\Pi}^{log}(I) | \lceil \log_2(n) \rceil = k\right\} \in O(k).$$

3. Příklad

Nejprve důkaz toho, že $L \in NTIME(n)$. V příkladu uvažuji následující definici $NTIME$ (viz. TIN):

$$NTIME(f(n)) = \{L|\text{existuje k-páskový NTS } M : L = L(M) \text{ a } T_M \in O(f(n))\}$$

Nechť L je bezkontextový jazyk. Potom k němu existuje gramatika $G = (N, \Sigma, P, S)$ v Chomského normální formě taková, že $L(G) = L$. Sestavíme 3-páskový NTS M, který bude simulovat provádění pravidel v CNF z množiny P.

- První páska obsahuje vstupní řetězec w.
- Druhá páska slouží pro simulaci nejlevější derivace. V průběhu výpočtu obsahuje řetězec, který je celý tvořen symboly z Σ a poslední symbol je z N.
- Třetí páska slouží jako zásobník pro neterminály, které jsou součástí větné formy, ale nejsou nejlevější. Kompletní větnou formu lze sestavit konkatenací užitečného obsahu 2. pásky s užitečným obsahem 3. pásky v reverzním pořadí.

NTS M potom pracuje tak, že se nejprve na 2. pásku vloží symbol S a na pozici $n+1$ na 2. a 3. pásce se vloží symbol, který slouží jako okraj (nikdy nemůžeme mít delší větnou formu než n). Tento okrajový symbol nemůže být přepsán (pokud by měl být přepsán, NTS M abnormálně zastaví).

- Nechť A je neterminál na 2. pásce. Potom se opakovaně nedeterministicky zvolí pravidlo $A \rightarrow \alpha \in P$. Pokud $\alpha = BC$, $B, C \in N$, tak se aktuální neterminál na 2. pásce přepíše na B a do zásobníku na 3. pásce se vloží neterminál C. Pokud $\alpha = b$, $b \in \Sigma$, tak se přepíše symbol neterminálu na 2. pásce za b, hlava se posune o pozici doprava a zapíše se neterminál, který je na vrcholu zásobníku na 3. pásce (symbol, který je na pozici hlavy na 3. pásce). Na 3. pásce je potom tento symbol vymazán.
- NTS M nakonec zkontroluje, zda obsah 2. pásky je shodný s obsahem 1. pásky (tedy jestli byl vygenerován řetězec w). Pokud ano, zastaví přechodem do q_F. V opačném případě abnormálně zastaví.

Co se týče časové složitosti (počtu kroků NTS M), tak přepis symbolů, vkládání a výběr ze zásobníku lze provést v konstantním počtu kroků (omezeno nějakou kladnou konstantou k, nezávislé na velikosti vstupního řetězce). Pro kontrolu, zda je obsah 2. pásky shodný s obsahem 1. pásky je potřeba $O(n)$ kroků (toto se ale provede pouze jednou během výpočtu). Vzhledem k tomu, že gramatika je CNF, tak počet použitých pravidel p pro vygenerování věty o délky n je $p = 2n - 1 \in O(n)$. Navíc kvůli omezení na užitečnou délku 2. a 3. pásky NTS skončí i v případě odmítnutí s počtem kroků $O(n)$. Celkový počet kroků je tedy vzhledem k výše uvedenému $O(n)$ a tedy $L \in NTIME(n)$.

Vzhledem k tomu, že funkce $f(n) = n$ je časově i prostorově zkonstruovatelná, přičemž $f(n) \geq \log_2(n)$, platí následující vztah

$$NTIME(f(n)) \subseteq DSPACE(f(n)).$$

Tím pádem platí $L \in DSPACE(n)$.