Jazyky rekurzívně vyčíslitelné a jazyky rekurzívní

Rekurzívní vyčíslitelnost a rekurzívnost

❖ Turingův stroj se nazývá úplný (*total*), právě když se pro každý vstup zastaví.

❖ *Poznámka*: Nedeterministický Turingův stroj je úplný, právě když pro každý vstup je každá výpočetní větev konečná (tj. pro každý vstup vždy zastaví).

Definice 8.1 Jazyk $L \subseteq \Sigma^*$ se nazývá

- rekurzívně vyčíslitelný, jestliže $L = L(M)$ pro nějaký TS M,
- rekurzívní, jestliže $L = L(M)$ pro nějaký *úplný* TS M.

❖ Je-li M úplný Turingův stroj, pak říkáme, že M rozhoduje jazyk $L(M)$.

❖ Ke každému rekurzívnímu jazyku existuje TS, který ho rozhoduje, tj. zastaví pro každé vstupní slovo – tento TS lze samozřejmě upravit tak, aby pro každý řetězec z daného jazyka zastavil s páskou $\Delta Y \Delta\Delta$... a jinak zastavil s páskou $\Delta N \Delta\Delta$....

❖ TS přijímající rekurzívně vyčíslitelný jazyk L zastaví pro každé $w \in L$, ovšem pro $w \notin L$ může zastavit, ale také může donekonečna cyklit.

Rozhodovací problémy

❖ Rozhodovací problém (*decision problem*) P může být chápán jako funkce f_P s oborem hodnot $\{true, false\}$.

❖ Rozhodovací problém je obvykle specifikován:

- definičním oborem A_P reprezentujícím množinu možných instancí problému (vstupů) a
- podmnožinou $B_P \subseteq A_P$, $B_P = \{p \mid f_P(p) = true\}$ instancí, pro které je hodnota f_P rovna $true$.

❖ V teorii formálních jazyků používáme ke kódování jednotlivých instancí problémů řetězce nad vhodnou abecedou Σ. Pak je rozhodovací problém P přirozeně specifikován jazykem $L_p = \{w \in \Sigma^* \mid w = code(p), p \in B_P\}$, kde $code : A_P \rightarrow \Sigma^*$ je injektivní funkce, která přiřazuje instancím problému příslušný řetězec (nezávisle na f_P).

Příklad 8.1 Příklady rozhodovacích problémů:

- P_1 – orientovaný graf je silně souvislý.
- P_2 – dvě bezkontextové gramatiky jsou ekvivalentní,
- P_3 – n je prvočíslo.

❖ *Poznámka*: Dále budeme o rozhodovacích problémech hovořit jednoduše jako o problémech.

Rozhodování problémů TS

Definice 8.2 Nechť P je problém specifikovaný jazykem L_P nad Σ. Problém P nazveme:

- rozhodnutelný, pokud L_P je rekurzívní jazyk, tj. existuje TS, který L_P rozhoduje (přijme každý řetězec $w \in L_P$, a zamítne každý řetězec $w \in \Sigma^* \setminus L_P$),
- nerozhodnutelný, když není rozhodnutelný, a
- částečně rozhodnutelný, jestliže L_P je rekurzívně vyčíslitelný jazyk.

❖ ***Poznámka:*** Z definice 8.2 plyne, že každý rozhodnutelný problém je současně částečně rozhodnutelný, ale některé nerozhodnutelné problémy nejsou ani částečně rozhodnutelné.

TS a jazyky typu 0

Jazyky přijímané TS jsou typu 0

❖ Pro zápis konfigurace TS v řídícím stavu q a s konfigurací pásky $\Delta x \underline{y} z \Delta ...$ zavedeme konvenci $[\Delta x q y z \Delta ...]$.

Věta 8.1 Každý rekurzívně vyčíslitelný jazyk je jazykem typu 0.

Důkaz. * Nechť $L = L(M)$ pro nějaký TS $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$. Sestrojíme gramatiku $G = (N, \Sigma, P, S)$ typu 0 takovou, že $L(G) = L(M)$. Gramatika G dovoluje vytvářet derivace odpovídající reverzi posloupnosti konfigurací TS M při přijetí $w \in L(M)$:

1. $N = \{S\} \cup Q \cup (\Gamma \setminus \Sigma) \cup \{[,]\}$ (množiny jsou po dvou disjunktní).
2. P je nejmenší množina obsahující následující pravidla:
   (a) $S \rightarrow [q_f \Delta Y \Delta]$,
   (b) $\Delta] \rightarrow \Delta\Delta]$ – *doplnění* Δ,
   (c) $qy \rightarrow px$, jestliže $\delta(p, x) = (q, y)$,
   (d) $xq \rightarrow px$, jestliže $\delta(p, x) = (q, R)$,
   (e) $qyx \rightarrow ypx$ pro každé $y \in \Gamma$, jestliže $\delta(p, x) = (q, L)$,
   (f) $[q_0\Delta \rightarrow \varepsilon$, $\Delta\Delta] \rightarrow \Delta]$, $\Delta] \rightarrow \varepsilon$ – *zajištění* $[q_0 \Delta w \Delta ... \Delta] \overset{+}{\underset{G}{\Rightarrow}} w$.

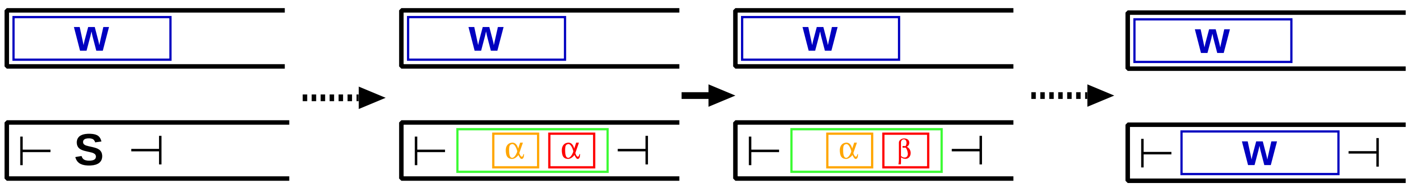
Snadno se nyní nahlédne, že $w \in L(M)$ právě tehdy, když existuje derivace $S \Rightarrow_G [q_F \Delta Y \Delta] \Rightarrow_G ... \Rightarrow_G [q_0 \Delta w \Delta ...] \Rightarrow_G ... \Rightarrow_G w$, a že $L(G) = L(M)$. □ *

Jazyky typu 0 jsou přijímány TS

Věta 8.2 Každý jazyk typu 0 je přijímán nějakým TS (tj. je rekurzívně vyčíslitelný).

Důkaz. Nechť $L = L(G)$ pro $G = (N, \Sigma, P, S)$ je jazykem typu 0. Sestrojíme nedeterministický dvoupáskový TS M takový, že $L(G) = L(M)$:

- 1. páska obsahuje přijímaný vstupní řetězec w.
- Na 2. pásce se M pokouší pomocí simulace použití přepisovacích pravidel $(\alpha \rightarrow \beta) \in P$ vytvořit derivaci w:



1. Stroj nejprve umístí na 2. pásku symbol S.
2. Stroj opakovaně simuluje na 2. pásce provádění pravidel $(\alpha \rightarrow \beta) \in P$. Nedeterministicky zvolí pravidlo a také výskyt α na pásce. Při přepisu α na β, $|\alpha| \neq |\beta|$, může využít posuv části užitečného obsahu pásky vlevo či vpravo.
3. Stroj srovná finální obsah 2. pásky s 1. páskou. Shodují-li se, zastaví přechodem do q_F. Jinak posouvá hlavu doleva až do abnormálního zastavení.

Snadno se nyní nahlédne, že skutečně $L(G) = L(M)$. Navíc lze M podobně jako u vícepáskových DTS převést na jednopáskový NTS a ten dále na jednopáskový DTS. $\square$

Jazyky typu 0 = jazyky přijímané TS

Věta 8.3 ***Třída jazyků přijímaných TS (neboli jazyků rekurzívně vyčíslitelných) je shodná se třídou jazyků typu 0.***

Důkaz. Důsledek dvou předchozích vět. □

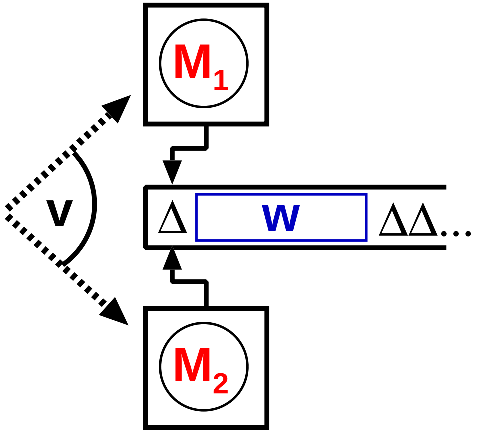
Vlastnosti jazyků rekurzívních a rekurzívně vyčíslitelných

Uzavřenost vůči $\cup$, $\cap$, . a $*$

Věta 8.4 Třídy rekurzívních a rekurzívně vyčíslitelných jazyků jsou uzavřeny vůči operacím $\cup$, $\cap$, . a $*$.

Důkaz. Nechť L_1, L_2 jsou jazyky přijímané TS M_1, M_2. Zřejmě můžeme předpokládat, že množiny stavů TS M_1, M_2 jsou disjunktní.

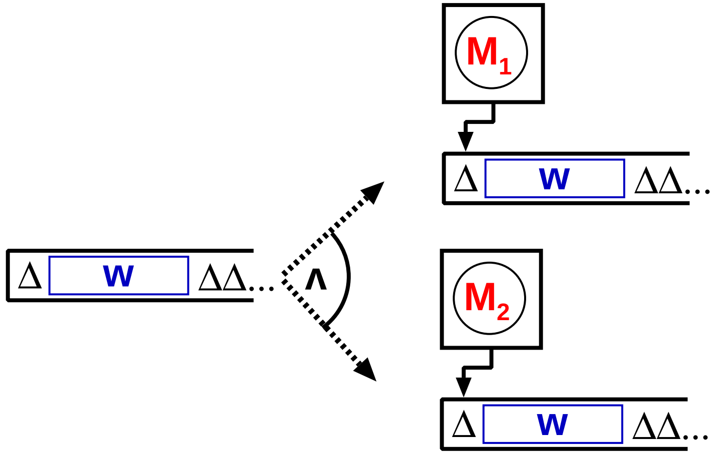
- NTS $M_{L_1 \cup L_2}$, $L(M_{L_1 \cup L_2}) = L_1 \cup L_2$, sestrojíme tak, že sjednotíme po složkách stroje M_1 a M_2, zavedeme nový počáteční stav, z něj nedeterministické přechody přes Δ/Δ do obou původních počátečních stavů a sloučíme původní koncové stavy do jediného nového koncového stavu.



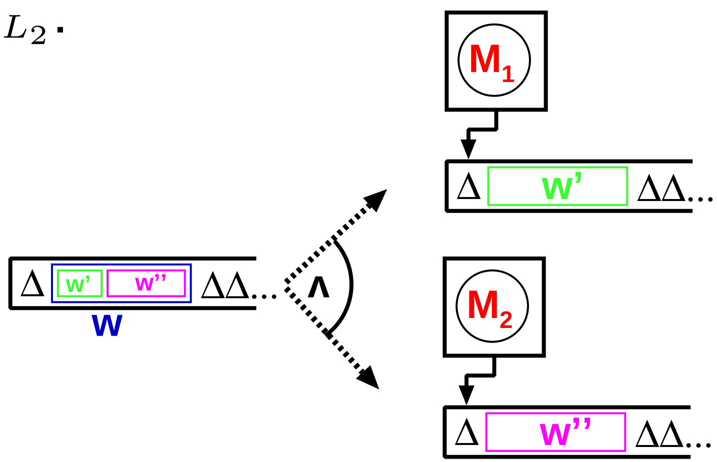
Důkaz pokračuje dále.

Pokračování důkazu.

- Třípáskový TS $M_{L_1 \cap L_2}$, $L(M_{L_1 \cap L_2}) = L_1 \cap L_2$, okopíruje vstup z první pásky na druhou, na ní simuluje stroj M_1, pokud ten přijme, okopíruje vstup z první pásky na třetí, na ní simuluje stroj M_2, pokud i ten přijme, přijme i stroj $M_{L_1 \cap L_2}$.



- Třípáskový NTS $M_{L_1.L_2}$, $L(M_{L_1.L_2}) = L_1.L_2$, okopíruje nedeterministicky zvolený prefix vstupu z první pásky na druhou, na ní simuluje stroj M_1, pokud ten přijme, okopíruje zbytek vstupu z první pásky na třetí, na ní simuluje stroj M_2, pokud i ten přijme, přijme i stroj $M_{L_1.L_2}$.



Důkaz pokračuje dále.

Pokračování důkazu.

- Dvoupáskový NTS $M_{L_1^*}$, $L(M_{L_1^*}) = L_1^*$, je zobecněním předchozího stroje: po částech kopíruje vstup z první pásky na druhou a na ní simuluje opakovaně stroj M_1. Obsah druhé pásky má ohraničený speciálními značkami a po každé simulaci stroje M_1 ho smaže. Umožňuje samozřejmě posuv pravé značky dále doprava při nedostatku místa.

Jsou-li stroje M_1 a M_2 úplné, je možné vybudovat stroje podle výše uvedených pravidel také jako úplné (u $M_{L_1 \cup L_2}$, $M_{L_1 \cap L_2}$, $M_{L_1 . L_2}$ je to okamžité, u $M_{L_1^*}$ nepřipustíme načítání prázdného podřetězce vstupu z 1. na 2. pásku – pouze umožníme jednorázově přijmout prázdný vstup). To dokazuje uzavřenost vůči uvedeným operacím také u rekurzívních jazyků.

$\square$

(Ne)uzavřenost vůči komplementu

Věta 8.5 Třída rekurzívních jazyků je uzavřena vůči komplementu.

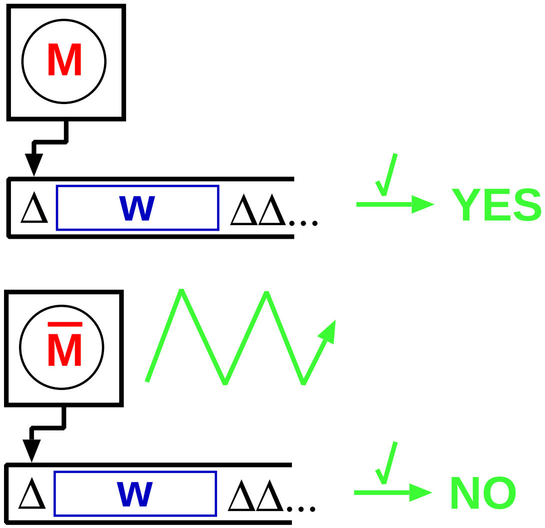
Důkaz. TS M přijímající rekurzívní jazyk L vždy zastaví. Snadno upravíme M na M', který při nepřijetí řetězce vždy přejde do unikátního stavu q_{reject}. TS $\overline{M}$, $L(\overline{M}) = \overline{L}$, snadno dostaneme z M' záměnou q_F a q_{reject}. □

❖ Třída rekurzívně vyčíslitelných jazyků není uzavřena vůči komplementu!

- Výše uvedené konstrukce nelze užít – cyklení zůstane zachováno.
- Důkaz neuzavřenosti bude uveden v dalších přednáškách.

Věta 8.6 Jsou-li L i $\overline{L}$ rekurzívně vyčíslitelné, pak jsou oba rekurzívní.

Důkaz.
Mějme M, $L(M) = L$, a $\overline{M}$, $L(\overline{M}) = \overline{L}$. Úplný TS přijímající L sestrojíme takto:



- Použijeme dvě pásky. Na jedné budeme simulovat M, na druhé $\overline{M}$. Simulace se bude provádět proložené krok po kroku: krok M, krok $\overline{M}$, krok M, ...
- Přijmeme, právě když by přijal M, zamítneme abnormálním zastavením, právě když by přijal $\overline{M}$. Jedna z těchto situací určitě nastane v konečném počtu kroků.

Existence úplného TS pro $\overline{L}$ plyne z uzavřenosti rekurzívních jazyků vůči komplementu.

□

❖ Důsledkem výše uvedených vět je mj. to, že pro L a $\overline{L}$ musí vždy nastat jedna z následujících situací:

- L i $\overline{L}$ jsou rekurzívní,
- L ani $\overline{L}$ nejsou rekurzívně vyčíslitelné,
- jeden z těchto jazyků je rekurzívně vyčíslitelný, ale ne rekurzívní, druhý není rekurzívně vyčíslitelný.

Lineárně omezené automaty

Lineárně omezené automaty

- Lineárně omezený automat (LOA) je nedeterministický TS, který nikdy neopustí tu část pásky, na níž je zapsán jeho vstup.

- Formálně můžeme LOA definovat jako NTS, který má v Γ speciální symbol, kterým unikátně označujeme pravý konec vstupu na pásce, přičemž tento symbol není možné přepsat, ani z něj provést posun doprava.

- Deterministický LOA můžeme přirozeně definovat jako (deterministický) TS, který nikdy neopustí část pásky se zapsaným vstupem.

- Není známo, zda deterministický LOA je či není striktně slabší než LOA.

LOA a kontextové jazyky

Věta 8.7 Třída jazyků, kterou lze generovat kontextovými gramatikami, odpovídá třídě jazyků, které lze přijímat LOA.

Důkaz.

- Uvážíme definici kontextových gramatik jako gramatik s pravidly v podobě $\alpha \to \beta$, kde $|\alpha| \leq |\beta|$, nebo $S \to \varepsilon$.
- *LOA $\longrightarrow$ G1:
  - Použijeme podobnou konstrukci jako u TS $\longrightarrow$ G0.
  - Na počátku vygenerujeme příslušný pracovní prostor, který se pak již nebude měnit: odpadá nekontextové pravidlo $\Delta\Delta] \to \Delta]$.
  - Užití nekontextových pravidel $[q_0\Delta \to \varepsilon$ a $\Delta] \to \varepsilon$ obejdeme (1) zavedením zvláštních koncových nonterminálů integrujících původní informaci a příznak, že se jedná o první/poslední symbol a (2) integrací symbolu reprezentujícího řídící stav a pozici hlavy s následujícím páskovým symbolem.*
- G1 $\longrightarrow$ LOA:
  - Použijeme podobnou konstrukci jako u G0 $\longrightarrow$ TS s tím, že nepovolíme, aby rozsah druhé pásky někdy překročil rozsah první pásky.

□

Kontextové a rekurzívní jazyky

Věta 8.8 Každý kontextový jazyk je rekurzívní.

Důkaz. (Idea)

- Počet konfigurací, které se mohou objevit při přijímání w příslušným LOA M je vzhledem k nemožnosti zvětšovat pracovní prostor pásky konečný: lze shora ohraničit funkcí c^n pro vhodnou konstantu c – exponenciála plyne z nutnosti uvažovat výskyt všechny možných symbolů na všech místech pásky.
- Pro zápis libovolného čísla z intervalu $0, ..., c^n - 1$ nikdy nebude třeba více než n symbolů, užijeme-li c-ární soustavu.
- Můžeme zkonstruovat úplný LOA ekvivalentní s M, který bude mít každý symbol na pásce strukturovaný jako dvojici:
  - S využitím 1. složek těchto dvojic simulujeme M.
  - V 2. složkách počítáme počet kroků; dojde-li k přetečení, odmítneme vstup.

$\square$

Věta 8.9 *Ne* každý rekurzívní jazyk je kontextový.

Důkaz. (Idea) Lze užít techniku diagonalizace prezentovanou dále . $\square$

Vlastnosti kontextových jazyků

Věta 8.10 Třída kontextových jazyků je uzavřena vůči operacím $\cup$, $\cap$, ., $*$ a komplementu.

Důkaz.

- Uzavřenost vůči $\cup$, $\cap$, . a $*$ lze ukázat stejně jako u rekurzívně spočetných jazyků.
- Důkaz uzavřenosti vůči komplementu je značně komplikovaný (všimněme si, že LOA je *nedeterministický* a nelze tudíž užít konstrukce použité u rekurzívních jazyků) – zájemci naleznou důkaz v doporučené literatuře.

$\square$

❖ Poznamenejme, že již víme, že u kontextových jazyků

- lze rozhodovat členství věty do jazyka (rekurzívnost) a
- nelze rozhodovat inkluzi jazyků (neplatí ani pro bezkontextové jazyky).

❖ Dále lze ukázat, že pro kontextové jazyky nelze rozhodovat prázdnost jazyka (užije se redukce z Postova problému přiřazení – viz další přednášky).