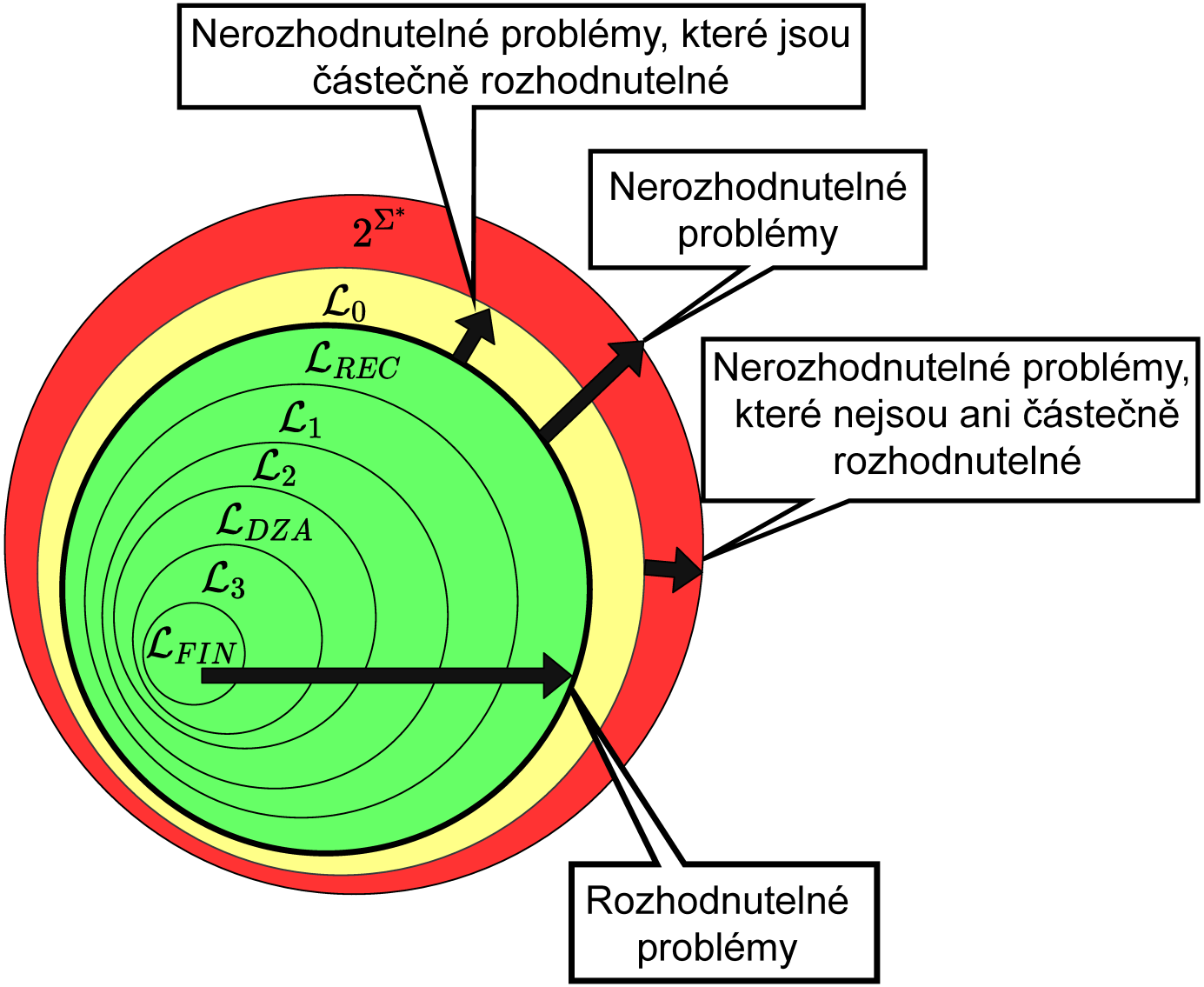
Meze rozhodnutelnosti

Existují jazyky (problémy), jež nejsou rekurzivně vyčíslitelné (částečně rozhodnutelné)?

Které jazyky, resp. problémy, nejsou rekurzívní (rozhodnutelné)?

Hierarchie jazyků/problémů



❖ **Ekvivalence tříd:** $\mathcal{L}_3 = \mathcal{L}_{KA} \quad \mathcal{L}_2 = \mathcal{L}_{ZA} \quad \mathcal{L}_1 = \mathcal{L}_{LOA} \quad \mathcal{L}_0 = \mathcal{L}_{TS} = \mathcal{L}_{RE}$

Jazyky mimo třídu 0

Existence jazyků mimo třídu 0

Věta 9.1 Pro každou abecedu Σ existuje jazyk nad Σ, který není typu 0 (tj. rekurzívně vyčíslitelný).

Důkaz.

1. Libovolný jazyk typu 0 nad Σ může být přijat TS s $\Gamma = \Sigma \cup \{\Delta\}$: Pokud M používá více symbolů, můžeme je zakódovat jako jisté posloupnosti symbolů ze $\Sigma \cup \{\Delta\}$ a sestrojit TS M', který simuluje M.
2. Nyní můžeme snadno systematicky vypisovat všechny TS s $\Gamma = \Sigma \cup \{\Delta\}$.
   Začneme stroji se dvěma stavy, pak se třemi stavy, ...
   Závěr: Množina všech takových strojů a tedy i jazyků typu 0 je spočetná.
3. Množina Σ^* ale obsahuje nekonečně mnoho řetězců a proto je množina 2^{Σ^*} zahrnující všechny jazyky nespočetná – důkaz viz další strana.
4. Z rozdílnosti mohutností spočetných a nespočetných množin plyne platnost uvedené věty.

□

Lemma 9.1 Pro neprázdné, konečné Σ je množina 2^{Σ^*} nespočetná.

Důkaz. Důkaz provedeme tzv. diagonalizací (poprvé použitou Cantorem při důkazu rozdílné mohutnosti $\mathbb{N}$ a $\mathbb{R}$).

- Předpokládejme, že 2^{Σ^*} je spočetná. Pak dle definice spočetnosti existuje bijekce $f : \mathbb{N} \longleftrightarrow 2^{\Sigma^*}$.
- Uspořádejme Σ^* do nějaké posloupnosti $w_1, w_2, w_3, ...$, např. ε, x, y, xx, xy, yx, yy, xxx, ... pro $\Sigma = \{x, y\}$. Nyní můžeme f zobrazit nekonečnou maticí:

	w_0	w_1	w_2	...	w_i	...
$L_0 = f(0)$	a_{00}	a_{01}	a_{02}	...	a_{0i}	...
$L_1 = f(1)$	a_{10}	a_{11}	a_{12}	...	a_{1i}	...
$L_2 = f(2)$	a_{20}	a_{21}	a_{22}	...	a_{2i}	...
...						

, kde $a_{ij} = \begin{cases} 0, \text{ jestliže } w_j \notin L_i, \\ 1, \text{ jestliže } w_j \in L_i. \end{cases}$

- Uvažujme jazyk $\overline{L} = \{w_i \mid a_{ii} = 0\}$. $\overline{L}$ se liší od každého jazyka $L_i = f(i)$, $i \in \mathbb{N}$:
  - je-li $a_{ii} = 0$, pak w_i patří do jazyka,
  - je-li $a_{ii} = 1$, pak w_i nepatří do jazyka.
- Současně ale $\overline{L} \in 2^{\Sigma^*}$, f tudíž není surjektivní, což je spor.

□

Problém zastavení

Problém zastavení TS

Věta 9.2 **Problém zastavení TS (Halting Problem)**, kdy nás zajímá, zda daný TS M pro danou vstupní větu w zastaví, **není rozhodnutelný**, ale je **částečně rozhodnutelný**.

Důkaz.

- Problému zastavení odpovídá rozhodování jazyka $HP = \{\langle M\rangle\#\langle w\rangle \mid M \text{ zastaví při } w\}$, kde $\langle M\rangle$ je kód TS M a $\langle w\rangle$ je kód w.
- Částečnou rozhodnutelnost ukážeme snadno použitím modifikovaného univerzálního TS T_U, který zastaví přijetím vstupu $\langle M\rangle\#\langle w\rangle$ právě tehdy, když M zastaví při w – modifikace spočívá v převedení abnormálního zastavení při simulaci na zastavení přechodem do q_F.
- Nerozhodnutelnost ukážeme pomocí diagonalizace:
  1. Pro $x \in \{0,1\}^*$, nechť M_x je TS s kódem x, je-li x legální kód TS. Jinak ztotožníme M_x s pevně zvoleným TS, např. TS, který pro libovolný vstup okamžitě zastaví.
  2. Můžeme nyní sestavit posloupnost $M_\varepsilon, M_0, M_1, M_{00}, M_{01}, M_{10}, M_{11}, M_{000}, \ldots$ zahrnující všechny TS nad $\Sigma = \{0,1\}$ indexované řetězci z $\{0,1\}^*$.

Důkaz pokračuje dále.

Pokračování důkazu.

3. Uvažme nekonečnou matici

	ε	0	1	00	01	10	...
M_ε	$H_{M_\varepsilon,\varepsilon}$	$H_{M_\varepsilon,0}$	$H_{M_\varepsilon,1}$	$H_{M_\varepsilon,00}$	$H_{M_\varepsilon,01}$	...	
M_0	$H_{M_0,\varepsilon}$	$H_{M_0,0}$	$H_{M_0,1}$	$H_{M_0,00}$	$H_{M_0,01}$	...	
M_1	$H_{M_1,\varepsilon}$	$H_{M_1,0}$	$H_{M_1,1}$	$H_{M_1,00}$	$H_{M_1,01}$	...	
M_{00}	$H_{M_{00},\varepsilon}$	$H_{M_{00},0}$	$H_{M_{00},1}$	$H_{M_{00},00}$	$H_{M_{00},01}$	...	
M_{01}	$H_{M_{01},\varepsilon}$	$H_{M_{01},0}$	$H_{M_{01},1}$	$H_{M_{01},00}$	$H_{M_{01},01}$	...	
...							

kde $H_{M_x,y} = \begin{cases} \textbf{C}, \text{ jestliže } M_x \text{ cyklí na } y, \\ \textbf{Z}, \text{ jestliže } M_x \text{ zastaví na } y. \end{cases}$

4. Předpokládejme, že existuje *úplný* TS K přijímající jazyk HP, tj. K pro vstup $\langle M \rangle \# \langle w \rangle$
   - zastaví normálně (přijme) právě tehdy, když M zastaví na w,
   - zastaví abnormálně (odmítne) právě tehdy, když M cyklí na w.
5. Sestavíme TS N, který pro vstup $x \in \{0,1\}^*$:
   - Sestaví M_x z x a zapíše $\langle M_x \rangle \# x$ na svou pásku.
   - Simuluje K na $\langle M_x \rangle \# x$, přijme, pokud K odmítne, a přejde do nekonečného cyklu, pokud K přijme.

Důkaz pokračuje dále.

Pokračování důkazu.

Všimněme si, že N v podstatě komplementuje diagonálu uvedené matice:

	ε	0	1	00	01	10	...
M_ε	$H_{M_\varepsilon,\varepsilon}$	$H_{M_\varepsilon,0}$	$H_{M_\varepsilon,1}$	$H_{M_\varepsilon,00}$	$H_{M_\varepsilon,01}$	...	
M_0	$H_{M_0,\varepsilon}$	$H_{M_0,0}$	$H_{M_0,1}$	$H_{M_0,00}$	$H_{M_0,01}$	...	
M_1	$H_{M_1,\varepsilon}$	$H_{M_1,0}$	$H_{M_1,1}$	$H_{M_1,00}$	$H_{M_1,01}$	...	
M_{00}	$H_{M_{00},\varepsilon}$	$H_{M_{00},0}$	$H_{M_{00},1}$	$H_{M_{00},00}$	$H_{M_{00},01}$	...	
M_{01}	$H_{M_{01},\varepsilon}$	$H_{M_{01},0}$	$H_{M_{01},1}$	$H_{M_{01},00}$	$H_{M_{01},01}$	...	
...							

6. Dostáváme, že

   N zastaví na x $\Leftrightarrow$ K odmítne $\langle M_x\rangle\#\langle x\rangle$ (definice N)

   $\Leftrightarrow$ M_x cyklí na x (předpoklad o K).

7. To ale znamená, že N se liší od každého M_x alespoň na jednom řetězci – konkrétně x. Což je ovšem spor s tím, že posloupnost $M_\varepsilon, M_0, M_1, M_{00}, M_{01}, M_{10}, M_{11}, M_{000}, \ldots$ zahrnuje všechny TS nad $\Sigma = \{0,1\}$. Tento spor plyne z předpokladu, že existuje TS K, který pro daný TS M a daný vstup x určí (rozhodne), zda M zastaví na x, či nikoliv.

□

❖ Ukázali jsme, že problém zastavení TS je částečně rozhodnutelný a tedy jazyk HP rekurzívně vyčíslitelný. Z věty 8.6 pak plyne, že **komplement problému zastavení** není ani částečně rozhodnutelný a jazyk co-$HP = \{\langle M \rangle \# \langle w \rangle \mid M$ nezastaví při $w\}$ je příkladem jazyka, jenž není ani rekurzívně vyčíslitelný.

S dalším příkladem takového jazyka se seznámíme v následujícím problému.

Redukce

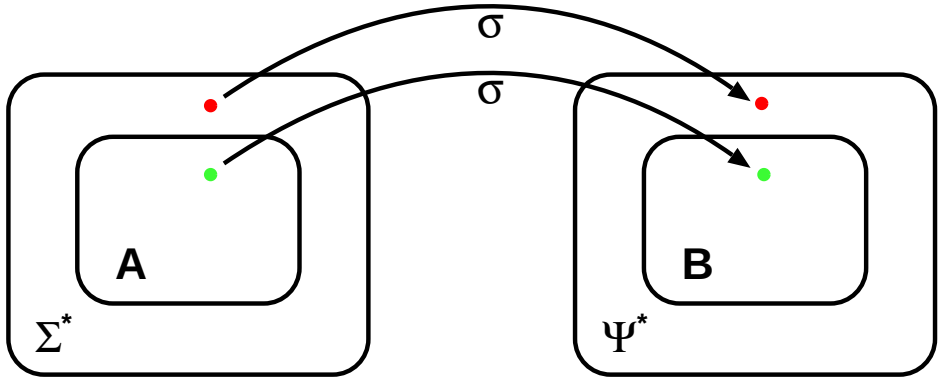
Důkaz nerozhodnutelnosti redukcí

❖ Technika redukce patří spolu s diagonalizací k nejpoužívanějším technikám důkazu, že nějaký problém není rozhodnutelný (částečně rozhodnutelný) – neboli, že určitý jazyk není rekurzívní (rekurzívně vyčíslitelný):

- víme, že jazyk A není rekurzívní (rekurzívně vyčíslitelný),
- zkoumáme jazyk B,
- ukážeme, že A lze úplným TS převést (redukovat) na B,
- to ale znamená, že B rovněž není rekurzívní (rekurzívně vyčíslitelný) – jinak by šlo použít úplný TS (ne-úplný TS) přijímající B a příslušné redukce k sestavení úplného TS (ne-úplného TS) přijímajícího A, což by byl spor.

❖ Argumentace výše samozřejmě ukazuje, že redukci lze použít i při dokazování, že určitý problém je rekurzívní (částečně rekurzívní).

Definice 9.1 Nechť A, B jsou jazyky, $A \subseteq \Sigma^*$, $B \subseteq \Psi^*$. Redukce jazyka A na jazyk B je totální, rekurzívně vyčíslitelná funkce $\sigma : \Sigma^* \to \Psi^*$ taková, že
$\forall w \in \Sigma^*.w \in A \Leftrightarrow \sigma(w) \in B$.



❖ Existuje-li redukce jazyka A na B, říkáme, že A je redukovatelný na B, což značíme $A \leq B$.

Věta 9.3 Nechť $A \leq B$.

1. Není-li jazyk A rekurzívně vyčíslitelný, pak ani jazyk B není rekurzívně vyčíslitelný.
2. Není-li jazyk A rekurzívní, pak ani jazyk B není rekurzívní.

$\overline{1}$. Je-li jazyk B rekurzívně vyčíslitelný, pak i jazyk A je rekurzívně vyčíslitelný.

$\overline{2}$. Je-li jazyk B rekurzívní, pak i jazyk A je rekurzívní.

Důkaz. Dokážeme, že pokud $A \leq B$, pak $(\overline{1})$ je-li jazyk B rekurzívně vyčíslitelný, pak i jazyk A je rekurzívně vyčíslitelný:

- Nechť M_R je úplný TS počítající redukci σ z A na B a M_B je TS přijímající B.
- Sestrojíme M_A přijímající A:
  1. M_A simuluje M_R na vstupu w, což transformuje obsah pásky na $\sigma(w)$.
  2. M_A simuluje výpočet M_B na $\sigma(w)$.
  3. Pokud M_B zastaví a přijme, M_A rovněž zastaví a přijme, jinak M_A zastaví abnormálně nebo cyklí.
- Zřejmě platí:

$$\begin{aligned} M_A \text{ přijme } w &\Leftrightarrow M_B \text{ přijme } \sigma(w) \\ &\Leftrightarrow \sigma(w) \in B \\ &\Leftrightarrow w \in A \qquad \text{(definice redukce).} \end{aligned}$$

Tvrzení (1) je kontrapozicí $(\overline{1})$; tvrzení $(\overline{2})$ dokážeme podobně jako $(\overline{1})$ při použití úplného TS M_B; tvrzení (2) je kontrapozicí $(\overline{2})$.

Kontrapozice: $p \to q \Leftrightarrow \neg q \to \neg p$ (Modus tollens)

□

Problém náležitosti a další problémy

Problém náležitosti pro $\mathcal{L}_0$

Věta 9.4 **Problém náležitosti (Membership Problem (MP))** řetězce w do jazyka L typu 0 **není rozhodnutelný**, ale **je částečně rozhodnutelný**.

$$MP = \{ <M> \# <w> \mid M \text{ je TS, který akceptuje } w\}$$

Důkaz. Částečná rozhodnutelnost je zřejmá (podobně jako u HP využijeme T_U). Nerozhodnutelnost ukážeme redukcí z problému zastavení: $HP \leq MP$.

- Požadovaná redukce je funkce $\sigma : \{0,1,\#\}^* \to \{0,1,\#\}^*$ definovaná jako $\sigma(<M> \# <w>) = <M'> \# <w>$.
- Pokud vstup $<M> \# <w>$ není korektní instance HP, tak funkce σ vrací kód TS M' takového, že $L(M') = \emptyset$ (tj. $<M'> \# <w> \notin MP$). Jinak σ vrací kód TS M', který pracuje následovně.
- – M' nejdříve spustí simulaci stroje M na vstupu w. Poznamenejme, že kódy M a w jsou přímo uloženy v kódu M'.
  – Pokud simulace cyklí, tak M' cyklí pro svůj vstup w.
  – Pokud simulace skončí, tak M' akceptuje svůj vstup w.

Důkaz pokračuje dále.

Pokračování důkazu.

- Tudíž pro M' platí:

$$<M> \# <w> \in HP \Rightarrow w \in L(M') \Rightarrow <M'> \# <w> \in MP \text{ a}$$

$$<M> \# <w> \notin HP \Rightarrow w \notin L(M') \Rightarrow <M'> \# <w> \notin MP \iff$$

$$<M'> \# <w> \in MP \Rightarrow <M> \# <w> \in HP \text{ (použijeme kontrapozici)}$$

$$\text{neboli } <M> \# <w> \in HP \iff \sigma(<M> \# <w>) \in MP.$$

- Je vidět, že výše popsanou konstrukci stroje M' lze implementovat pomocí úplného TS a tudíž funkce σ je totální, rekurzivně vyčíslitelná funkce.

□

❖ Podobně jako u problému zastavení nyní z věty 8.6 plyne, že

- **komplement problému náležitosti** není ani částečně rozhodnutelný a
- jazyk co-$MP = \{\langle M'' \rangle \# \langle w'' \rangle \mid w'' \notin L(M'')\}$ je dalším příkladem jazyka, jenž není ani rekurzívně vyčíslitelný.

Příklady dalších problémů pro TS

❖ Konstrukcí příslušného úplného TS (a v případě složitější konstrukce důkazem její korektnosti) lze ukázat, že např. následující problémy jsou rozhodnutelné:

- Daný TS má alespoň 2005 stavů.
- Daný TS učiní více než 2005 kroků na vstupu ε.
- Daný TS učiní více než 2005 kroků na *nějakém* vstupu.

❖ Konstrukcí příslušného (ne-úplného) TS a důkazem nerekurzívnosti redukcí lze ukázat, že např. následující problémy jsou částečně rozhodnutelné:

- Jazyk daného TS je neprázdný.
- Jazyk daného TS obsahuje alespoň 2005 slov.

❖ Důkazem redukcí, že jazyky odpovídající následujícím problémům nejsou ani parciálně rekurzívní lze ukázat, že např. následující problémy nejsou ani částečně rozhodnutelné:

- Jazyk daného TS je prázdný.
- Jazyk daného TS obsahuje nanejvýš 2005 slov.
- Jazyk daného TS je konečný (regulární, bezkontextový, kontextový, rekurzívní).

Jak poznat, že jazyk je/není REC/RE?

❖ **Tato charakterizace nelze efektivně použít k důkazu dané vlastnosti.**

❖ Jazyk je L rekurzivní pokud existuje pro všechna $w \in L$ konečný certifikát příslušnosti do jazyka a pro všechna $w \notin L$ konečný certifikát nepříslušnosti.

- $L_1 = \{ <M> \# <w> \mid M$ je LOA, který akceptuje $w\}$
- $L_2 = \{ <A> \mid A$ je KA, t.ž. $L(A) = \emptyset\}$
- $L_3 = \{ <M> \mid M$ je TS, který učiní více než 2005 kroků na všech vstupech$\}$

❖ Jazyk je L rekurzivně vyčíslitelný pokud existuje pro všechna $w \in L$ konečný certifikát příslušnosti do jazyka (certifikát nepříslušnosti může být nekonečný) .

- $MP = \{ <M> \# <w> \mid M$ je TS, který akceptuje $w\}$
- $L_4 = \{ <M> \mid M$ je TS, t.ž. $L(M) \neq \emptyset\}$
- $L_5 = \{ <M> \mid M$ je LOA, t.ž. $L(M) \neq \Sigma^*\}$

❖ Jazyk je L není ani rekurzivně vyčíslitelný pokud pro nějaké $w \in L$ neexistuje konečný certifikát příslušnosti do jazyka (certifikát nepříslušnosti může být konečný) .

- co-$MP = \{ <M> \# <w> \mid M$ je TS, který neakceptuje $w\}$
- $L_4 = \{ <M> \mid M$ je TS, t.ž. $L(M) = \emptyset\}$
- $L_5 = \{ <M> \mid M$ je LOA, t.ž. $L(M) = \Sigma^*\}$

Příklad: L ani $\overline{L}$ není RE

$$L = \{<M_1> \# <M_2> |\ M_1 \text{ a } M_2 \text{ jsou TS, t.ž. } L(M_1) \subseteq L(M_2)\}$$

❖ Napřed dokážeme, že $L \notin \mathcal{L}_0$ pomocí redukce co-$HP \leq L$.

Důkaz.

- Požadovaná redukce je funkce $\sigma : \{0,1,\#\}^* \to \{0,1,\#\}^*$ definovaná takto:

$$\sigma(<M> \# <w>) = <M_1> \# <M_2>$$

- Funkce σ vrací kód TS M_2, pro který platí $L(M_2) = \{a\}$.
- Pokud vstup $<M> \# <w>$ není korektní instance co-HP, tak funkce σ vrací kód TS M_1 takový, že $L(M_1) = \Sigma^*$ (tj. $<M_1> \# <M_2> \notin L$). Jinak σ vrací kód TS M_1, který pracuje následovně.
- – M_1 nejdříve spustí simulaci stroje M na vstupu w. Poznamenejme, že kódy M a w jsou přímo uloženy v kódu M_1.
  – Pokud simulace cyklí, tak M_1 cyklí pro svůj každý vstup
  – Pokud simulace skončí, tak M_1 akceptuje každý svůj vstup.

Důkaz pokračuje dále.

Pokračování důkazu.

- Tudíž pro M' platí:

$$<M> \# <w> \in \text{co-}HP \;\Rightarrow\; L(M_1) = \emptyset \;\Rightarrow <M_1> \# <M_2> \in L \text{ a}$$

$$<M> \# <w> \notin \text{co-}HP \;\Rightarrow\; L(M_1) = \Sigma^* \;\Rightarrow <M_1> \# <M_2> \notin L$$

neboli $<M> \# <w> \in \text{co-}HP \iff \sigma(<M> \# <w>) \in L.$

- Je vidět, že výše popsanou konstrukci stroje M_1 a M_2 lze implementovat pomocí úplného TS a tudíž funkce σ je totální, rekurzivně vyčíslitelná funkce.

□

Příklad: L ani $\overline{L}$ není RE (pokračování)

$$L = \{<M_1> \# <M_2> |\ M_1 \text{ a } M_2 \text{ jsou TS, t.ž. } L(M_1) \subseteq L(M_2)\}$$

❖ Nyní dokážeme, že $\overline{L} \notin \mathcal{L}_0$ pomocí redukce co-$HP \leq \overline{L}$.

Důkaz.

- Požadovaná redukce je funkce $\sigma : \{0,1,\#\}^* \to \{0,1,\#\}^*$ definovaná takto:

$$\sigma(<M> \# <w>) = <M_1> \# <M_2>$$

- Funkce σ vrací kód TS M_1, pro který platí $L(M_1) = \{a\}$.
- Pokud vstup $<M> \# <w>$ není korektní instance co-HP, tak funkce σ vrací kód TS M_2 takový, že $L(M_2) = \Sigma^*$ (tj. $<M_1> \# <M_2> \notin \overline{L}$). Jinak σ vrací kód TS M_2, který pracuje následovně.
- – M_2 nejdříve spustí simulaci stroje M na vstupu w. Poznamenejme, že kódy M a w jsou přímo uloženy v kódu M_2.
  – Pokud simulace cyklí, tak M_2 cyklí pro svůj každý vstup
  – Pokud simulace skončí, tak M_2 akceptuje každý svůj vstup.

Důkaz pokračuje dále.

Pokračování důkazu.

- Tudíž pro M' platí:

$$<M> \# <w> \in \text{co-}HP \;\Rightarrow\; L(M_2) = \emptyset \;\Rightarrow <M_1> \# <M_2> \in \overline{L} \text{ a}$$

$$<M> \# <w> \notin \text{co-}HP \;\Rightarrow\; L(M_2) = \Sigma^* \;\Rightarrow <M_1> \# <M_2> \notin \overline{L}$$

$$\text{neboli } <M> \# <w> \in \text{co-}HP \iff \sigma(<M> \# <w>) \in \overline{L}.$$

- Je vidět, že výše popsanou konstrukci stroje M_1 a M_2 lze implementovat pomocí úplného TS a tudíž funkce σ je totální, rekurzivně vyčíslitelná funkce.

□

Příklad komplikovanější redukce

❖ Dokažte, že následující jazyk není RE.

$$L_{NB} = \{<M>|\ M \text{ je TS t.ž. } L(M) \notin \mathcal{L}_2\}$$

❖ Opět použijeme redukci co-$HP \leq L_{NB}$.

Důkaz.

- Požadovaná redukce je funkce $\sigma : \{0,1,\#\}^* \to \{0,1\}^*$ definovaná takto:

$$\sigma(<M> \# <w>) = <M'>$$

- Pokud vstup $<M> \# <w>$ není korektní instance co-HP, tak funkce σ vrací kód TS M' takový, že $L(M') = \Sigma^*$ (tj. $<M'> \notin L_{NB}$). Jinak σ vrací kód TS M', který pracuje následovně.

Důkaz pokračuje dále.

Pokračování důkazu.

- – M' nejdříve ověří, zda jeho vstup $w' \in \{a^n b^n c^n \mid n > 0\}$. Pokud ano, M' akceptuje w', jinak pokračuje.
  – M' spustí simulaci stroje M na vstupu w. Poznamenejme, že kódy M a w jsou přímo uloženy v kódu M'.
  – Pokud simulace cyklí, tak M' cyklí pro svůj vstup w'.
  – Pokud simulace skončí, tak M' akceptuje svůj vstup w'.

- Tudíž pro M' platí:

$$<M> \# <w> \in \text{co-}HP \;\Rightarrow\; L(M') = \{a^n b^n c^n \mid n > 0\} \;\Rightarrow <M'> \in L_{NB} \text{ a}$$

$$<M> \# <w> \notin \text{co-}HP \;\Rightarrow\; L(M') = \Sigma^* \;\Rightarrow <M'> \notin L_{NB}$$

$$\text{neboli } <M> \# <w> \in \text{co-}HP \iff \sigma(<M> \# <w>) \in L_{NB}.$$

- Je vidět, že výše popsanou konstrukci stroje M' lze implementovat pomocí úplného TS a tudíž funkce σ je totální, rekurzivně vyčíslitelná funkce.

□

Postův korespondenční problém

Postův korespondenční problém

Definice 9.2

- Postův systém nad abecedou Σ je dán neprázdným seznamem S dvojic neprázdných řetězců nad Σ, $S = \langle(\alpha_1, \beta_1), ..., (\alpha_k, \beta_k)\rangle$, $\alpha_i, \beta_i \in \Sigma^+$, $k \geq 1$.
- Řešením Postova systému je každá neprázdná posloupnost přirozených čísel $I = \langle i_1, i_2, ..., i_m\rangle$, $1 \leq i_j \leq k$, $m \geq 1$, taková, že:

$$\alpha_{i_1}\alpha_{i_2}...\alpha_{i_m} = \beta_{i_1}\beta_{i_2}...\beta_{i_m}$$

(Pozor: m není omezené a indexy se mohou opakovat!)

- Postův problém (PCP) zní: Existuje pro daný Postův systém řešení?

Příklad 9.1

- Uvažujme Postův systém $S_1 = \{(b, bbb), (babbb, ba), (ba, a)\}$ nad $\Sigma = \{a, b\}$. Tento systém má řešení $I = \langle 2, 1, 1, 3\rangle$: $babbb\ b\ b\ ba = ba\ bbb\ bbb\ a$.
- Naopak Postův systém $S_2 = \{(ab, abb), (a, ba), (b, bb)\}$ nad $\Sigma = \{a, b\}$ nemá řešení, protože $|\alpha_i| < |\beta_i|$ pro $i = 1, 2, 3$.

Nerozhodnutelnost PCP

Věta 9.5 **Postův korespondenční problém je nerozhodnutelný.**

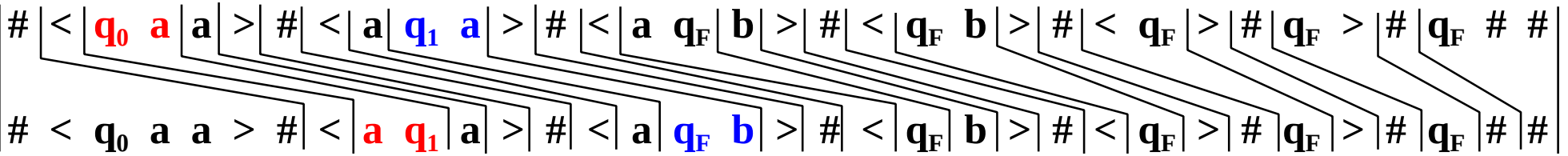
Důkaz. *(Idea) Dá se ukázat, že nerozhodnutelnost PCP plyne z nerozhodnutelnosti tzv. iniciálního PCP, u kterého požadujeme, aby řešení začínalo vždy jedničkou. Nerozhodnutelnost iniciálního PCP se dá ukázat redukcí z problému náležitosti pro TS:

- Konfiguraci výpočtu TS lze zakódovat do řetězce: použitý obsah pásky ohraničíme speciálními značkami (a jen ten si pamatujeme), řídicí stav vložíme na aktuální pozici hlavy na pásce.
- Posloupnost konfigurací TS při přijetí řetězce budeme reprezentovat jako konkatenaci řetězců, která vzniká řešením PCP.
- Jedna z uvažovaných konkatenací bude celou dobu (až na poslední fázi) delší: na začátku bude obsahovat počáteční konfiguraci a pak bude vždy o krok napřed. V posledním fázi výpočtu konkatenace „zarovnáme“ (bude-li možné výpočet simulovaného TS ukončit přijetím).
- Výpočet TS budeme modelovat tak, že vždy jednu konkatenaci postupně prodloužíme o aktuální konfiguraci simulovaného TS a současně v druhé konkatenaci vygenerujeme novou konfiguraci TS.

Důkaz pokračuje dále.

Pokračování důkazu.

- Jednotlivé dvojice PCP budou modelovat následující kroky:
  - Vložení počáteční konfigurace simulovaného TS do jedné z konkatenací např. pravostranné $(\#, \#\text{počáteční\_konfigurace})$, $\# \notin \Gamma$ používáme jako oddělovač konfigurací.
  - Kopírování symbolů na pásce před a po aktuální pozici hlavy (z, z) pro každé $z \in \Gamma \cup \{\#, <, >\}$, kde $<$, $>$ ohraničují použitou část pásky.
  - Základní změna konfigurace: přepis $\delta(q_1, a) = (q_2, b)$: $(q_1 a, q_2 b)$, posuv doprava $\delta(q_1, a) = (q_2, R)$: $(q_1 a, a q_2)$, posuv doleva $\delta(q_1, b) = (q_2, L)$: $(a q_1 b, q_2 a b)$ pro každé $a \in \Gamma \cup \{<\}$. Navíc je zapotřebí ošetřit nájezd na $>$: čtení Δ, rozšiřování použité části pásky.
  - Pravidla pro „zarovnání“ obou konkatenací při přijetí: na levé straně umožníme přidat symbol v okolí q_F, aniž bychom ho přidali na pravé straně.
- Simulace výpočtu TS, který načte a, posune hlavu doprava, přepíše a na b a zastaví, na vstupu aa by pak vypadala takto:



- Obecná korektnost konstrukce se dá dokázat indukcí nad délkou výpočtu.

□ *

Nerozhodnutelnost redukcí z PCP

❖ Redukce z PCP (resp. jeho doplňku) se velmi často používají k důkazům, že určitý problém není rozhodnutelný (resp. není ani částečně rozhodnutelný).

❖ Jako příklad uvedeme důkaz faktu, že problém prázdnosti jazyka dané kontextové gramatiky není ani částečně rozhodnutelný:

- Použijeme redukci z komplementu PCP. Redukce přiřadí seznamu $S = (\alpha_1, \beta_1), ..., (\alpha_k, \beta_k)$, definujícímu instanci PCP, kontextovou gramatiku G takovou, že PCP založený na S nemá řešení právě tehdy, když $L(G) = \emptyset$.
- Uvažme jazyky L_α, L_β nad $\Sigma \cup \{\#, 1, ..., k\}$ (předp. $\Sigma \cap \{\#, 1, ..., k\} = \emptyset$):
  - $L_\alpha = \{\alpha_{i_1} ... \alpha_{i_m} \# i_m ... i_1 \mid 1 \leq i_j \leq k, j = 1, ..., m, m \geq 1\}$,
  - $L_\beta = \{\beta_{i_1} ... \beta_{i_m} \# i_m ... i_1 \mid 1 \leq i_j \leq k, j = 1, ..., m, m \geq 1\}$.
- Je zřejmé, že L_α, L_β jsou kontextové (dokonce deterministické bezkontextové) a tudíž $L_\alpha \cap L_\beta$ je také kontextový jazyk (věta 8.10) a můžeme tedy efektivně sestavit gramatiku G, která tento jazyk generuje (např. konstrukcí přes LOA).
- $L_\alpha \cap L_\beta$ zřejmě obsahuje právě řetězce $u\#v$, kde v odpovídá reverzi řešení dané instance PCP.
- Hledaná redukce tedy přiřadí dané instanci PCP gramatiku G.

□

Souhrn některých vlastností jazyků

❖ Uvedeme nyní souhrn některých důležitých vlastností různých tříd jazyků; některé jsme dokázali, důkazy jiných lze nalézt v literatuře[a] (u otázek nerozhodnutelnosti se často užívá redukce z PCP) – R = rozhodnutelný, N = nerozhodnutelný, A = vždy splněno:

	Reg	DCF	CF	CS	Rec	RE
$w \in L(G)$?	R	R	R	R	R	N
$L(G)$ prázdný? konečný?	R	R	R	N	N	N
$L(G) = \Sigma^*$?	R	R	N	N	N	N
$L(G) = R$, $R \in \mathcal{L}_3$?	R	R	N	N	N	N
$L(G_1) = L(G_2)$?	R	R	N	N	N	N
$L(G_1) \subseteq L(G_2)$?	R	N	N	N	N	N
$L(G_1) \in \mathcal{L}_3$?	A	R	N	N	N	N
$L(G_1) \cap L(G_2)$ je stejného typu?	A	N	N	A	A	A
$L(G_1) \cup L(G_2)$ je stejného typu?	A	N	A	A	A	A
Komplement $L(G)$ je stejného typu?	A	A	N	A	A	N
$L(G_1).L(G_2)$ je stejného typu?	A	N	A	A	A	A
$L(G)^*$ je stejného typu?	A	N	A	A	A	A
Je G víceznačná?	R	N	N	N	N	N

[a] Např. I. Černá, M. Křetínský, A. Kučera. Automaty a formální jazyky I. FI MU, 1999.

Riceova věta

Nerozhodnutelnost je pravidlo, ne výjimka.

Riceova věta – první část

Věta 9.6 ***Každá* netriviální vlastnost rekurzívně vyčíslitelných jazyků je nerozhodnutelná.**

Definice 9.3 Budiž dána abeceda Σ. Vlastnost rekurzívně vyčíslitelných množin je zobrazení $P : \{$ rekurzívně vyčíslitelné podmnožiny množiny $\Sigma^* \} \to \{\bot, \top\}$, kde $\top$, resp. $\bot$ reprezentují pravdu, resp. nepravdu.

Příklad 9.2 Vlastnost prázdnosti můžeme reprezentovat jako zobrazení

$$P(A) = \begin{cases} \top, \text{ jestliže } A = \emptyset, \\ \bot, \text{ jestliže } A \neq \emptyset. \end{cases}$$

❖ Zdůrazněme, že nyní mluvíme o vlastnostech rekurzívně vyčíslitelných množin, *nikoliv* TS, které je přijímají – následující vlastnosti tedy nejsou vlastnostmi r. v. množin:

- TS M má alespoň 2005 stavů.
- TS M zastaví na všech vstupech.

Definice 9.4 Vlastnost rekurzívně vyčíslitelných množin je netriviální, pokud není vždy pravdivá ani vždy nepravdivá.

Důkaz 1. části Riceovy věty

Důkaz.

- Nechť P je netriviální vlastnost r.v. množin. Předpokládejme beze ztráty obecnosti, že $P(\emptyset) = \bot$, pro $P(\emptyset) = \top$ můžeme postupovat analogicky.
- Jelikož P je netriviální vlastnost, existuje r.v. množina A taková, že $P(A) = \top$. Nechť K je TS přijímající A.
- Redukujeme HP na $\{\langle M\rangle \mid P(L(M)) = \top\}$. Z $\langle M\rangle\#\langle w\rangle$ sestrojíme $\sigma(\langle M\rangle\#\langle w\rangle) = \langle M'\rangle$, kde M' je 2-páskový TS, který na vstupu x:
  1. Uloží x na 2. pásku.
  2. Zapíše na 1. pásku w – w je „uložen“ v řízení M'.
  3. Odsimuluje na 1. pásce M – M je rovněž „uložen“ v řízení M'.
  4. Pokud M zastaví na w, odsimuluje K na x a přijme, pokud K přijme.
- Dostáváme:
  - M zastaví na $w \Rightarrow L(M') = A \Rightarrow P(L(M')) = P(A) = \top$,
  - M cyklí na $w \Rightarrow L(M') = \emptyset \Rightarrow P(L(M')) = P(\emptyset) = \bot$,

  A máme tedy skutečně redukci HP na $\{\langle M\rangle \mid P(L(M)) = \top\}$.
  Protože HP není rekurzivní, není rekurzivní ani $P(L(M))$ a tudíž není rozhodnutelné, zda $L(M)$ splňuje vlastnost P.

□

Riceova věta – druhá část

Definice 9.5 Vlastnost P r.v. množin nazveme monotónní, pokud pro každé dvě r.v. množiny A, B takové, že $A \subseteq B$, $P(A) \Rightarrow P(B)$.

Příklad 9.3 Mezi monotónní vlastnosti patří např.:

- A je nekonečné.
- $A = \Sigma^*$.

Naopak mezi nemonotónní vlastnosti patří např.:

- A je konečné.
- $A = \emptyset$.

Věta 9.7 ***Každá*** **netriviální nemonotónní vlastnost rekurzívně vyčíslitelných jazyků není ani částečně rozhodnutelná.**

Důkaz. Redukcí z co-HP – viz např. D. Kozen. Automata and Computability. □

Alternativy k TS

Některé alternativy k TS

❖ Mezi výpočetní mechanismy mající ekvivalentní výpočetní sílu jako TS patří např. automaty s (jednou) frontou:

- Uvažme stroj s konečným řízením, (neomezenou) FIFO frontou a přechody na nichž je možno načíst ze začátku fronty a zapsat na konec fronty symboly z frontové abecedy Γ.
- Pomocí „rotace“ fronty je zřejmě možné simulovat pásku TS.

❖ Ekvivalentní výpočetní sílu jako TS mají také zásobníkové automaty se dvěma (a více) zásobníky:

- Intuitivně: obsah pásky simulovaného TS máme v jednom zásobníku; chceme-li ho změnit (obecně nejen na vrcholu), přesuneme část do druhého zásobníku, abychom se dostali na potřebné místo, provedeme patřičnou změnu a vrátíme zpět odloženou část zásobníku.
- Poznámka: rovněž víme, že pomocí dvou zásobníků můžeme implementovat frontu.

❖ Jiným výpočetním modelem s plnou Turingovskou silou jsou automaty s čítači (pro dva a více čítačů) a operacemi $+1$, -1 a test na 0:

- Zmíněné automaty mají konečné řízení a k čítačů, přičemž v každém kroku je možné tyto čítače nezávisle inkrementovat, dekrementovat a testovat na nulu (přechod je podmíněn tím, že jistý čítač obsahuje nulu).
- Pomocí čtyř čítačů je snadné simulovat dva zásobníky:
  - U ZA postačí mít $\Gamma = \{0, 1\}$: různé symboly můžeme kódovat určitým počtem 0 oddělených 1. Obsah zásobníku má pak charakter binárně zapsaného čísla. Vložení 0 odpovídá vynásobení 2, odebrání 0 vydělení 2. Podobně je tomu s vložením/odebráním 1.
  - Binární zásobník můžeme simulovat dvěma čítači: při násobení/dělení 2 odečítáme 1 (resp. 2) z jednoho čítače a přičítáme 2 (resp. 1) k druhému.
- Postačí ovšem i čítače dva:
  - Obsah čtyř čítačů i, j, k, l je možné zakódovat jako $2^i 3^j 5^k 7^l$.
  - Přičtení/odečtení je pak možné realizovat násobením/dělením 2, 3, 5, či 7.

❖ Mezi další Turingovsky úplné výpočetní mechanismy pak patří např. λ-kalkul či parciálně-rekurzívní funkce (viz další přednáška).