Prednáška 10 - Rozšírené Turingove stroje, vlastnosti RE jazykov

Ing. Viliam Hromada, PhD.

C-510
Ústav informatiky a matematiky
FEI STU

viliam.hromada@stuba.sk

1.12.2020

Opakovanie

- Na minulej prednáške sme sa bavili o Turingovych strojoch, ako o matematickom modeli výpočtového zariadenia.
- Povedali sme si, že ku každému algoritmu sa dá zostrojiť Turingov stroj, ktorý ho modeluje, resp. realizuje.
- Taktiež bola reč o nerozhodnuteľných a rozhodnuteľných problémoch.

- Rozhodnuteľné problémy sú také rozhodovacie, pre ktoré existuje algoritmus, ktorý dokáže v konečnom čase poskytnúť pre jednotlivé inštancie problémov odpoveď áno/nie
- To znamená, že existuje Turingov stroj, ktorý pre dokáže spracovať ľubovoľný vstup a vždy zastaví.
- Pre nerozhodnuteľné problémy môžu existovať Turingove stroje, ktoré síce vedia korektne vrátiť odpoveď áno, avšak pre vstupy, ktoré majú mať odpoveď nie, nemusia nikdy zastaviť - a teda počas výpočtu TS nevieme rozlíšiť, či výpočet beží kvôli tomu, že sa odpoveď ešte nenašla, alebo pretože neexistuje.

- Samotný Turingov stroj je zariadenie pozostávajúce z konečno-stavovej jednotky a z vstupno/výstupnej pásky, z ktorej TS číta a na ktorú zapisuje tzv. páskové symboly (t.j. dáta).
- Na základe aktuálneho stavu a čítaného symbolu z pásky sa následne Turingov stroj môže prepnúť do iného stavu, na pásku zapísať iný symbol a posunúť čítaciu hlavu na páske doľava alebo doprava o 1 pozíciu.
- Dnešnú prednášku začneme tým, že si popíšeme, aké **rozšírenia** Turingovych strojov môžeme uvažovať - tieto rozšírenia **neovplyvnia to**, čo s TS vieme robiť, avšak nám môžu **uľahčiť** to, čo s nimi vieme robiť.

Viacpáskový Turingov stroj

Prvé rozšírenie Turingovych strojov, ktoré budeme uvažovať sú tzv. viacpáskové Turingove stroje.

- Ako z názvu vyplýva, uvažujeme Turingov stroj, ktorý nemá len 1, ale môže používať viacero pások.
- Tieto pásky sú, rovnako ako v prípade TS s jednou páskou, nekonečné.
- Každá páska má svoju čítaciu hlavu.
- Na začiatku činnosti sa vstupné slovo napíše na prvú pásku, ostatné pásky majú všetky bunky prázdne. Čítacia hlava prvej pásky je nastavená na prvom symbole vstupu, ostatné čítacie hlavy ostatných pások sú na ľubovoľnej prázdnej bunke.

Viacpáskový Turingov stroj

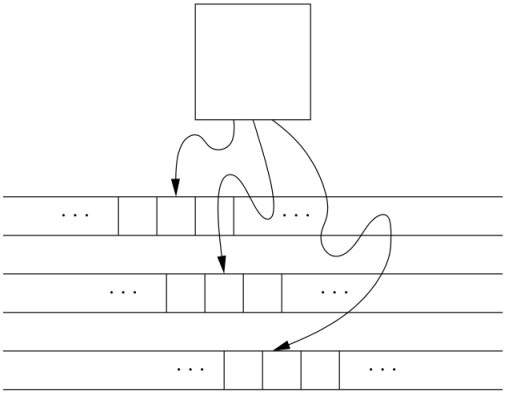


Figure: Viacpáskový Turingov stroj [1]

Krok výpočtu závisí na aktuálnom stave a čítanom symbole z každej pásky. V jednom kroku vykoná TS:

- Presunie sa do nového stavu, prípadne zostáva v starom.
- Z každej pásky prečíta symbol a následne ho prepíše podľa prechodovej funkcie.
- Hlava každej pásky vykoná pohyb - uvažujeme, že hlavy sa môžu pohnúť doľava, doprava, alebo aj **zostať stáť**, t.j. možné pohyby budú L, R, S.
- Jednotlivé hlavy sa hýbu nezávisle - pri každom prechode sa môžu všetky posunúť rovnako, niektoré môžu ísť opačným smerom, a podobne.

Príklad

Niekedy je výhodnejšie uvažovať, že máme k dispozícii TS, ktorý používa viacero pások - môže to uľahčovať návrh Turingovho stroja pre riešenie zadanej úlohy. Napríklad uvažujme, že by sme chceli akceptovať jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$

- Pri jeho riešení môžeme uvažovať, že máme k dispozícii 2 pásky.
- Na prvej páske máme vstup, ktorý postupne čítame zľava-doprava.
- Na druhej páske si ukladáme prečítané symboly.

- Najprv si na druhú pásku uložíme každé prečítané a z prvej pásky.
- Potom pri čítaní každého b z prvej pásky prepíšeme jedno a na druhej páske na b.
- Na záver pri čítaní každého c z prvej pásky zmažeme jedno b.
- Ak po prečítaní slova z prvej pásky zároveň zostane druhá páska prázdna, tak vstup na prvej páske bol v tvare $\{a^n b^n c^n \mid n \geq 1\}$.

Taký TS by bol napríklad:

- $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b, c\}$,$\Gamma = \{a, b, c, B\}$
- q_0 je počiatočný stav, B je symbol prázdnej páskovej bunky, $F = \{q_3\}$
- Prechodová funkcia δ:
  1. $\delta(q_0, (a, B)) = (q_0, (a, a), (R, R))$
  2. $\delta(q_0, (b, B)) = (q_1, (b, B), (S, L))$
  3. $\delta(q_1, (b, a)) = (q_1, (b, b), (R, L))$
  4. $\delta(q_1, (c, B)) = (q_2, (c, B), (S, R))$
  5. $\delta(q_2, (c, b)) = (q_0, (c, B), (R, R))$
  6. $\delta(q_2, (B, B)) = (q_3, (B, B), (S, S))$
- V prechodovej funkcii sú v tomto prípade dvojice - lebo sú 2 pásky, napríklad: $\delta(q_0, (a, B)) = (q_0, (a, a), (R, R))$ znamená, že ak je TS v stave q_0, na prvej páske číta a, na druhej páske číta prázdnu bunku, tak na prvú pásku zapíše a, na druhú pásku zapíše a, prvú hlavu posunie vpravo a druhú hlavu posunie vpravo.

Viacpáskový Turingov stroj

Pre viacpáskové Turingove stroje platí nasledovná veta:

Veta

Každý jazyk akceptovaný viacpáskovým Turingovym strojom je rekurzívne vyčísliteľný.

- Táto veta hovorí, že každý jazyk akceptovateľný viacpáskovým Turingovym strojom je zároveň akceptovateľný aj jednopáskovým (t.j. "klasickým") Turingovym strojom.
- Teda viacpáskové Turingove stroje sú z hľadiska výpočtových schopností **ekvivalentné** jednopáskovým Turingovym strojom.

- Existencia viacerých pások je teda len "kozmetická úprava", nijako nezosilňuje schopnosti TS.
- Dôkaz je založený na tom, že viacpáskový TS je simulovateľný jednopáskovým TS - nebudeme ho robiť, záujemcovia ho nájdu v literatúre [1]
- Keďže viacpáskové TS sú ekvivalentné jednopáskovým TS, tak ak je to potrebné, môžeme pre riešenie nejakej úlohy uvažovať, že máme viacpáskový TS - a získaný výsledok musí byť rovnako aplikovateľný aj na jednopáskový TS.

Nedeterministické Turingove stroje

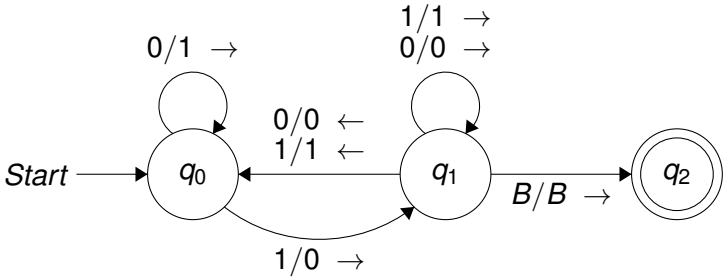
Druhým veľmi dôležitým rozšírením TS, ktoré budeme uvažovať, sú tzv. **nedeterministické Turingove stroje**.

- Nedeterministický Turingov stroj (NTS) je taký, ktorého prechodová funkcia môže pre situáciu, že sa nachádza v stave q a číta z pásky symbol X definovať prechody do viacerých rôznych situácií:

$$\delta(q, X) = \{(q_1, Y_1, D_1), (q_2, Y_2, D_2), ..., (q_k, Y_k, D_k)\}$$

- NTS si pri každom kroku výpočtu náhodne môže vybrať, ktorý prechod použije (podobne ako nedeterministický konečný automat).
- Podobne ako pri KA, ZA, aj pre NTS, že akceptuje také reťazce w, pre ktoré **existuje výpočet**, v ktorom sa pri spracúvaní reťazca w vie NTS dostať do akceptačného stavu.

Príklad



Výpočty pre vstup $w = 01$:

$$q_0 01 \vdash 1 q_0 1 \vdash 10 q_1 B \vdash 10 B q_2 B$$

Výpočty pre vstup $w = 010$: prvý sa zastaví v akceptačnom stave, druhý sa zastaví v neakceptačnom stave:

$$q_0 010 \vdash 1 q_0 10 \vdash 10 q_1 0 \vdash 100 q_1 B \vdash 100 B q_2 B$$

$$q_0 010 \vdash 1 q_0 10 \vdash 10 q_1 0 \vdash 1 q_0 00 \vdash 11 q_0 0 \vdash 111 q_0 B$$

Nedeterministický Turingov stroj

Pre nedeterministické Turingove stroje platí nasledovná veta:

Veta

Ak M_N je nedeterministický Turingov stroj, potom existuje deterministický Turingov stroj M_D taký, že $L(M_N) = L(M_D)$.

- Táto veta hovorí, že každý jazyk akceptovateľný NTS je zároveň akceptovateľný aj deterministickým (t.j. "klasickým") Turingovym strojom.
- Teda nedeterministické Turingove stroje sú z hľadiska výpočtových schopností **ekvivalentné** deterministickým Turingovym strojom (analogická situácia ako pri konečných automatoch)

- Nedeterminizmus v TS je teda rovnako, ako viacero pások, len "kozmetická úprava", nijako nezosilňuje schopnosti TS.
- Dôkaz je založený na tom, že ku každému NTS sa dá zostrojiť deterministický TS (DTS), ktorý "simuluje" výpočet nedeterministického Turingovho stroja - t.j. sleduje všetky možné výpočtové cesty, ktorými by sa mohol NTS vydať.
- Dôkaz robiť nebudeme, znovu sa dá nájsť v literatúre [1].
- Z hľadiska toho, aké jazyky vedia Turingove stroje akceptovať nie je rozdiel medzi tým, čo dokážu nedeterministické a čo dokážu deterministické Turingove stroje.

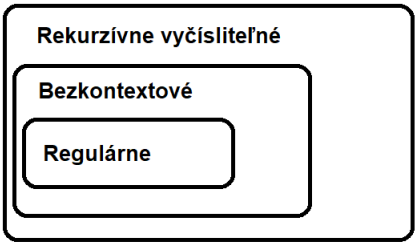
- Podstatný rozdiel je však v tom, že **momentálny** stav ľudského poznania je taký, že spôsob, ako pomocou deterministického TS simulovať nedeterministický TS je výpočtovo-náročný (má tzv. exponenciálnu zložitosť).
- Otázka, či sa dá nedeterministický TS simulovať deterministickým spôsobom s tzv. polynomiálnou zložitosťou má momentálne hodnotu milión dolárov - patrí medzi tzv. problémy tisícročia.
- Z hľadiska predmetu TZI je dôležité, že NTS a DTS sú - čo do schopnosti riešiť problémy - ekvivalentné.
- Ale keď nás zaujíma aj to, s akou zložitosťou tieto problémy riešia - tam do hry vstupuje predmet Analýza a zložitosť algoritmov.

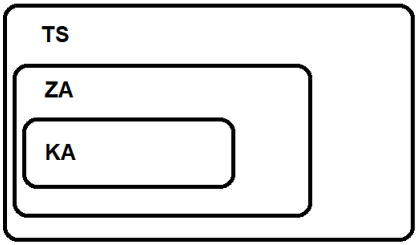
Rozšírenie TS - zhrnutie

- Povedali sme si teda, že okrem Turingovho stroja s jednou páskou, ktorý sa správa deterministicky môžeme uvažovať aj:
  - TS s viacerými páskami
  - Nedeterministické Turingove stroje
  - Rovnako môžeme uvažovať aj nedeterministické Turingove stroje s viacerými páskami
- Z hľadiska toho, čo Turingove stroje dokážu, sú však všetky tieto úpravy len formálne, keďže žiadna z nich nespôsobí to, že by sme vedeli zostrojiť niečo "silnejšie" než štandardný Turingov stroj, keďže oba typy úprav sú konvertovateľné na ekvivalentný deterministický Turingov stroj s jednou páskou.
- Poďme sa teraz bližšie pozrieť na vlastnosti rekurzívne vyčísliteľných jazykov.

Jazyky nad $A = \{0, 1\}$

- Uvažujme abecedu $A = \{0, 1\}$ a všetky možné jazyky nad touto abecedou $L \subseteq \{0, 1\}^*$.
- Aké sú vzťahy medzi triedami jazykov a výpočtovými zariadeniami, o ktorých sme sa doteraz bavili:



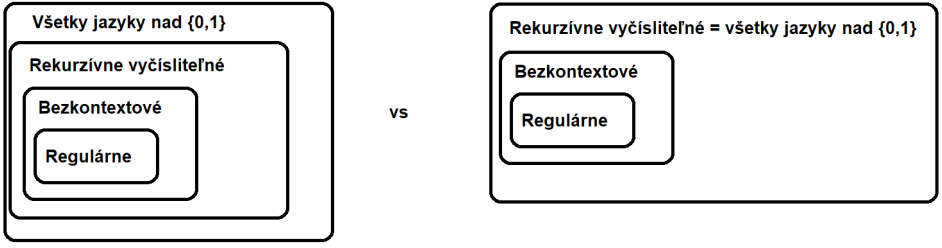


Nad abecedou $\{0, 1\}$ sú napríklad jazyky:

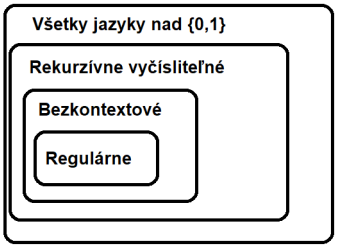
- Regulárny jazyk $(01)(0 + 1)^*$ - ten je zároveň aj bezkontextový, aj rekurzívne vyčísliteľný
- Bezkontextový jazyk $\{0^n 1^n \mid n \geq 0\}$ - ten je zároveň aj rekurzívne vyčísliteľný, ale nie je regulárny
- Rekurzívne vyčísliteľný jazyk $\{ww \mid w \in \{0, 1\}^*\}$ - ten nie je ani regulárny, ani bezkontextový

Na mieste je teraz otázka: *Existuje jazyk nad* $\{0, 1\}^*$*, ktorý nie je ani* ***rekurzívne vyčísliteľný****? T.j. existuje jazyk, pre ktorý sa* ***nedá zostrojiť Turingovy stroj****, ktorý ho akceptuje?*

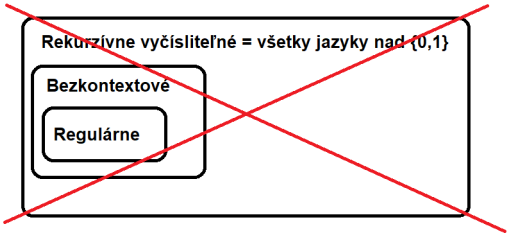
Inými slovami, ktorá z 2 uvedených možnosti platí:



Ukážeme si, že platí prvá možnosť:



vs



Jazyk, ktorý nie je RE

- Ukážeme si, že **existuje jazyk**, ktorý nie je rekurzívne vyčísliteľný
- Teda **neexistuje Turingov stroj**, ktorý by dokázal akceptovať práve reťazce, patriace do daného jazyka
- Najprv si potrebujeme povedať, ako je možné "zakódovať" Turingov stroj do binárnej postupnosti a spätne "dekódovať" binárnu postupnosť na nejaký Turingov stroj.
- To znamená, priradíme každému Turingovmu stroju nejakú binárnu postupnosť, ktorá ho popisuje (stavy, symboly, prechody) a naopak, každú binárnu postupnosť budeme vedieť interpretovať ako nejaký Turingov stroj.

Očíslovanie binárnych postupností

- Nech w je nejaká binárna postupnosť. Uvažujme celé číslo i, ktorého binárnou reprezentáciou je postupnosť $1w$. Potom w budeme nazývať i-ty reťazec.
- ε má číslo $i = 1$ (lebo $1\varepsilon = 1$)
- 0 má číslo $i = 2$ (lebo $10 = 2$)
- 1 má číslo $i = 3$ (lebo $11 = 3$)
- 00 má číslo $i = 4$ (lebo $100 = 4$)
- 01 má číslo $i = 5$ (lebo $101 = 5$)
- T.j. vieme usporiadať binárne reťazce a očíslovať ich - zároveň vidíme, že ich usporiadanie je podľa počtu symbolov a tie, ktoré majú rovnako veľa symbolov, sú ďalej usporiadané lexikograficky
- **Reťazec s číslom i budeme označovať ako w_i.**

Kódovanie Turingovho stroja

- Potrebujeme navrhnúť spôsob, ako "zakódovať" TS do binárnej postupnosti
- Uvažujme pre jednoduchosť, že vstupná abeceda TS je $\{0, 1\}$
- Nech TS obsahuje $|Q| = r$ stavov, $Q = \{q_1, q_2, ..., q_r\}$. Stav q_1 budeme vždy uvažovať ako počiatočný, stav q_2 budeme vždy uvažovať ako **jediný** akceptačný stav. **Každý TS** vieme upraviť tak, aby jeho počiatočný stav bol označený q_1 a zároveň mal jediný akceptačný stav q_2.
- Nech TS obsahuje $|\Gamma| = s$ páskových symbolov, $\Gamma = \{X_1, X_2, X_3, ...X_s\}$. X_1 bude vždy symbol 0, X_2 bude vždy symbol 1, X_3 bude vždy symbol B. Ostatné symboly môžu byť podľa ľubovôle.
- Smer pohybu hlavy L označíme ako D_1, smer pohybu hlavy R označíme ako D_2.

Kódovanie Turingovho stroja

- Každý stav, páskový symbol a pohyb hlavy teda vieme označiť nejakým číslom. Vďaka tomu vieme zakódovať prechodovú funkciu.
- Nech $\delta(q_i, X_j) = (q_k, X_l, D_m)$ pre nejaké celé čísla i, j, k, l, m. Takýto prechod zakódujeme do binárnej postupnosti ako $0^i10^j10^k10^l10^m$.
- Keďže všetky i, j, k, l, m sú celé čísla od jednotky, v uvedenej postupnosti je vždy medzi oddeľujúcimi jednotkami aspoň jedna nula.
- Viacero prechodov zapíšeme tak, že ich vzájomne oddelíme dvomi jednotkami, t.j. postupnosť $C_1 11 C_2 11 ... C_{n-1} 11 C_n$ reprezentuje prechodovú funkciu, kde prechody sú $C_1, C_2, ..., C_n$.

Príklad

Nech TS $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$, kde prechodová funkcia:

- $\delta(q_1, 1) = (q_3, 0, R)$
- $\delta(q_3, 0) = (q_1, 1, R)$
- $\delta(q_3, 1) = (q_2, 0, R)$
- $\delta(q_3, B) = (q_3, 1, L)$

Potom zakódovanie jednotlivých prechodov:

- $0^1 10^2 10^3 10^1 10^2 = 0100100010100$
- $0^3 10^1 10^1 10^2 10^2 = 0001010100100$
- $0^3 10^2 10^2 10^1 10^2 = 0001001001010 0$
- $0^3 10^3 10^3 10^2 10^1 = 0001000100010010$

A celý TS by bol zakódovaný ako:
0100100010100111000101010010011000100100101001100010001000100010

- Je zrejmé, že každý TS sa dá takto zakódovať do nejakej binárnej postupnosti.
- Taktiež, že jeden a ten istý TS má viacero binárnych postupností, do ktorých sa dá zakódovať - napríklad v uvedenom príklade existuje 4! = 24 rôznych poradí, v ktorých možno prechody zapísať.
- Taktiež je zrejmé, že nie každá binárna postupnosť sa "dá" spätne interpretovať ako nejaký Turingov stroj - napríklad postupnosť samých núl. V takom prípade definujeme, že ak sa nedá binárna postupnosť interpretovať ako TS podľa vyššie uvedeného predpisu, tak uvažujeme, že TS prislúchajúci takej binárnej postupnosti je TS s jedným stavom a žiadnymi prechodmi, t.j. taký, ktorého jazyk $L(M) = \emptyset$ a ktorý sa v momente zastaví pre hocijaký vstup.

- Vďaka uvedenej konštrukcii teda vieme uvažovať tzv i-ty Turingov stroj, M_i, t.j. Turingov stroj, ktorý ze zakódovaný ako i-ta binárna postupnosť w_i.
- Napríklad $i = 1$, t.j. TS M_1 je Turingov stroj zodpovedajúci číslu 1, teda $w_1 = \varepsilon$. To je práve reťazec, ktorý sa nedá interpretovať ako TS, preto uvažujeme, že TS M_1 má 1 stav a žiadne prechody.
- Podobne, $i = 57$ je postupnosť 111001, t.j. $w_{57} = 11001$. Táto postupnosť nezačína nulou, teda nepredstavuje nejaký TS.
- Naopak, TS popísaný v príklade ako 010010001010011000101010010011000100100101001100010001000100010010 by mal binárnu postupnosť 101001000101001100010101001001100010010010100110001000100010010010 a teda zodpovedá číslu $i = 5920415549427486994$

Ak teda vieme zakódovať TS ako nejakú binárnu postupnosť, uvažujme nasledovný jazyk nad abecedou $A = \{0, 1\}$:

Definícia

*Jazyk L_d, nazvaný **diagonalizačný jazyk**, je množina reťazcov w_i takých, že $w_i \notin L(M_i)$.*

T.j. diagonalizačný jazyk L_d obsahuje tie binárne reťazce w_i, ktoré predstavujú tie Turingove stroje, ktoré neakceptujú reťazec w_i, t.j. svoj vlastný kód v tvare binárnej postupnosti.

Uvažujme, že by sme si v tabuľke zaznačili, ktorý TS akceptuje ktoré reťazce. Nech v tabuľke i je poradové číslo Turingovho stroja, t.j. M_i a j označuje binárny vstup, t.j. w_j. 1 na pozícii (i, j) znamená, že TS M_i akceptuje reťazec w_j, 0 na pozícii (i, j) znamená, že TS M_i neakceptuje reťazec w_j.

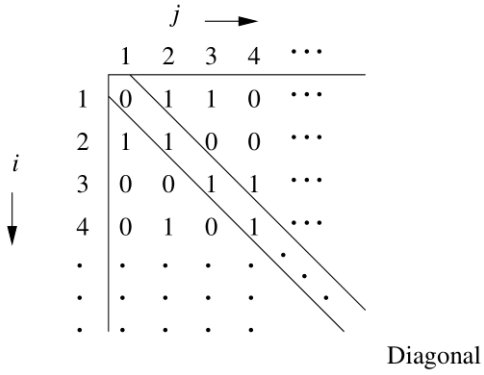


Figure: Tabuľka akceptácie j-teho reťazca i-tym TS, [1]

UPOZORNENIE: Uvedená tabuľka má ilustračné účely. Totižto niekoľko prvých TS M_1, M_2, M_3..., t.j. tie, ktoré prislúchajú najmenším kladným číslam, **nie sú reprezentácie platného TS**, t.j. všetky predstavujú TS s 1 stavom a žiadnymi prechodmi.

Predpokladajme však, že tabuľka predstavuje akceptácie reťazcov Turingovymi strojmi, napríklad:

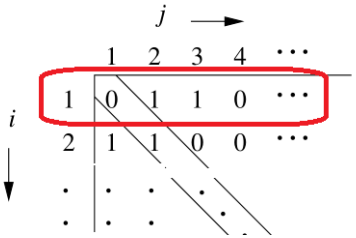


Figure: TS M_1 akceptuje reťazce [1]

Vidíme, že TS M_1 by:

1. **neakceptoval** reťazec w_1, t.j. $w_1 = \varepsilon$
2. **akceptoval** reťazec $w_2 = 0$
3. **akceptoval** reťazec $w_3 = 1$
4. **neakceptoval** reťazec $w_4 = 00$, atď.

- Zvýraznený 1 riadok popisujúci správanie konkrétneho TS (v našom prípade M_1) nazývame **charakteristický vektor** jazyka $L(M_1)$, t.j. jazyka akceptovaného TS M_1. Daný vektor teda jednoznačne popisuje, ktoré reťazce w_j patria do príslušného jazyka.
- **Diagonála** jednoznačne udáva, či TS M_i akceptuje reťazec w_i, t.j. **svoj vlastný binárny kód**.
- Ak by diagonála vyzerala tak, ako na slajde č. 32, t.j. 0111... a teda M_1 neakceptuje w_1, M_2 akceptuje w_2, M_3 akceptuje w_3, M_4 akceptuje w_4, ..., tak potom **doplnok diagonály**, t.j. vektor 1000... je práve **charakteristickým vektorom** jazyka L_d.

- Jazyk L_d tvoria **práve tie reťazce** w_i, ktoré reprezentujú Turingove stroje M_i, ktoré neakceptujú svoj vlastný kód w_i.
- A ak by sme predpokladali, že situácia je taká, že M_1 neakceptuje w_1, M_2 akceptuje w_2, M_3 akceptuje w_3, M_4 akceptuje w_4, t.j. diagonála začína 0111..., tak potom **musí platiť**, že $w_1 \in L_d$, $w_2 \notin L_d$, $w_3 \notin L_d$, $w_4 \notin L_d$,...
- A teda ak by negácia diagonály začínala 1000..., tak potom dostávame práve **charakteristický vektor** jazyka L_d.

A teraz trochu intelektuálnej mágie:

1. **Všetky možné Turingove stroje** sú popísateľné ako binárna postupnosť - t.j. v tabuľke na slajde č. 32 predstavuje **každý jeden Turingov stroj** nejaký riadok v danej tabuľke, resp. **každý riadok v tabuľke** predstavuje **charakteristický vektor** jazyka nejakého **Turingovho stroja**.
2. **Negáciou diagonály** dostávame vektor, ktorý sa **nemôže nachádzať** v tabuľke, pretože sa od i-teho riadka v tabuľke líši na i-tej pozícii - keďže na tejto pozícii dochádzalo k negácii hodnoty v tabuľke (i, i)
3. Preto negácia diagonály - t.j. jazyk L_d - **nemôže predstavovať charakteristický vektor** jazyka akceptovaného Turingovým strojom.
4. A teda jazyk L_d **nie je akceptovateľný žiadnym Turingovym strojom** - a teda to je príklad jazyka, ktorý nepatrí medzi **rekurzívne vyčísliteľné jazyky**.

L_d nie je rekurzívne vyčísliteľný jazyk

Veta

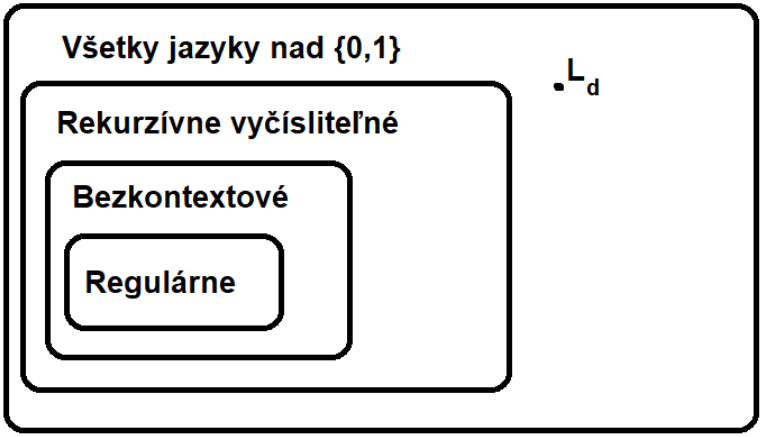
Jazyk L_d nie je rekurzívne vyčísliteľný jazyk. Teda neexistuje Turingov stroj, ktorý by dokázal akceptovať jazyk L_d.

Formálny dôkaz (sporom): Nech existuje TS M taký, že $L_d = L(M)$. Keďže jazyk $L_d \subseteq \{0,1\}^*$, tak určite má TS M reprezentáciu ako nejaký binárny reťazec, t.j. existuje číslo i, že $M = M_i$. Uvažujme, či $w_i \in L_d$:

1. Nech $w_i \in L_d$. V takom prípade M_i akceptuje w_i. Avšak v takom prípade by podľa definície L_d malo platiť $w_i \notin L_d$, pretože L_d obsahuje len tie w_j, pre ktoré $w_j \notin L(M_j)$.
2. Nech $w_i \notin L_d$. V takom prípade by $w_i \notin L(M_i)$. Avšak potom by podľa definície malo platiť, že $w_i \in L_d$.

Keďže vidíme, že w_i nemôže byť v L_d a súčasne nemôže nebyť v L_d, tak z toho vyplýva, že náš predpoklad, že existuje TS M, taký, že $L(M) = L_d$ bol zlý. A teda taký TS **nemôže existovať** a L_d **nie je** rekurzívne vyčísliteľný.

Čo potvrdzuje situáciu, že určite existuje minimálne jeden taký jazyk nad abecedou $\{0,1\}^*$, ktorý nie je rekurzívne vyčísliteľný, konkrétne L_d, a teda platí:



Vzťah nerozhodnuteľných problémov a TS

- Jazyk L_d je teda príklad jazyka, ktorý nie je rekurzívne vyčísliteľný.
- Teda k nemu neexistuje Turingov stroj, ktorý by ho akceptoval.
- Jazyk L_d je teda príklad rozhodovacieho problému, ku ktorému neexistuje ani Turingov stroj, a už vôbec nie algoritmus, ktorý by ho riešil, teda tento problém je určite nerozhodnuteľný.
- Rekurzívne vyčísliteľné jazyky sú také, pre ktoré existuje Turingov stroj, ktorý ich akceptuje, avšak tu ich vieme rozdeliť do 2 skupín, podľa toho, čo urobí Turingov stroj pre slová, ktoré do jazyka **nepatria**.

Vzťah nerozhodnuteľných problémov a TS

1. RE jazyky, pre ktoré existuje taký Turingov stroj, ktorý nielen že akceptuje slová z jazyka, avšak rovnako dokáže v konečnom čase povedať, že slovo do jazyka nepatrí, t.j. tento Turingov stroj **vždy zastaví** bez ohľadu na vstupný reťazec.
2. RE jazyky, pre ktoré existuje len taký Turingov stroj, že síce akceptuje daný jazyk, ale bez garancie zastavenia, t.j. ak je vstup z jazyka, tak o tom TS vie informovať, avšak ak vstup z jazyka nie je, tak je možné, že TS bude počítať *donekonečna*.

Je zrejmé, že prvá množina RE jazykov tvorí podmnožinu tej druhej množiny.

Rekurzívne jazyky

Definícia

*Jazyk L nazývame **rekurzívny**, ak $L = L(M)$ pre nejaký Turingov stroj M taký, že:*

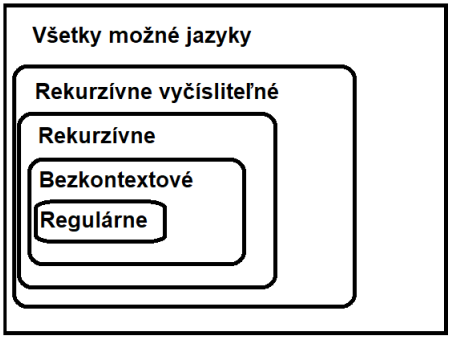
1. *Ak $w \in L$, potom M akceptuje w (a teda zastaví)*
2. *Ak $w \notin L$, tak výpočet M určite zastaví (a slovo neakceptuje).*

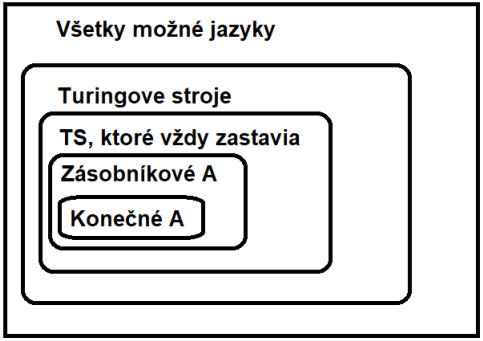
Rekurzívne jazyky predstavujú práve jazyky z bodu č. 1 na predchádzajúcom slajde. Turingove stroje, ktoré ich akceptujú, sa nazývajú aj **rozhodovače** (deciders) a predstavujú práve matematický model algoritmu, t.j. dobre definovanej postupnosti krokov, ktorá vždy skončí a vráti výsledok.

Rekurzívne jazyky

- Ak sa dívame na jazyk L ako na rozhodovací problém, resp. na reťazce jazyka L ako na inštancie problému, pre ktoré je odpoveď **áno** a reťazce, ktoré do jazyka L nepatria ako na inštancie, pre ktoré je odpoveď na rozhodovací problém **nie**, tak potom sú **rekurzívne jazyky** práve **rozhodnuteľné problémy**.
- A jazyky, ktoré nie sú rekurzívne, sú **nerozhodnuteľné** problémy.

Vzťahy jazykov (aktualizované o rekurzívne jazyky):





- V praxi nás viac zaujíma, či existuje / neexistuje **algoritmus** (t.j. TS, ktorý vždy zastaví), ktorý danú úlohu rieši, než či len existuje nejaký Turinov stroj, ktorý by dokázal jazyk akceptovať.
- Keďže TS, pri ktorých nevieme, či vždy zastavia, nám nemusia poskytnúť dostatok informácií na to, aby sme sa rozhodli, že reťazec nepatrí do príslušného jazyka - t.j. že daná inštancia problému má odpoveď **nie**.
- Preto aj v praxi je dôležitejšie delenie jazykov / problémov na rozhodnuteľné / nerozhodnuteľné je dôležitejšie, než na tie, ktoré sú / nie sú rekurzívne vyčísliteľné, t.j. či existuje / neexistuje Turingov stroj.

- Ukážme si teraz príklad jazyka, ktorý síce je rekurzívne vyčísliteľný, avšak nie je rekurzívny.
- Takýmto jazykom je napríklad tzv. **univerzálny jazyk** L_u.
- Najprv sa však zamyslime nad tzv. doplnkom jazyka.

Doplnok (komplement) jazyka

Definícia

Nech A je abeceda a L je jazyk nad touto abecedou. Potom jazyk $\overline{L}$ definovaný ako $\overline{L} = A^* \setminus L$ (t.j. rozdiel množín A^* a L) je ***doplnok (komplement)*** *jazyka L vzhľadom na abecedu A.*

T.j. ak L je jazyk nad abecedou A, tak doplnok jazyka L je množina $\overline{L}$, ktorú tvoria všetky také reťazce, ktoré **nepatria** do jazyka L. Samozrejme aj $\overline{L}$ je jazyk, keďže sa jedná o množinu reťazcov.

Príklady:

- Nech $A = \{0, 1\}$. Nech $L = 01(0 + 1)^*$, t.j. L sú všetky reťazce začínajúce 01. Potom $\overline{L} = (\varepsilon + 0 + 1) + 11(0 + 1)^* + 10(0 + 1)^* + 00(0 + 1)^*$, t.j. všetky reťazce **nezačínajúce 01**.
- Nech $A = \{a, b\}$. Nech $L = \{w \in \{a, b\}^* \mid \sharp_a(w) = \sharp_b(w)\}$, t.j. L sú všetky reťazce z písmen a, b obsahujúce rovnaký počet a a b. Teda doplnok $\overline{L}$ je jazyk obsahujúci všetky reťazce z písmen a a b, ktoré neobsahujú rovnaký počet a a b.
- Nech $A = \{0, 1\}$. Majme jazyk L_d, t.j. všetky binárne reťazce predstavujúce TS, ktoré neakceptujú svoj vlastný kód. Potom jazyk $\overline{L_d}$ je jazyk, ktorý tvoria všetky binárne reťazce predstavujúce TS, ktoré akceptujú svoj vlastný kód.

Uzavretosť rekurzívnych jazykov vzhľadom na doplnok

Pre rekurzívne jazyky platí nasledovná veta:

Veta

Ak L je rekurzívny jazyk, potom aj jeho doplnok $\overline{L}$ je rekurzívny jazyk.

Dôkaz: Potrebujeme dokázať, že ak L má Turingov stroj M, ktorý ho akceptuje $L = L(M)$ a vždy zastaví (lebo taká je definícia rekurzívnych jazykov), potom aj doplnok $\overline{L}$ má Turingov stroj $\overline{M}$, ktorý ho akceptuje a vždy zastaví.

1. Nech M je Turingov stroj, ktorý vždy zastaví a $L = L(M)$.
2. Zostrojme Turingov stroj $\overline{M}$ z TS M nasledovným spôsobom:
   - 2.1 $\overline{M}$ prevezme z M stavy a prechody medzi nimi. Navyše:
   - 2.2 Akceptačné stavy M budú neakceptačné stavy $\overline{M}$ bez prechodov, t.j. ak $\overline{M}$ dôjde do týchto neakceptačných stavov, tak v nich zastane
   - 2.3 $\overline{M}$ bude mať 1 nový akceptačný stav r, z ktorého nie sú definované prechody.
   - 2.4 Pre každú kombináciu (neakceptačný stav,páskový symbol) v TS M takú, že v M pre ňu nie je definovaný prechod, bude v $\overline{M}$ definovaný prechod do akceptačného stavu r.

Dôkaz (pokr.) Z konštrukcie je zrejmé, že:

1. Ak M zastaví v akceptačnom stave, tak $\overline{M}$ zastaví v neakceptačnom stave.
2. Ak M zastaví v neakceptačnom stave, tak $\overline{M}$ umožní podľa dodefinovaného prechodu prechod do nového akceptačného stavu r.
3. Preto $\overline{M}$ akceptuje práve tie reťazce, ktoré M neakceptoval, a naopak.
4. A teda $L(\overline{M}) = \overline{L(M)}$ a navyše, ak M vždy zastavil, aj $\overline{M}$ vždy zastaví, teda ak L je rekurzívny jazyk, potom aj jeho doplnok $\overline{L}$ je rekurzívny jazyk.

Konštrukcia je znázornená aj na obrázku:

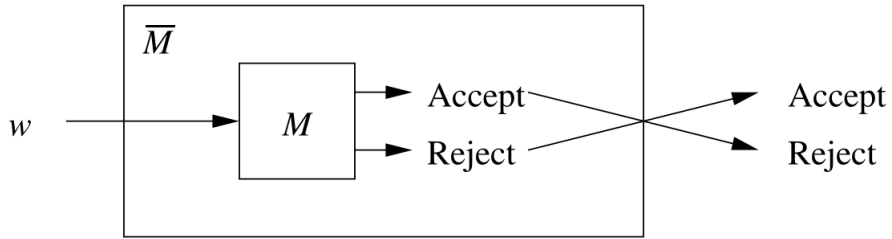


Figure: Prevzaté z [1]

Uzavretosť RE jazykov na doplnok

Veta

Nech oba jazyky, L aj jeho doplnok $\overline{L}$ sú rekurzívne vyčísliteľné. Potom L je určite rekurzívny. A z vety na slajde č. 50 vyplýva, že aj $\overline{L}$ je rekurzívny.

Dôkaz:

1. Nech platí, že L aj $\overline{L}$ sú rekurzívne vyčísliteľné. Potom musia existovať Turingove stroje M_1, M_2 také, že $L(M_1) = L, L(M_2) = \overline{L}$.
2. To znamená, že M_1 pre reťazce z jazyka L zastaví, lebo ich akceptuje. Pre reťazce, ktoré nie sú z jazyka L však zastaviť nemusí - lebo o jazyku L predpokladáme len, že je rekurzívne vyčísliteľný, t.j. nemusí byť rekurzívny. Analogicky aj M_2 zastaví určite len pre reťazce z $\overline{L}$.
3. Nech teda máme 2 Turingove stroje, M_1 pre jazyk L, M_2 pre jazyk $\overline{L}$. M_1 zastaví pre všetky reťazce z jazyka L a akceptuje ich, M_2 zastaví pre všetky reťazce z jazyka $\overline{L}$ a akceptuje ich.
4. Zostrojme Turingov stroj M, ktorý dokáže simulovať paralelne činnosť oboch M_1, M_2.

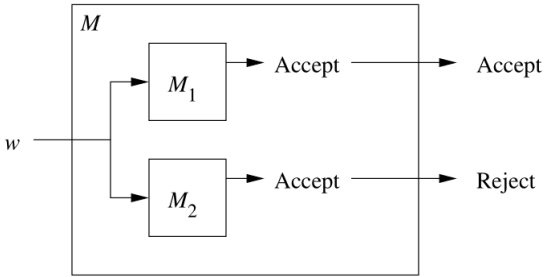


Figure: TS simulujúci súčasne M_1, M_2 [1]

Uvedený Turingov stroj bude:

- akceptovať w práve vtedy, keď ho akceptuje M_1
- neakceptovať w práve vtedy, keď ho akceptuje M_2

Turingov stroj M bude fungovať nasledovne:

- Má 2 pásky - na jednej páske simuluje pásku M_1, na druhej páske simuluje pásku M_2.
- Paralelne dokáže simulovať správanie oboch TS.
- Ak je na vstupe $w \in L$, tak potom tento reťazec akceptuje M_1. To znamená, že simulácia činnosti M_1 vie zastaviť - vtedy zároveň zastaví aj M a navyše vstup w akceptuje.
- Ak je na vstupe $w \in \overline{L}$, tak potom tento reťazec akceptuje M_2. To znamená, že simulácia činnosti M_2 vie zastaviť - vtedy zároveň zastaví aj M a vstup w neakceptuje.
- Keďže M je tým pádom Turingov stroj, ktorý pre **každý reťazec** zastaví a zároveň dokáže akceptovať práve reťazce jazyka L, znamená to, že jazyk L je rekurzívny, lebo preň existuje TS M, $L = L(M)$ a M vždy zastaví.

Zároveň z vety zo slajdu č. 50 vyplýva, že aj $\overline{L}$ je rekurzívny.

Z uvedených 2 viet vyplývajú nasledovné možnosti vzhľadom na pozíciu jazyka a jeho doplnku v triedach jazykov:

1. Aj jazyk L, aj jeho doplnok $\overline{L}$ sú rekurzívne.
2. Ani jazyk L, ani jeho doplnok $\overline{L}$ nie sú rekurzívne vyčísliteľné.
3. L je rekurzívne vyčísliteľný, ale nie je rekurzívny a jeho doplnok $\overline{L}$ nie je rekurzívne vyčísliteľný.
4. Doplnok $\overline{L}$ je rekurzívne vyčísliteľný ale nie je rekurzívny a L nie je ani rekurzívne vyčísliteľný.

Príklad

- Uvažujme jazyk L_d, t.j. diagonalizačný jazyk. O tomto jazyku sme si dokázali, že nie je rekurzívne vyčísliteľný, t.j. že preň neexistuje Turingov stroj, ktorý by ho akceptoval. Z uvedeného vyplýva, že jeho doplnok, t.j.

  $\overline{L_d}$ = množina binárnych reťazcov predstavujúcich TS, ktorý akceptuje sám seba
- alebo rovnako nie je rekurzívne vyčísliteľný,
- alebo je rekurzívne vyčísliteľný, avšak nie je rekurzívny.
- Správna je druhá možnosť, t.j. $\overline{L_d}$ je RE ale nie rekurzívny.

Univerzálny jazyk L_u

- Vráťme sa k reprezentácii Turingovho stroja ako binárneho reťazca.
- Podobne, ako vieme reprezentovať Turingov stroj M, vieme reprezentovať aj pár (M, w), t.j. Turingov stroj M s binárnou vstupnou abecedou a nejaký jeho vstup w, $w \in \{0,1\}^*$. Stačí, aby sme za binárnu reprezentáciu M vložili 111 a následne reťazec w.
- Napríklad ak by sme pre Turingov stroj zo slajdu č. 28 uvažovali vstup 1011, potom by bol TS s týmto vstupom zakódovaný ako:

010010001010011000101010010011000100100101001100010001000100101111011

- Uvažujme teraz jazyk, ktorý tvoria všetky binárne reťazce, ktoré reprezentujú pár (M, w) taký, že $w \in L(M)$, t.j. pár Turingov stroj M a slovo w z jazyka akceptovaného TS M.
- Takýto jazyk, ktorý vlastne tvoria **všetky možné Turingove stroje so všetkými svojimi akceptovanými vstupmi**, sa nazýva **univerzálny jazyk** a označuje sa L_u.
- Ukážeme, že existuje jeden Turingov stroj U, nazvaný **univerzálny Turingov stroj**, pre ktorý platí $L_u = L(U)$
- To znamená, že existuje Turingov stroj, ktorý dokáže simulovať činnosť ľubovoľného Turingovho stroja pre ľubovoľný vstup.

Univerzálny Turingov stroj U

Univerzálny Turingov stroj U:

- Pozostáva z viacerých pások: na prvej páske sú na začiatku uložené prechody M a reťazec w
- Druhá páska bude simulovať obsah pásky M, pričom bude používať rovnaký formát, v akom je M uložený v binárnom reťazci. T.j. páskový symbol X_i Turingovho stroja M bude na tejto páske stroja U reprezentovaný ako 0^i a jednotlivé páskové symboly TS M budú na tejto páske stroja U oddelené jednotkami.
- Tretia páska bude obsahovať aktuálny stav TS M, pričom stav q_i TS M bude na tejto páske reprezentovaný ako 0^i.

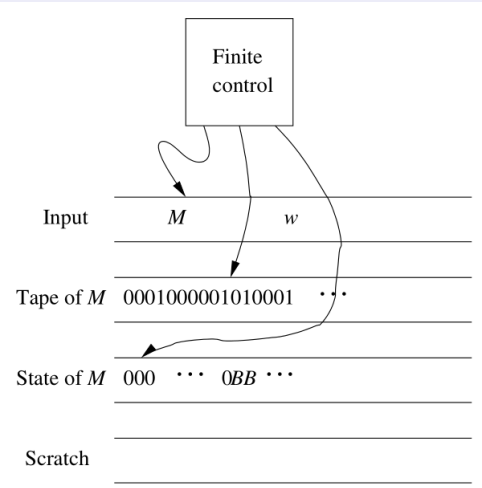


Figure: Náčrt Univerzálneho TS U [1]

Operácie U:

1. Najprv preskúma vstup, či M je valídna reprezentácia TS. Ak nie, U zastaví bez akceptácie, keďže taký binárny reťazec predstavuje TS s 1 stavom, bez prechodov, ktorý akceptuje prázdny jazyk.
2. Na druhú pásku zapíše vstup w v zakódovanom tvare. T.j. pre každú nulu slova w na pásku zapíše 10, pre každú jednotku na pásku zapíše 100. Ostatné bunky U sú prázdne - ak na ne narazí počas činnosti, tak vie, že keďže táto páska má simulovať pásku M, tak na potrebné prázdne pozície najprv zapíše 1000 - prázdny symbol simulovaného M.
3. Na tretiu pásku sa zapíše 0, t.j. reprezentácia počiatočného stavu q_1 TS M.

Operácie U:

4. Aby U simuloval činnosť M danú prechodom $0^i10^j10^k10^l10^m$, tak na tretej páske musí byť 0^i, na druhej páske musí byť od aktuálnej pozície čítacej hlavy 0^j. Následne sa vykoná:
   - 4.1 Zmena obsahu pásky č. 3 na 0^k - najprv sa z pásky zmaže 0^i a následne sa tam nakopíruje z pásky č. 1 0^k, t.j. simuluje sa zmena stavu M
   - 4.2 Na páske č. 2 sa nahradí 0^j reťazcom 0^l, t.j. simuluje sa zmena obsahu pásky M. Ak je potrebné, aby sa dáta na páske posunuli (ak $j \neq l$), tak sa to vykoná pomocou "Scratch" pásky
   - 4.3 Posun čítacej hlavy na páske č. 2 na najbližšiu jednotku vľavo/vpravo, aby sa simuloval posun hlavy na páske M vľavo/vpravo.
5. Ak M nemá prechody pre aktuálny stav a vstupný symbol, tak M by sa zastavil - a rovnako sa zastaví aj U.
6. Ak by M prešiel do akceptačného stavu, U vstup akceptuje.

Tým U dokáže simulovať správanie M pre vstup w. U akceptuje binárny reťazec kódujúci (M, w) vtedy a len vtedy, ak M akceptuje w.

Jazyk L_u je nerozhodnuteľný

- Uvedený univerzálny jazyk L_u je určite rekurzívne vyčísliteľný, pretože sme si práve uviedli, ako vyzerá tzv. univerzálny Turingov stroj U, ktorý ho akceptuje.
- Dôležitou vlastnosťou jazyka L_u je, že **nie je rekurzívny**, t.j. univerzálny Turingov stroj nemusí zastaviť pre ľubovoľný vstup.
- To znamená, že príslušnosť do jazyka L_u je nerozhodnuteľný problém.
- Táto vlastnosť je veľmi dôležitá, lebo sa pomocou nej dá dokázať, že nejaký iný problém P je tiež nerozhodnuteľný.

Veta

Jazyk L_u je rekurzívne vyčísliteľný jazyk. Zároveň tento jazyk nie je rekurzívny.

Dôkaz:

- To, že L_u je RE jazyk, sme dokázali pred chvíľou - tým, že sme našli TS, ktorý ho akceptuje.
- Teraz chceme dokázať, že nie je rekurzívny - t.j. že neexistuje taký TS, ktorý by ho akceptoval a súčasne zastavil pre ľubovoľný vstup.
- Dôkaz vykonáme sporom.

Predpoklad: Nech L_u je rekurzívny.

1. Potom podľa vety zo slajdu č. 50 aj jazyk $\overline{L_u}$ je rekurzívny.
2. Ak by jazyk $\overline{L_u}$ bol rekurzívny, znamená to, že existuje TS M taký, že $\overline{L_u} = L(M)$.
3. Vezmime teraz Turingov stroj M, ktorý údajne existuje a vie akceptovať jazyk $\overline{L_u}$ a vyrobme z neho TS M' nasledovným spôsobom:

Nech M je TS akceptujúci $\overline{L_u}$ a nech M' je Turingov stroj, ktorý robí nasledovné:

1. Vezme vstup w a vyrobí z neho reťazec $w111w$.
2. Následne TS M' simuluje činnosť TS M pre reťazec $w111w$.
3. Ak M bude reťazec $w111w$ akceptovať, potom aj M' bude reťazec w akceptovať.
4. Ak M nebude reťazec $w111w$ akceptovať, potom aj M' reťazec w akceptovať nebude.
5. Keďže M je TS ktorý vždy zastaví a zároveň akceptuje jazyk $\overline{L_u}$, tak ak reťazec $w111w$
   - **akceptuje** - potom w predstavuje Turingov stroj, ktorý neakceptuje sám seba
   - **neakceptuje** - potom w predstavuje Turingob stroj, ktorý akceptuje sám seba
6. To znamená, že M' by bol Turingov stroj, ktorý akceptuje práve také reťazce, ktoré predstavujú Turingove stroje, ktoré neakceptujú samých seba - t.j. M' by bol Turingov stroj, ktorý by **akceptoval jazyk** L_d

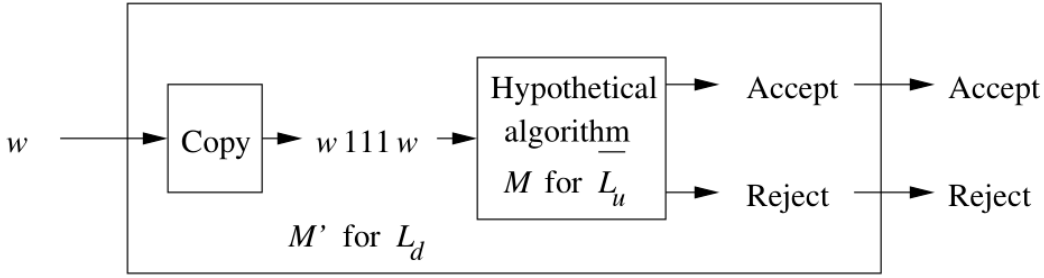


Figure: Znázornenie TS M' [1]

Dôkaz(pokr.)

- Avšak, ako vieme, taký Turingov stroj **nemôže existovať!**
- Preto náš predpoklad, že L_u je rekurzívny jazyk, bol zlý.
- Tým sme dokázali, že L_u určite **nie je** rekurzívny jazyk.

- Keďže L_u je síce RE ale nie rekurzívny jazyk, tak to znamená, že ako rozhodovací problém patrí k **nerozhodnuteľným** problémom.
- Teda neexistuje algoritmus, ktorý by v konečnom čase dokázal povedať, či (M, w) predstavuje / nepredstavuje pár Turingov stroj / vstup taký, že $w \in L(M)$.
- Inými slovami, "membership problem" $w \in L(M)$ je pre **rekurzívne vyčísliteľné jazyky nerozhodnuteľný problém**.
- To je samozrejme logické, keďže ako sme sa bavili, Turingov stroj vo všeobecnosti vie zastaviť len, ak vstup akceptuje, ale ak vstup neakceptuje, tak nemusí poskytnúť odpoveď v konečnom čase.
- Pre porovnanie, pre bezkontextové a regulárne (a dokonca aj pre rekurzívne) jazyky to rozhodnuteľný problém je, keďže ich príslušné výpočtové modely **vždy** zastavia.

Halting problém

S uvedeným súvisí aj veľmi dôležitý problém v teoretickej informatike - tzv. **problém zastavenia (halting problém)**:

Nech M je Turingov stroj a nech w je reťazec, predstavujúci jeho vstup. Viete rozhodnúť, či sa Turingov stroj M zastaví alebo či pobeží bez zastavenia navždy?

Tento problém je veľmi podobný nášmu problému *hello, world* z predchádzajúcej prednášky.

Halting problém

- Nech $H(M)$ je množina reťazcov w, pre ktoré sa Turingov stroj M zastaví, bez ohľadu na to, či w akceptuje alebo nie.
- Potom halting problém je zároveň aj jazyk, ktorý tvoria také páry (M, w), pre ktoré platí, že $w \in H(M)$. Znovu, (M, w) sú binárna reprezentácia TS a jeho vstupu.
- Takýto jazyk je, rovnako ako L_u rekurzívne vyčísliteľný a zároveň nie rekurzívny, t.j. predstavuje **nerozhodnuteľný problém**.
- Veľmi pekné a obľúbené video o probléme zastavenia sa dá nájsť tu: `https://www.youtube.com/watch?v=92WHN-pAFCs`

Použitá literatúra

1 Hopcroft, Motwani, Ullman - Introduction to Automata Theory, Languages and Computations, 3rd Ed.