

# Kapitola 6

## Zložitosť triedy

V tejto časti sa budeme zaoberať zložitosťou na TS. Bude nás zaujímať, aký vplyv má množstvo príslušnej miery zložitosti na triedu problémov, ktoré sa s danou zložitosťou dajú riešiť. Pritom sa sústredíme nielen na výsledky, ale aj na metódy, ktoré sa na ich riešenie používajú.

V časti 6.1 sa budeme venovať dolným odhadom; vysvetlíme použitie prechodovej postupnosti pri získaní dolného odhadu na čas a prechodovej matice pri dolnom odhade na priestor. Potom vysvetlíme dôvody pre zavedenie pojmu konštruovateľnej/počítateľnej funkcie a ukážeme, že existuje nekonečná hierarchia času(6.2.2) a priestoru(6.2.1). V časti 6.3 zavedieme nedeterministický TS a napokon v časti 6.4 sa budeme venovať vzťahu determinizmu a nedeterminizmu a hierarchii zložitosťných tried.

### 6.1 Niektoré metódy dolných odhadov

Najprv uvedieme použitie prechodovej postupnosti pre získanie dolného odhadu na čas pri rozpoznávaní jazyka na jednopáskovom TS. Pri dolných odhadoch na priestor sa zas využíva pojem prechodovej matice.

Zopakujme si:

$\sup_{n \rightarrow \infty} f(n)$  je limita najnižšej hornej hranice postupnosti  $f(n), f(n+1), \dots$

$\inf_{n \rightarrow \infty} f(n)$  je limita najvyššej dolnej hranice postupnosti  $f(n), f(n+1), \dots$

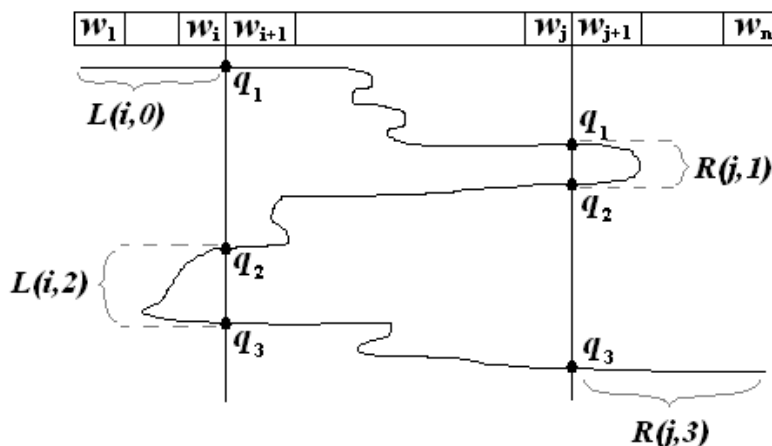
Tieto pojmy sa používajú v prípade, keď funkcia limitu nemá, napriek tomu  $\sup$ , resp.  $\inf$  existuje. Napr.

$$f(n) = \begin{cases} \frac{1}{n} & \text{pre } n \text{ párne;} \\ n & \text{pre } n \text{ nepárne.} \end{cases}$$

$\inf_{n \rightarrow \infty} f(n) = 0$ ,  $\sup_{n \rightarrow \infty} f(n) = \infty$  ale limita funkcie neexistuje.

#### 6.1.1 Prechodová postupnosť a dolný odhad na čas

Majme jednopáskový TS, ktorého jediná jednosmerne nekonečná páska je čítacia aj prepisovacia zároveň. Predpokladajme, že stroj najprv zmení stav a až potom pohne hlavou. Potom pre fixovaný výpočet a každé prirodzené  $i$  má zmysel uvažovať o postupnosti stavov, v ktorých stroj prechádza hranicu medzi políčkami  $i$  a  $i+1$ . Túto postupnosť stavov (pre fixovaný výpočet) nazveme prechodová postupnosť medzi  $i$ -tým a  $(i+1)$ -ým políčkami.



Obrázok 6.1: Prechodová postupnosť

**Lema 6.1** *Nech  $w = a_1a_2 \dots a_n$  je akceptované jednopáskovým TS  $T$  a nech prechodová postupnosť medzi  $i, i + 1$  je rovnaká ako medzi  $j, j + 1$  ( $i < j$ ). Potom  $T$  akceptuje aj slovo  $\omega = a_1a_2 \dots a_i a_{j+1} \dots a_n$ .*

**Dôkaz:** Bez ujmy na všeobecnosti predpokladáme, že stroj končí v konfigurácii s hlavou umiestnenou na poslednom symbole vstupu.

Majme uvažovaný akceptujúci výpočet  $A(w)$  na slove  $w = a_1a_2 \dots a_n$ . Nech uvažovaná prechodová postupnosť medzi  $i, i + 1$  (resp.  $j, j + 1$ ) je  $q_1, q_2, \dots, q_s$ .

Označme  $L(i, t)$  tú časť výpočtu  $A(w)$ , ktorá leží medzi príchodom sprava na políčko  $a_i$  v stave  $q_t$  a odchodom doprava z políčka  $a_i$  v stave  $q_{t+1}$ ,  $t$  párne ( $L(i, 0)$  je začiatok výpočtu  $A(w)$  do okamihu, keď stroj prvýkrát prekročí hranicu medzi  $i, i + 1$ ).

Nech  $R(i, t)$  označuje tú časť výpočtu, ktorá začína príchodom zľava na políčko  $a_{i+1}$  v stave  $q_t$  a odchodom z tohto políčka doľava v stave  $q_{t+1}$ ,  $t$  nepárne.

Potom akceptujúci výpočet na slove  $a_1a_2 \dots a_i a_{j+1} \dots a_n$  môže prebiehať nasledovne:

$$L(i, 0), R(j, 1), L(i, 2), R(j, 3), \dots$$

□

Analogicky sa dá dokázať nasledovné tvrdenie.

**Lema 6.2** *Nech  $v = a_1a_2 \dots a_n$ ,  $u = b_1b_2 \dots b_m$  sú akceptované jednopáskovým TS  $T$  a nech prechodová postupnosť medzi  $a_i, a_{i+1}$  je rovnaká ako medzi  $b_j, b_{j+1}$ . Potom  $T$  akceptuje aj slovo  $a_1a_2 \dots a_i b_{j+1} \dots b_m$ , resp.  $b_1b_2 \dots b_j a_{i+1} \dots a_n$ .*

Prechodová postupnosť akýmsi spôsobom zachytáva "komunikáciu" medzi dvomi časťami vstupného slova, resp. vstupných slov.

*použitie prechodovej postupnosti* **Cvičenie 6.1** *Ukážte, že žiadny jednopáskový TS rozhodujúci jazyk  $\{a^n b^n \mid n \geq 0\}$  nemá dĺžku prechodových postupností ohraničenú konštantou.*

Ako ale súvisia prechodové postupnosti s odhadom na čas?

**Fakt 6.3** Ak  $T$  je  $T(n)$  časovo ohraničený TS, potom súčet dĺžok prechodových postupností je najviac  $T(n)$ .

Je zrejmé, že ak chceme zdola odhadnúť čas, stačí zdola odhadnúť súčet dĺžok prechodových postupností.

**Veta 6.4** Ak je jazyk  $L = \{w_1 w_2 w_3^R | w \in \{0, 1\}^*\}$  rozpoznávaný jednopáskovým TS  $T$ , dolný odhad na čas pre jeho časovú zložitosť  $T(n)$  platí

$$\sup_{n \rightarrow \infty} \frac{T(n)}{n^2} > 0$$

**Dôkaz:** Na základe predchádzajúcich liem je zrejmé, že nemôžu existovať také slová  $w_1 w_2 c w_2^R w_1^R$ ,  $w_3 w_4 c w_4^R w_3^R$ , pre ktoré platí (\*):

- $|w_1| = |w_3|$
- $w_1 \neq w_3$
- v akceptujúcich výpočtoch na slovách  $w_1 w_2 c w_2^R w_1^R$ ,  $w_3 w_4 c w_4^R w_3^R$  je prechodová postupnosť medzi  $w_1, w_2$  rovnaká ako medzi  $w_3, w_4$ .

V opačnom prípade by totiž stroj akceptoval<sup>1</sup> aj slová  $w_1 w_4 c w_4^R w_3^R$ , resp.  $w_3 w_2 c w_2^R w_1^R$ , ktoré nie sú z jazyka.

Zafixujme dĺžku slova  $n$  a označme  $p(i)$  priemernú dĺžku prechodovej postupnosti medzi  $i$ -tým a  $(i + 1)$ -ým políčkom – pritom priemer uvažujeme cez všetky slová dĺžky  $n$  z jazyka. Potom platí, že aspoň polovica slov (dĺžky  $n$ ) z jazyka má dĺžku prechodovej postupnosti medzi  $i, i + 1$  nanaajvyš  $2p(i)$ . Týchto slov (označme ich  $W_n$ ) je aspoň

$$\frac{1}{2} 2^{\frac{(n-1)}{2}} = 2^{\frac{(n-1)}{2}-1} \quad (6.1)$$

Koľko je všetkých možných prechodových postupností dĺžky nanaajvyš  $2p(i)$ ?

$$\sum_{j=1}^{2p(i)} s^j \leq s^{2p(i)+1} \quad (6.2)$$

Ak teda spravíme rozklad množiny slov  $W_n$  podľa relácie - mať rovnakú prechodovú postupnosť medzi  $i, i + 1$  - vieme odhadnúť priemerný počet prvkov v jednej triede tohto rozkladu. Tento počet dostaneme ako počet slov z  $W_n$  (6.1) lomeno počet prechodových postupností dĺžky nanaajvyš  $2p(i)$  (6.2). Pritom tento počet nesmie byť väčší ako počet všetkých možných dokončení medzi  $i + 1$  a  $(n - 1)/2$ . V opačnom prípade by sme totiž v jednej triede mali dve slová, ktoré spĺňajú (\*).

Teda

$$\frac{2^{\frac{(n-1)}{2}-1}}{s^{2p(i)+1}} \leq 2^{\frac{(n-1)}{2}-i}$$

$$2^{i-1} \leq s^{2p(i)+1}$$

$$(i - 1) \log 2 \leq (2p(i) + 1) \log s \leq 3p(i) \log s$$

<sup>1</sup>vyplýva z Lemy

Pre priemernú dĺžku prechodovej postupnosti teda dostávame:

$$\frac{i-1}{3 \log s} \leq p(i)$$

Ak sčítame priemerné dĺžky, dostaneme odhad na priemerný čas. Keďže aspoň jeden výpočet musí mať zložitosť aspoň priemernú, poskytuje dolný odhad na priemerný čas aj dolný odhad na čas ako taký.

$$\begin{aligned} \sum_{i=1}^{(n-1)/2} \frac{i-1}{3 \log s} &= \frac{1}{3 \log s} \left( \sum_{i=1}^{(n-1)/2} i - \sum_{i=1}^{(n-1)/2} 1 \right) \\ &= \frac{1}{3 \log s} \left( \frac{(n-1)(n+1)}{8} - \frac{1}{2}(n-1) \right) = \frac{1}{24 \log s} (n^2 - 4n + 3) \end{aligned}$$

$$\sup_{n \rightarrow \infty} \frac{T(n)}{n^2} \geq \frac{1}{24 \log s} > 0$$

□

### 6.1.2 Prechodová matica a dolný odhad na pamäť

Na riešenie niektorých problémov pamäť, presnejšie pamäťovú pásku, nepotrebuje; vystačíme si s konštantnou pamäťou, ktorú nám poskytuje stav. Ako je to však s pamäťovou zložitosťou v prípade, keď nám konštantná pamäť nestačí?

*stav pamäte*

Pre TS s jednou vstupnou a jednou pamäťovou páskou je *stav pamäte* daný

- obsahom pamäťovej pásky
- polohou hlavy na pamäťovej páske
- stavom konečnosťovej riadiacej jednotky

Uvedomme si, že ak  $L(n)$  je pásková zložitosť TS, tak počet rôznych stavov pamäte je zhora ohraničený hodnotou

$t^{L(n)}L(n)s$ , kde  $t$  označuje mohutnosť páskovej abecedy  
 $s$  je počet stavov

Majme vstupné slovo dĺžky  $n$ . Nech  $r$  je počet stavov pamäte, ktoré stroj pri spracovaní slova dĺžky  $n$  môže dosiahnuť. Môžeme predpokladať, že jednotlivé stavy pamäte sú očíslované  $1, 2, \dots, r$ .

*prechodová matica*

*Prechodová matica* pre  $L(n)$  páskovo ohraničený TS je matica  $T_{r \times r}^n$ , ktorej prvky sú z množiny  $\{00, 01, 10, 11\}$ .

Nech  $w$  je vstupné slovo dĺžky  $n$ ,  $u$  jeho koncové podslovo (sufix),  $r = t^{L(n)}L(n)s$ . Hovoríme že *prechodová matica*  $T_{r \times r}^n$  *popisuje koncové podslovo*  $u$  ak platí

- ak TS začne čítať  $u$  v stave pamäti  $i$  a dosiahne stav pamäte  $j$  bez toho, aby opustil koncové podslovo  $u$ , tak  $T^n(i, j) = 1*$  (inak  $T^n(i, j) = 0*$ )<sup>2</sup>
- ak TS začne čítať  $u$  v stave pamäti  $i$  a prvýkrát opustí  $u$  doľava v stave pamäti  $j$ , tak  $T^n(i, j) = *1$  (inak  $T^n(i, j) = *0$ )

**Veta 6.5** *Nech TS je  $L(n)$  páskovo ohraničený TS s jednou vstupnou a jednou pamäťovou páskou. Nech  $w_1, w_2$  sú vstupné slová dĺžky  $n$ , ktoré TS akceptuje. Nech  $w_i = v_i u_i$ ,  $i = 1, 2$  a nech nejaká prechodová matica  $T$  popisuje aj  $u_1$  aj  $u_2$ . Potom TS akceptuje aj slovo  $v_2 u_1$ , resp.  $v_1 u_2$ .*

<sup>2\*</sup>označuje ľubovoľný zo symbolov  $\{0, 1\}$

**Dôkaz:** Môžeme predpokladať, že po akceptovaní stroj neurobí žiadne kroky. Majme postupnosť konfigurácií  $C$  stroja TS, ktorá vedie k akceptovaniu slova  $v_1u_1$ . Nech  $Q_1, Q_2, \dots, Q_p$  je postupnosť stavov pamäte, v ktorých pri tomto výpočte prekračuje hranicu medzi  $v_1$  a  $u_1$ . Potom  $C$  možno napísať

$$C = \alpha_1\beta_1\alpha_2\beta_2\dots,$$

kde

$\alpha_i$  je časť výpočtu na časti slova  $v_1$  pred  $i$ -tým prekročením hranice medzi  $v_1$  a  $u_1$   
 $\beta_i$  je časť výpočtu na časti slova  $u_1$  pred  $i$ -tým prekročením hranice medzi  $u_1$  a  $v_1$

Všimnime si výpočet TS pri vstupe na  $v_1u_2$ . Výpočet začína  $\alpha_1$ . Potom stroj v stave pamäte  $Q_1$  vstúpi na časť slova  $u_2$ . Keďže T popisuje aj  $u_1$  aj  $u_2$  a keďže sa TS pri výpočte  $C$  vracia na časť slova  $v_1$  v stave pamäte  $Q_2$ , existuje výpočet  $\gamma_1$  TS, ktorý začína vstúpením TS na  $u_2$  v stave pamäte  $Q_1$  a jeho opustením smerom doľava v stave pamäte  $Q_2$ . Analogicky ďalej. Výpočet  $D$  TS na slove  $v_1u_2$  teda možno popísať nasledovne

$$D = \alpha_1\gamma_1\alpha_2\gamma_2\dots$$

Ak pri výpočte  $C$  stroj akceptoval s hlavou v časti  $v_1$ , bude tomu tak aj v prípade  $D$ . Ak pri výpočte  $C$  stroj akceptoval s hlavou v časti  $u_1$ , tak hranicu medzi  $u_1$  a  $v_1$  poslednýkrát prekračoval v stave pamäte  $Q_p$ . Z tohto stavu pamäte sa bez opustenia slova  $u_1$  vedel dostať do stavu pamäte  $Q_A$  odpovedajúcim akceptovaniu. Druhá položka  $T(Q_p, Q_A)$  je teda 1 a zaručuje, že ak stroj vstúpi na  $u_2$  v stave pamäte  $Q_p$ , môže bez opustenia slova  $u_2$  dosiahnuť stav pamäte  $Q_A$  - môže teda akceptovať.

□

Analogické tvrdenie vieme dokázať aj pre prípad, keď prechodová matica popisuje dve rôzne koncové podslova toho istého vstupného slova.

Teraz už môžeme prejsť k využitiu prechodovej matice pri dolnom odhade na priestor.

**Veta 6.6** *Nech  $T$  je  $L(n)$  páskovo ohraničený, kde  $L(n)$  nie je zhora ohraničená dolný odhad na žiadnou konštantou. Potom*

$$\sup_{n \rightarrow \infty} \frac{L(n)}{\log \log n} > 0$$

**Dôkaz:**

- Keďže hodnota  $L(n)$  nie je ohraničená konštantou, potom pre každé  $k$  existuje najmenšie také  $n_k$ , že  $L(n_k) \geq k$
- Nech  $r$  je počet stavov pamäti, ktoré neobsahujú viac ako  $L(n_k)$  políčok
- Nech  $w_k$ ,  $|w_k| = n_k$  je najkratšie také, že sa pri jeho spracovaní použije aspoň  $k$  políčok

1. Ukážeme, že  $w_k$  nemôže mať dve rôzne koncové podslova popísané tou istou prechodovou maticou. Inak by sme totiž vedeli skonštruovať slovo kratšie ako  $n_k$ , pri spracovaní ktorého sa použije aspoň  $k$  políčok.

2. Potom spočítame počet všetkých možných prechodových matíc, ktorý nesmie byť menší, ako počet všetkých rôznych koncových podslov slova  $w_k$ .

**K bodu 1** Nech  $w = v_1v_2v_3$ ,  $v_2 \neq \epsilon$ ,  $T^{(n_k)}$  popisuje aj  $v_2v_3$ , aj  $v_3$ . Potom stroj pri spracovaní slova  $v_1v_3$  použije aspoň  $k$  políčok pásky.

Prečo? Dôkaz je podobný ako v prípade Veta 6.6, treba len akceptujúcu konfiguráciu nahradiť konfiguráciou, ktorá použije  $k$  políčok pásky. To je ale spor s predpokladom, že  $w_k$  bolo najkratšie také, pri spracovaní ktorého sa použilo aspoň  $k$  políčok pásky.

**K bodu 2** Musí teda existovať aspoň  $n_k - 1$  rôznych prechodových matíc typu  $r \times r$ . Takýchto matíc je  $4^{r^2}$ , kde  $r = s \cdot t^{L(n_k)L(n_k)}$ . Preto

$$\begin{aligned} n_k - 1 &\leq 4^{r^2} \\ \log(n_k - 1) &\leq r^2 \log 4 = (s \cdot t^{L(n_k)} \cdot L(n_k))^2 \cdot \log 4 \\ \log \log(n_k - 1) &\leq 2(\log s + L(n_k) \log t + \log L(n_k)) + 1 \\ \frac{1}{2} \log \log n_k &\leq 3L(n_k) \log 2st \end{aligned}$$

<sup>3</sup> Pre nekonečne veľa  $n$  takých, že  $n = n_k$  (pre nejaké  $k$ ) teda platí

$$\frac{L(n)}{\log \log n} \geq \frac{1}{6 \log 2st}$$

□

## 6.2 Hierarchia času a priestoru

Pri definovaní výpočtového modelu definujeme aj miery zložitosti, ktoré sú preň relevantné. Potom má zmysel uvažovať o triede problémov, ktoré sa dajú riešiť s nejakým obmedzením na tú-ktorú mieru zložitosti. Prirodzene sa tak dostávame k zložitostným triedam. Hlavným výpočtovým modelom je pre nás TS, pričom základné zložitostné triedy sú definované ohraničením jeho mier zložitosti - času a priestoru. Vo všeobecnosti sa používa značenie  $DTIME(f(n))$  na označenie triedy problémov, ktoré sa dajú na deterministickom TS riešiť s časovou zložitosťou  $f(n)$  a  $DSPACE(f(n))$  označuje tú triedu problémov, ktoré sa dajú na deterministickom TS riešiť s priestorovou zložitosťou  $f(n)$ .

Majme teda TS, ktorý je  $f(n)$  - či už časovo, alebo priestorovo - ohraničený. Ak by sme nejaké inému TS túto informáciu nejakou vedeli odovzdať, mohol by ju možno využiť. Bol by napríklad schopný spočítať počet všetkých možných konfigurácií, do ktorých sa pôvodný TS pri výpočte na danom vstupe (dĺžky  $n$ ) môže dostať a tak by vhodnou simuláciou a počítaním krokov mohol identifikovať, že sa pôvodný TS dostal do cyklu. Teraz už zrejme budeme rozumieť, prečo uvádzame nasledujúce definície, zavádzame definované pojmy.

*páskovo konštruovateľná funkcia* **Definícia 6.1** Funkcia  $L(n)$  sa nazýva páskovo konštruovateľná, ak existuje deterministický Turingov stroj  $T$ , ktorý je  $L(n)$  priestorovo ohraničený a ktorý na každom vstupe dĺžky  $n$  použije na prvej páске presne  $L(n)$  políčok.

*časovo konštruovateľná funkcia* **Definícia 6.2** Funkcia  $T(n)$  sa nazýva časovo konštruovateľná, ak existuje deterministický Turingov stroj  $T$ , ktorý na každom vstupe dĺžky  $n$  spraví presne  $T(n)$  krokov.

**Veta 6.7** Ku každej konštruovateľnej funkcii  $f(n)$  existuje rekurzívny jazyk  $LT$ , resp.  $LS$  taký, že  $LT \notin DTIME(f(n))$  a  $LS \notin DSPACE(f(n))$ .

---

<sup>3</sup> $\log \log(n_k - 1) \geq \frac{1}{2} \log \log n_k$  pre  $n_k \geq 3$

**Dôkaz:** Uvedieme dôkaz len pre prípad času; dôkaz pre priestor je analogický. Nech  $f(n)$  je konštruovateľná funkcia a  $M_f$  TS, ktorý ju konštruuje. Nech  $x$  je binárny reťazec. Potom označme  $M_x$  ten DTS, ktorého kódom je práve  $x$ ; ak  $x$  nie je kódom TS, môžeme ho považovať za kód TS rozhodujúceho prázdny jazyk. Definujme jazyk  $L_f$  nasledovne:

$L_f = \{x \mid \text{výpočet } M_x \text{ na vstupe } x \text{ sa zastaví po najviac } f(|x|) \text{ krokoch v zamietajúcej konfigurácii}\}$

**Jazyk  $L_f$  je rekurzívny.** Dôkaz spravíme konštrukciou DTS  $T$ , ktorý ho rozhoduje.

- $T$  so vstupom  $w$  najprv odsimuluje  $M_f$
- chápe  $w$  ako kód  $M_w$  a simuluje  $f(|w|)$  krokov výpočtu  $M_w$  so vstupom  $w$
- ak  $M_w$  zastavil v zamietajúcej konfigurácii, akceptuje.

$L_f \notin \mathbf{DTIME}(f(n))$  vyargumentujeme sporom. Nech existuje DTS  $M$  rozhodujúci  $L_f$  v čase  $f(n)$ . Nech  $M = M_x$  pre nejaké  $x$ .

$$x \in L_f = L(M) \Leftrightarrow x \notin L(M_x) = L(M)$$

□

### 6.2.1 Priestorová hierarchia

**Veta 6.8** Nech  $f_1(n), f_2(n)$  sú konštruovateľné funkcie a nech  $f_1(n) = o(f_2(n))$ . Potom existuje jazyk  $L$ , ktorý patrí do  $DSPACE(f_2(n))$ , ale nepatrí do  $DSPACE(f_1(n))$ .

**Dôkaz:** Jazyk  $L$  popíšeme konštrukciou TS  $T$ , ktorý ho rozhoduje. Tento budeme konštruovať tak, aby bol  $f_2(n)$  priestorovo ohraničený. Potom ukážeme, že  $L$  sa nedá rozhodovať žiadnym TS s priestorovou zložitou  $f_1(n)$ .

Nech  $T_2 = (\Sigma_2, \Gamma_2, K_2, \delta_2, B, q_{02}, F_2)$  je TS, ktorý konštruuje funkciu  $f_2(n)$ . Stroj  $T$ ,  $T = (\Sigma, \Gamma, K, \delta, B, q_0, F)$ , rozhodujúci jazyk  $L$ , pracuje nasledovne:

1. abeceda  $\Sigma = \{a_x \mid a \in \Sigma_2, x \in \{0, 1\}\}$
2. nech  $w$  je vstupné slovo; označme  $bin(w)$  binárny reťazec tvorený indexami vstupného slova  $w$
3. ignorujúc indexy  $T$  simuluje  $T_2$  a vyhradí na páske priestor veľkosti  $f_2(n)$ ; odteraz zastaví bez akceptovania vždy, keď by chcel opustiť tento vyhradený priestor. Takto sme zabezpečili, že  $L \in DSPACE(f_2(n))$
4.  $T$  chápe  $bin(w)$  ako binárny zápis čísla  $i$ ; vygeneruje kód  $i$ -teho TS  $M_i$
5.  $T$  simuluje  $M_i$  so vstupom  $bin(w)$
6.  $T$  akceptuje vtedy a len vtedy ak  $M_i$  zastane a zamietá, resp. ak sa výpočet  $M_i$  zacyklí

Diagonalizáciou ukážeme, že neexistuje  $f_1(n)$  priestorovo ohraničený TS rozhodujúci  $L = L(T)$ .

Nech takýto TS existuje. Nech je v číslování strojov  $j$ -ty, teda  $M_j$ . Uvažujme vstupné slovo  $w$  také, že  $bin(w)$  je dvojkovým zápisom čísla  $j$ . Uvedomme si, že takýchto slov je vlastne nekonečne veľa, lebo ak  $bin(w)$  je zápisom čísla  $j$ , tak aj  $0 * bin(w)$  je zápisom čísla  $j$ . Potrebujeme vyargumentovať, že sa  $T$  v svojej práci úspešne dostane cez body 3.,4.,5. Potom v bode 6. nastane spor.

**K bodu 4.** Nech  $N_4$  je také, že  $f_2(N_4) \geq$  dĺžka kódu  $M_j$

**K bodu 5.** Vzhľadom k tomu, že abeceda stroja  $T$  je fixná, musí abecedu stroja  $M_j$  kódovať. Na kódovanie znakov použije  $c(j)$  znakov (je to konštanta závislá od  $j$ ). Takto sa priestorová zložitosť zmení z  $f_1(w)$  na  $c(j)f_1(w)$ . Keďže  $f_1(n) = o(f_2(n))$ , existuje  $N_5$  také, že  $\forall n \geq N_5: c(j)f_1(n) \leq (f_2(n))$ .

**K bodu 6.** Aby sme zistili, či sa  $M_j$  zacyklil, potrebujeme vedieť (stačí) počet všetkých možných konfigurácií stroja  $M_j$  na slove dĺžky  $|w|$ . Na výpočet tejto hodnoty  $H$  stačí znalosť  $f_1(n)$  - to vieme, mohutnosť pracovnej abecedy - to vyčítame z kódu, dĺžka vstupného slova - tú vieme zistiť.

Keďže  $f_1(n)$  je konštruovateľná, nie je problém  $H$  vypočítať (napísať TS, ktorý bude  $H$  počítať). Pri simulovaní  $M_j$  počítame počet odsimulovaných krokov a pri presiahnutí hodnoty  $H$  vieme, že stroj sa zacyklil.  $H = |\Gamma|^{f_1(n)} f_1(n) |K_j|$ , čo znamená, že existuje  $N_6$  také, že počet bitov, potrebných na zapísanie hodnoty  $H$  je menší ako  $f_2(N_6)$ .

Nech  $N = \max\{N_4, N_5, N_6, \lfloor \log j \rfloor + 1\}$ . Uvažujme slovo  $v = O^*bin(w)$ ,  $|v| \geq N$ . Pre takto zvolené slovo sa  $T$  dostane až do bodu 6 a platí:  $v \in L = L(T) = L(M_j) \Leftrightarrow v \notin L(M_j)$

□

## 6.2.2 Hierarchia času

**Veta 6.9** Nech  $f_2(n)$  je konštruovateľná funkcia a nech  $f_1(n) \log f_1(n) = o(f_2(n))$ . Potom existuje jazyk, ktorý patrí do  $DTIME(f_2(n))$ , ale nepatrí do  $DTIME(f_1(n))$ .

**Dôkaz:** Analogicky ako v prípade priestorovej hierarchie. Navrhujeme  $f_2(n)$ -časovo ohraničený TS  $M$ , ktorý rozpoznáva jazyk  $L \in \{0, 1\}^*$ .

Stroj  $M$  na vstupe  $w$ ,  $|w| = n$ ,

1. vypočíta  $f_2(|w|)$
2.  $w = 1^*0v$   $v$  chápe ako kód TS  $M_v$
3.  $M$  simuluje prvých nanaajvyš  $f_2(|w|)$  krokov stroja  $M_v$
4.  $M$  akceptuje  $w$  práve vtedy, ak  $M$  úspešne ukončil simuláciu  $M_v$ , pričom  $M_v$  neakceptoval  $w$

Predpokladajme, že  $M$  je  $f_1(n)$ -časovo ohraničený. Potom existuje taký vstup  $\alpha$ , že  $M = M_\alpha$ . Keďže  $M_\alpha$  môže mať ľubovoľný počet pásov a my ho simulujeme na stroji s fixovaným počtom pásov, môže dôjsť k logaritmickejmu spomaleniu. To je jeden zdroj spomalenia. Druhý zdroj je počítanie krokov simulovaného stroja - počítadlo je veľkosti  $4 \log f_1(n)$  - teda tiež logaritmickejmu spomaleniu.

Vzhľadom na platnosť podmienky  $f_1(n) \log f_1(n) = o(f_2(n))$  má  $M_\alpha$  dostatočne veľa času na simuláciu  $M_\alpha$  a v bode 4. dochádza k sporu, keď  $M$  akceptuje vstupné slovo  $\alpha$  práve vtedy, keď ho  $M_\alpha$  neakceptuje. Pritom však  $L(M) = L(M_\alpha)$ . □

Požiadavka na konštruovateľnosť funkcie v predchádzajúcej vete je podstatná. Ak by sme od nej upustili, dostaneme pomerne prekvapivý výsledok:

**Veta 6.10** Existuje rekurzívna funkcia  $f : N \rightarrow N$  taká, že  $DTIME(f(n)) = DTIME(2^{f(n)})$

**Dôkaz:** Dôkaz je založený na vhodnej konštrukcii funkcie  $f(n)$ . Zdefinujeme funkciu  $f$  tak, že výpočet každého DTS na vstupe dĺžky  $n$  sa zastaví alebo nanaajvýš po  $f(n)$  krokoch, alebo po viac ako  $2^{f(n)}$  krokoch, alebo sa nezastaví vôbec.

Uvažujme rastúce usporiadanie kódov TS -  $M_0, M_1, \dots$ . Pre každú dvojicu prirodzených čísel  $i, k$  definujeme vlastnosť  $P(i, k)$

$\mathbf{P(i, k)} = 1 \Leftrightarrow$  výpočet každého spomedzi strojov  $M_0, M_1, \dots, M_i$  na každom vstupe dĺžky  $i$  sa buď zastaví po nanaajvýš  $k$  krokoch, alebo sa zastaví po viac ako  $2^k$  krokoch, alebo sa nezastaví vôbec.

Napriek tomu, že niektoré spomedzi uvažovaných výpočtov môžu byť nekonečné, je vlastnosť  $P(i, k)$  rozhodnuteľná - pre každý zo strojov  $M_0, M_1, \dots, M_i$  stačí odsimulovať nanaajvýš  $2^{k+1}$  krokov výpočtu na každom vstupe dĺžky  $i$ .

**Uvažujme** postupnosť čísel

definícia  $f(i)$

$$k_1 = 2i \text{ a pre } j = 2, 3, \dots \quad k_j = 2^{k_{j-1}} + 1$$

**Uvedomme si**, že každý zo vstupov dĺžky  $i$  môže narušiť platnosť vlastnosti  $P(i, k_j)$  pre nanaajvýš jednu hodnotu  $j$ , resp.  $k_j$ . Ak totiž stroj na danom vstupe zastal po  $t$  krokoch, pričom  $k_j < t < 2^{k_j}$ , tak zastal po nanaajvýš  $2^{k_{j+1}}$  krokoch.

**Existuje** číslo  $\mathbf{c}$  také, že  $\mathbf{P(i, c)} = 1$ .

**Potom** pomocou tejto hodnoty definujeme hodnotu našej funkcie v bode  $i$  -  $\mathbf{f(i)} := \mathbf{k_c}$

Keďže  $DTIME(f(n)) \subseteq DTIME(2^{f(n)})$ , stačí ukázať iba platnosť opačnej inklúzie.

$\mathbf{L} \in \mathbf{DTIME}(2^{f(n)}) \Rightarrow \mathbf{L} \in \mathbf{DTIME}(f(n))$  Jazyk  $L$  je rozhodovaný nejakým TS, nech je to stroj  $M_j$ , v čase  $2^{f(n)}$ . Potom pre každý vstup  $w$  dĺžky  $|w| \geq j$  (teda pre všetky vstupy až na konečne veľa) nie je možné, aby sa výpočet stroja  $M_j$  zastavil po viac ako  $f(|w|)$  a menej ako  $2^{f(|w|)}$  krokoch.<sup>4</sup> Keďže ale stroj určite zastaví po nanaajvýš  $2^{f(|w|)}$  krokoch, tak vlastne zastane po nanaajvýš  $f(|w|)$  krokoch.

Táto vlastnosť ale platí len pre slová dĺžky aspoň  $j$ . Pre ostatné - ktorých je ale *konečne veľa*, stačí na ich rozpoznanie konečný stav. Pre túto konečnú množinu slov teda na rozpoznanie stačí lineárny čas.

□

## 6.3 Nedeterministický TS

Poslednou modifikáciou Turingovho stroja je pridanie nedeterministického rozhodovania. Nedeterministickým rozhodovaním myslíme potenciálnu možnosť výberu nasledujúceho kroku z *konečnej množiny* možností. Ak pridáme TS možnosť nedeterministického rozhodovania, dostaneme nedeterministický TS.

Formálnejšie: *Nedeterministický k-páskový TS* (NTS) je sedmica  $(\Sigma, \Gamma, K, \delta, B, q_0, F)$ , *NTS* kde

$\Sigma, \Gamma, K, B, q_0, F$  majú význam ako pri definícii TS,

<sup>4</sup>pri definovaní hodnoty  $f(d)$ ,  $d \geq j$  sme brali do úvahy aj stroj  $M_j$

$\delta : (\Sigma \times \Gamma^k \times K) \rightarrow 2^{\Gamma^k \times K \times \{0,1,-1\}^k \times \{0,1,-1\}}$  je prechodová funkcia<sup>5</sup>.

Pojmy konfigurácia, krok, výpočet, akceptujúci výpočet ostávajú rovnaké ako v prípade deterministického Turingovho stroja. Čo si treba vyjasniť, je akceptovanie. Vzhľadom k tomu, že z danej konfigurácie existuje viacero možností, ako pokračovať vo výpočte, máme možnosť realizovať pre jedno vstupné slovo viacero výpočtov. Pritom niektoré z týchto výpočtov môžu končiť v akceptujúcej konfigurácii, iné v zamietajúcej. Čo v takom prípade?

NTS *akceptuje* práve vtedy, keď existuje akceptujúci výpočet a *zamietá*, ak akceptujúci výpočet neexistuje. To znamená, že zamietá, ak všetky výpočty na danom vstupe končia v zamietajúcej konfigurácii (resp. neskončia)

Zamyslime sa nad tým, čo v prípade TS nedeterminizmus prináša. Ukážeme, že v porovnaní s deterministickým TS sa výpočtová sila nemení. Mení sa len zložitosť výpočtov.

**Veta 6.11** *Nech  $N$  je nedeterministický TS rozhodujúci jazyk  $L$ . Potom existuje ekvivalentný deterministický TS  $D$ .*

**Dôkaz:** Bez ujmy na všeobecnosti môžeme predpokladať, že NTS  $N$  je jednopáskový. Uvažujme strom výpočtu  $\mathcal{T}$  stroja  $N$ . V koreni stromu je počiatočná konfigurácia, vrchol–konfigurácia  $C$  má ako syna vrchol–konfiguráciu  $C'$  práve vtedy, keď  $C \mapsto_N C'$ .

DTS  $D$  bude dvojpáskový a bude simulovať výpočet NTS  $N$  "prechádzaním" stromu výpočtu do šírky. V každom okamihu simulácie je na jednej z pásek DTS uložená postupnosť konfigurácií z  $i$ -tej úrovne  $\mathcal{T}$  (stará), na druhej páske sa vytvára  $(i+1)$ -á úroveň stromu výpočtu  $\mathcal{T}$  (nová). Úlohu pásek (ktorá je nová a ktorá stará) rozlišujeme stavom.

$i \rightarrow i+1$

Nech obsahom starej pásky je  $i$ -ta úroveň  $\mathcal{T}$ .

- hlavy DTS  $D$  stoja na začiatku pásek
- kým nie je koniec starej pásky
  - nech  $C$  je konfigurácia zo starej pásky;
  - vytvor na novej páske konfigurácie  $C_1, \dots, C_k$  také, že  $C \mapsto_N C_k$
- posuň hlavy na oboch páskach doľava
- vymeň úlohy starej a novej pásky

*ukončenie*

Ukončenie práce DTS nastane z dvoch dôvodov:

- ak DTS  $D$  vytvorí akceptujúcu konfiguráciu, zastane a akceptuje
- ak vytvorená úroveň obsahuje len listy, pričom každý odpovedá zamietajúcej konfigurácii, DTS  $D$  zastane a zamietá.

Časová aj priestorová zložitosť touto simuláciou narastie exponenciálne. Efektívnejšie simulácie v nasledujúcej časti.

□

<sup>5</sup>Pre množinu  $A$  označíme symbolom  $2^A$  množinu všetkých podmnožín množiny  $A$

## 6.4 Vzťahy medzi zložitostnými triedami

V tejto časti nás budú zaujímať vzťahy medzi zložitostnými triedami. Špeciálne si budeme všímať vzťah nedeterministického a deterministického času a priestoru, resp. to, koľko "zaplatíme" za prechod od nedeterminizmu k determinizmu.

Pre niektoré (najčastejšie používané) zložitostné triedy používame špeciálne označenie:

$$\begin{array}{llll}
 P = & = \bigcup_{k \geq 0} DTIME(n^k) & DLOG & = DSPACE(\log n) \\
 NP = & = \bigcup_{k \geq 0} NTIME(n^k) & NLOG & = NSPACE(\log n) \\
 ETIME & = \bigcup_{k \geq 0} DTIME(2^{kn}) & PSPACE & = \bigcup_{k \geq 0} DSPACE(n^k) \\
 NETIME & = \bigcup_{k \geq 0} NTIME(2^{kn}) & NPSPACE & = \bigcup_{k \geq 0} NSPACE(n^k) \\
 EXPTIME & = \bigcup_{k \geq 0} DTIME(2^{n^k}) & & \\
 NEXPTIME & = \bigcup_{k \geq 0} NTIME(2^{n^k}) & & 
 \end{array}$$

**Veta 6.12** *Nech  $f(n)$  je konštruovateľná funkcia. Potom*

1.  $DSPACE(f(n)) \subseteq NSPACE(f(n))$  a  $DTIME(f(n)) \subseteq NTIME(f(n))$
2.  $NTIME(f(n)) \subseteq DSPACE(f(n))$
3.  $NSPACE(f(n)) \subseteq \bigcup_{c > 0} DTIME(c^{\log n + f(n)})$
4.  $NTIME(f(n)) \subseteq \bigcup_{c > 0} DTIME(c^{f(n)})$
5.  $NSPACE(f(n)) \subseteq DSPACE(f(n)^2)$  pre funkciu  $f(n) \geq \log n$  [**Savitchova veta**]

**Dôkaz:**

1. platí triviálne – determinizmus je špeciálnym prípadom nedeterminizmu
2. Nech  $N$  je NTS,  $f(n)$  jeho časová zložitosť,  $d$  miera nedeterminizmu (odchádzajúci stupeň vrchola v strome výpočtu; max počet možností výberu na pravej strane prechodovej funkcie). Uvažujme nasledovnú simuláciu NTS  $N$  deterministickým strojom  $D$ :
  - $D$  si na prvej páske pamätá vstupné slovo
  - na druhej páske systematicky generuje postupnosť  $a_1 a_2 \dots a_f(n)$ ,  $a_i \in \{1, 2, \dots, d\}$
  - na tretej páske simuluje výpočet  $N$  so vstupom  $w$ , pričom v  $i$ -tom kroku vyberá  $a_i$ -tu možnosť z množiny možností; ak ich toľko nie je, prechádza na ďalšiu postupnosť
  - ak sa pri simulácii dostane  $D$  do akceptujúcej konfigurácie  $N$ , akceptuje a zastane. Inak zamietá.
3. Nech  $N$  je  $f(n)$  priestorovo ohraničený TS. Každá konfigurácia stroja  $N$  je jednoznačne určená pozíciou hlavy na vstupe, obsahom pásky a polohou hlavy na nej, stavom. Preto je počet  $\#(CN)$  všetkých možných konfigurácií stroja  $N$  možno ohraničiť nasledovne

$$\#(CN) \leq n |K| \log(f(n)) |\Gamma|^{f(n)} \leq d^{\log n + f(n)}$$

Ak chceme vedieť, či  $N$  akceptuje vstupné slovo  $w$ , pýtame sa, či sa zo vstupnej konfigurácie (ktorá je jednoznačná) vie  $N$  dostať do akceptujúcej konfigurácie (konfigurácia s akceptujúcim stavom; bez ujmy na všeobecnosti predpokladáme, že  $N$  má jediný akceptujúci stav). Ak si predstavíme, že všetky konfigurácie tvoria (orientovaný) konfiguračný graf :

- množina vrcholov je tvorená množinou konfigurácií
- $C_1$  je synom  $C_2$  práve vtedy keď z  $C_1$  sa dá v jednom kroku dostať do  $C_2$ , zredukovali sme náš problém na problém dosiahnuteľnosti akceptujúcej konfigurácie z počiatočnej konfigurácie. Samotné prehľadávanie grafov je zložitosti  $O(m) = O(\#(CN)^2)$ . Predpokladá ale, že poznáme susedov. My ich vždy, keď treba, vygenerujeme. Toto generovanie spôsobí zdržanie  $O(\log n + f(n))$ . Preto je celkový čas  $O((\log n + f(n))(\#(CN))^2) = O(c^{\log n + f(n)})$

4. Analýza času simulácie z bodu 2.
5. **dôkaz Savitchovej vety** Uvažujme nasledujúcu funkciu  $OVER(C_1, C_2, t)$ , kde  $C_1, C_2$  sú konfigurácie,  $t$  je čas

$$OVER(C_1, C_2, t) = \begin{cases} 1, & \text{ak sa z } C_1 \text{ do } C_2 \text{ dostaneme za } t \text{ krokov;} \\ 0, & \text{inak.} \end{cases}$$

Nech  $T(n)$  je počet všetkých možných konfigurácií nedeterministického TS  $N$  s priestorovou zložitou  $S(n)$ . Zrejme  $T(n) = \#(CN)$ . Bez ujmy na všeobecnosti predpokladajme, že každý výpočet tohto stroja trvá presne  $T(n)$  krokov (prečo môžeme?) Potom

$$w \in L(N) \Leftrightarrow \sum_{C_A} OVER(C_0, C_A, T(n)) = 1, \text{ kde}$$

$C_0$  je počiatočná konfigurácia

$C_A$  je akceptujúca konfigurácia

disjunkcia ide cez všetky akceptujúce konfigurácie.

Funkciu  $OVER(C_1, C_2, t)$  môžeme počítať rekurzívne:

$$\begin{aligned} t = 0, & \quad OVER(C_1, C_2, t) = 1 \Leftrightarrow C_1 = C_2; \\ t = 1, & \quad OVER(C_1, C_2, t) = 1 \Leftrightarrow C_1 \mapsto C_2; \\ t \geq 2, & \quad OVER(C_1, C_2, t) = \sum_C OVER(C_1, C, t/2) \wedge OVER(C, C_2, t/2), \\ & \quad \text{kde disjunkcia ide cez všetky konfigurácie } C \text{ stroja } N. \end{aligned}$$

Výpočet rekurzívnej funkcie simulujeme na TS simuláciou systémového zásobníka. Potrebujeme si uchovávať jednotlivé volania = parametre  $C_1, C_2, t$ . O priestorovej zložitosti  $S_D(n)$  platí:

$$\begin{aligned} S_D(n) &= O(\text{hlúbka vnorenia} \cdot \text{veľkosť hlavičky volania}) \\ &= O(\log T(n) \cdot S(n)) \end{aligned}$$

Preto  $S_D(n) = O(S^2(n))$ .

□

Zhrnutím dostávame

$$DLOG \subseteq NLOG \subseteq P \subseteq NP \subseteq NPSpace = PSPACE \subseteq EXPTIME \subseteq NEXPTIME$$

Pri všetkých inklúziách predpokladáme, že sú ostré, ale ani pre jednu z nich sa to nepodarilo ani dokázať ani vyvrátiť. Z výsledkov o hierarchiách vieme, že

$$\begin{array}{ll} DLOG \subseteq PSPACE & NLOG \subseteq NPSPACE \\ P \subseteq EXPTIME & NP \subseteq NEXPTIME \end{array}$$

Ukázať ostrosť niektorej z inklúzií je centrálnym problémom teórie zložitosti. Prezentujeme techniku, ktorá by mohla byť nápomocná.

**Veta 6.13** *Ak  $DSPACE(n) \subseteq P$ , tak  $P = PSPACE$*

**Dôkaz:** Keďže  $P \subseteq PSPACE$ , potrebujeme vlastne ukázať, že za predpokladu, že  $DSPACE(n) \subseteq P$ , platí aj opačná inklúzia  $PSPACE \subseteq P$ .

Nech teda  $DSPACE(n) \subseteq P$  a  $L \in PSPACE$ . Ukážeme, že potom  $L \in P$ .

Uvažujme ten DTS  $M$  polynomiálnej priestorovej zložitosti  $p(n)$ , ktorý rozhoduje  $L$ ,  $L(M) = L$ . Skonstruujme nový jazyk  $L'$  nasledovne:

$$L' = \{w10^{p(|w|)} \mid w \in L\}$$

Konstruáciou TS  $M'$ ,  $L(M') = L'$ , lineárnej časovej zložitosti ukážeme, že  $L' \in DSPACE(n)$ .  $M'$  pracuje (napríklad) takto:

- $M'$  nájde najpravší symbol 1 = tým pozná aj vstupné slovo  $w$ , aj má vymedzený priestor veľkosti  $p(|w|)$
- $M'$  simuluje  $M$  so vstupom  $w$ , na čo mu stačí priestor  $p(|w|)$ , čo je však lineárne vzhľadom na veľkosť vstupu do  $M'$ !

Preto  $M'$  patrí do  $DSPACE(n)$  a teda aj  $P$ . To znamená, že existuje polynomiálne časovo ohraničený TS  $M''$ , ktorý rozpoznáva  $L'$ . Tento stroj  $M''$  využijeme na rozpoznanie pôvodného jazyka  $L$  v polynomiálnom čase nasledovne:

- v polynomiálnom čase vyrobí zo vstupu  $w$  na pracovnú pásku  $w10^{p(|w|)}$
- simuluje výpočet  $M''$  na vstupe  $w10^{p(|w|)}$

Tým sme ukázali, že  $PSPACE \subseteq P$ .

□

**Dôsledok 6.14**  $P \neq DSPACE(n)$

## 6.5 Uzavretosť nedeterministického TS na komplement

V poslednej časti tejto kapitoly sa budeme venovať otázke uzavretosti nedeterministického priestoru na komplement. Otázka uzavretosti kontextových jazykov, a teda lineárneho priestoru, na komplement bola otvorená veľmi dlho. Napokon sa ju podarilo vyriešiť v rovnakom čase nezávisle dvom ľuďom—Neilovi Immermanovi a Robertovi szelepcsenyimu<sup>6</sup>. Získaný výsledok je všeobecnejší, hovorí o uzavretosti nedeterministického priestoru na komplement.

**Veta 6.15 (Immerman-Szelepcsenyi)** *Nech funkcia  $f(n)$  je konštruovateľná, pričom  $f(n) \geq \log n$ . Potom trieda  $NSPACE(f(n))$  je uzavretá na komplement.*

<sup>6</sup>Robert bol v tom čase študentom informatiky na našej fakulte

**Dôkaz:** Majme jazyk  $L$  rozpoznávaný  $f(n)$ -priestorovo ohraničeným TS  $T$ . Kedy slovo  $w$  patrí/nepatrí do jazyka  $L$ ? Slovo  $w$  patrí do  $L(T)$  práve vtedy, ak existuje akceptujúci výpočet  $T$  na  $w$  a nepatrí do  $L(T)$  práve vtedy, ak žiadna konfigurácia, dosiahnuteľná výpočtom začínajúcom v počiatočnej konfigurácii  $s$  na páske, nie je akceptujúca. Na tomto jednoduchom fakte je založený algoritmus, ktorý rozhoduje jazyk  $L^C = L^C(T)$ . Označme

**KON** množinu všetkých konfigurácií stroja  $T$ . Môžeme predpokladať, že na tejto množine existuje usporiadanie (napr. lexikografické na reťazcoch odpovedajúcich konfiguráciám)

**K(n)** množinu všetkých konfigurácií, do ktorých sa  $T$  môže dostať pri spracovaní slova dĺžky  $n$

**D(w)** množinu konfigurácií, dosiahnuteľných strojom  $T$  pri spracovaní slova  $w$

Pri konštrukcii stroja  $TC$ , ktorý rozhoduje komplement jazyka  $L(T)$  sa zameriame na prehľadávanie množiny  $D(w)$ —platí  $w \in L^C \iff D(w)^7$  neobsahuje akceptujúcu konfiguráciu. Problém sa teda zredukoval na prehľadávanie  $D(w)$  v priestore  $f(|w|)$ . Označme **card(M)** mohutnosť množiny  $M$

*Predpokladajme, že  $card(D(w))$  poznáme. Potom konštrukcia NTS, ktorý nielenže postupne vypíše všetky prvky z  $D(w)$ , ale aj správne odpovie na otázku  $w \in L^C?$  je jednoduchá:*

1. card:=0
2. postupne generuj konfigurácie z  $K(|w|)$ . Pre každú vygenerovanú konfiguráciu  $k$  nedeterministicky uhádni, či  $k \in D(w)$ . Odpoveď áno prever nedeterministickou simuláciou pôvodného stroja  $T$  so vstupom  $w$ ; pri dosiahnutí konfigurácie  $k$  je odpoveď potvrdená -  $card \leftarrow card + 1$ .
3. ak po skončení je  $card = card(D(w))$ , mali sme "v ruke" všetky dosiahnuteľné konfigurácie a môžeme korektne odpovedať  
NIE - ak niektorá z konfigurácií v  $D(w)$  bola akceptujúca, ÁNO v opačnom prípade

Vzhľadom k tomu, že nepotrebujeme všetky konfigurácie naraz, stačí nám k takému výpočtu priestor  $f(|w|)$ . Ostáva však **výpočet card(D(w))**.

- Nech  $D_i(w)$  označuje množinu tých konfigurácií, ktoré sú strojom  $T$  dosiahnuteľné zo vstupného slova  $w$  maximálne v  $i$  krokoch.
- $D_0(w)$  obsahuje len počiatočnú konfiguráciu, preto poznáme  $card(D_0(w))$ ,  $Max(D_0(w))$ .
- **výpočet card(D<sub>i+1</sub>(w))**  
Zrejme  $card(D_{i+1}(w)) = card(D_i(w)) + \Delta$ , kde  $\Delta$  je počet tých konfigurácií  $k \notin D_i(w)$ , pre ktoré existuje konfigurácia  $k_0 \in D_i(w)$  taká, že  $k_0 \rightarrow k$ . Preto
  - vyššie popísaným spôsobom začne stroj vymenovávať všetky konfigurácie  $k_0 \in D_i(w)$ . Pri overení príslušnosti k  $D_i(w)$  simuluje len  $i$  krokov výpočtu.
  - nech  $k_0 \in D_i(w)$ ,  $k_0 \rightarrow k$  a *count* je momentálna hodnota počítadla konfigurácií z  $D_i(w)$ . Ostáva overiť, či sa má  $k$  započítať do  $\Delta$ . Odpoveď je áno, ak

<sup>7</sup> $D(w)$  sa vzťahuje k pôvodnému stroju  $T$  pre jazyk  $L$

(a)  $k \notin D_i(w)$

(b) neexistuje konfigurácia  $l \in D_i(w)$  menšia ako  $k_0$  taká, že  $l \rightarrow k$ . Overenie spravíme opäť postupným vymenovávaním - tentokrát konfigurácií z  $D_i(w)$ , ktoré sú menšie ako  $k_0$ . Pre každú z nich overíme, či sa z nej v jednom kroku nedostaneme do  $k$ . Ak taká neexistuje a nám sa podarilo vymenovať všetky konfigurácie z  $D_i(w)$  menšie ako  $k_0$  (dopočítali sme sa do momentálnej hodnoty *count* počítadla pre porovnanie s hodnotou  $\text{card}(D_i(w))$ ), tak konfiguráciu  $k$  "zaradíme" do  $D_{i+1}(w)$ , teda zvýšime hodnotu  $\Delta$ , prípadne zmeníme hodnotu  $\text{Max}(D_{i+1}(w))$

Nie je ťažké si uvedomiť, že popísanú konštrukciu môžeme na viacpáskovom TS robiť v pamäti  $f(|w|)$ .

□