

Problémy, které nejsou řešitelné, a problémy které nejsou ani částečně řešitelné. Uzávěrové vlastnosti rekurzivních a částečně rekurzivních jazyků.

Jan Konečný

9. října 2013

Z PŘEDNÁŠKY 1 víme:

- jazyk $\approx$ problém
- jazyků je nespočetně mnoho, T. strojů je spočetně mnoho:

Nutně musí existovat jazyky, které nejsou přijímané (nebo rozhodované) TS – a není jich málo.

Dále víme, že $R \subseteq \check{R}$

Dnes:

- Jazyky $\notin \check{R}$, jazyky $\notin R$,
- $R \subset \check{R}$,
- uzávěrové vlastnosti TS.

Uzávěrové vlastnosti rekurzivních jazyků

Operace s jazyky (opáčko z KMI/FJAA)

Množinové operace (protože jazyk je množina řetězců).

- průnik $L_1 \cap L_2$,
- sjednocení $L_1 \cup L_2$,
- doplněk $\Sigma^* \setminus L$

Další jednoduché, ale mocné operace

- konkatenace $L_1 L_2$ (nebo taky $L_1 \circ L_2$)
- Kleeneho uzávěr L^*
- pozitivní uzávěr L^+

Operace s jazyky: konkaténace

Definice

Nechť L_1 a L_2 jsou jazyky. Jejich *konkaténací* rozumíme jazyk

$$L_1 \circ L_2 = \{w_1w_2 \mid w_1 \in L_1, w_2 \in L_2\}$$

Příklad

$$L_1 = \{00, 10\}, L_2 = \{0, 1\}, L_1 \circ L_2 = \{000, 001, 100, 101\}.$$

$$L \circ \{\varepsilon\} = L$$

$$L \circ \emptyset = L$$

Operace s jazyky: uzávěry

Definice

Nechť L je jazyk, *Kleeneho uzávěr* L^* jazyka L definujeme jako

$$L^* = \bigcup_{i \geq 0} L^i, \quad \text{kde } L^n = \begin{cases} \{\varepsilon\} & \text{pokud } n = 0, \\ L^{n-1} \circ L & \text{jinak.} \end{cases}.$$

Pozitivní uzávěr je pak definován jako

$$L^+ = \bigcup_{i \geq 1} L^i.$$

Poznámka

Pozitivním uzávěrem se zabývat nebudeme, dá se vyjádřit jako

$$L^+ = L^* \circ L.$$

Uzávěrové vlastnosti třídy R: doplněk

Věta

Pokud L je rekurzivní, pak $\Sigma^ \setminus L$ je rekurzivní.*

Důkaz.

Pokud je L rekurzivní, existuje TS T , který ho rozhoduje. Sestavíme $\neg T$ takto.

TS $\neg T$ pro w :

- ❶ Spustí T pro w ,
pokud
 - T přijme w , $\neg T$ zamítne.
 - T zamítne w , $\neg T$ přijme.

Takový stroj bude rozhodovat $\Sigma^* \setminus L$, a tedy $\Sigma^* \setminus L$ je rekurzivní. □

Poznámka

Vlastně $\neg T$ vyrobíme z T tak, že zaměníme koncové stavy.

Věta

Nechť $L_1, L_2 \in R$ jsou rekurzivní jazyky, pak $L_1 \cup L_2 \in R$ a $L_1 \cap L_2 \in R$.

Důkaz (pro $L_1 \cup L_2$).

Nechť TS T_1 rozhoduje L_1 a TS T_2 rozhoduje L_2 . Sestavíme TS $T_\cup$, který rozhoduje $L_1 \cup L_2$, takto:

TS $T_\cup$ pro w :

- ❶ *spustí T_1 pro w , pokud*
 - *T_1 přijme w , $T_\cup$ přijme w .*
 - *T_1 zamítne w , pokračujeme krokem 2.*
- ❷ *spustí T_2 pro w , pokud*
 - *T_2 přijme w , $T_\cup$ přijme w .*
 - *T_2 zamítne w , $T_\cup$ zamítne w .*

Pro $L_1 \cap L_2$ se to dokáže analogicky.



Otázka: jak by to bylo se sjednocením (či průnikem) nekonečně mnoha jazyků?

Věta

Nechť $L_1, L_2 \in R$, pak $L_1 \circ L_2 \in R$.

Důkaz.

Nechť TS T_1 rozhoduje L_1 a TS T_2 rozhoduje L_2 .

Sestavíme nedeterministický TS $T_\circ$ rozhodující $L_1 \circ L_2$ takto:

NTS $T_\circ$ pro w :

- ❶ *pokud čte prázdný symbol pokračuje bodem 2. Jinak se nedeterministicky rozhodne mezi následujícími dvěma možnostmi:*
 - *pokračuje bodem 2.*
 - *posune hlavu doprava, a pokračuje bodem 1.*
- ❷ *spustí T_1 pro w_1 (část w od hlavy vlevo), pokud*
 - *T_1 přijme w_1 , pokračujeme krokem 3.*
 - *T_1 zamítne w_1 , $T_\circ$ zamítne w .*
- ❸ *spustí T_2 pro w_2 (část w od hlavy vpravo), pokud*
 - *T_2 přijme w_2 , $T_\circ$ přijme w .*
 - *T_2 zamítne w_2 , $T_\circ$ zamítne w .*

Věta

Pokud $L \in R$, pak $L^ \in R$.*

Důkaz.

Nechť TS T rozhoduje L .

NTS T_0 pro w :

- ❶ *pokud je prázdný vstup, přijme w*
- ❷ *pokud čte prázdný symbol pokračuje bodem 3. Jinak se nedeterministicky rozhodne mezi následujícími dvěma možnostmi:*
 - *pokračuje bodem 3.*
 - *posune hlavu doprava, a pokračuje bodem 2.*
- ❸ *spustí T_1 pro w_1 (část w od hlavy doleva), pokud*
 - *T_1 přijme w_1 , smaže w_1 ze vstupu pokračujeme krokem 1.*
 - *T_1 zamítne w_1 , T_0 zamítne w .*



Uzávěrové vlastnosti částečně rekurzivních jazyků

Věta

Nechť $L, L_1, L_2 \in \check{R}$ pak

- $L_1 \cup L_2 \in \check{R}$,
- $L_1 \cap L_2 \in \check{R}$,
- $L_1 \circ L_2 \in \check{R}$,
- $L^* \in \check{R}$.

Poznámka

Všimněte si, že tam není doplněk: to proto že třída částečně rekurzivních jazyků není uzavřená na doplněk.

Další vlastnosti tříd R a ČR

Věta

Pokud $L \in \check{C}R$ a $\Sigma^ \setminus L \in \check{C}R$, pak $L \in R$.*

Idea důkazu.

Mějme TS T_L , který přijímá L a TS $T_{\Sigma^* \setminus L}$, který přijímá $\Sigma^* \setminus L$. Můžu sestavit TS, který pro vstupní slovo w bude simulovat výpočty T_L pro w a $T_{\Sigma^* \setminus L}$ pro w („paralelně“ po jednom kroku). Protože platí buďto $w \in L$ nebo $w \in \Sigma^* \setminus L$, jeden z těchto výpočtů musí skončit přijetím.

- pokud skončí přijetím simulace T_L , přijme w ,
- pokud skončí přijetím simulace $T_{\Sigma^* \setminus L}$ zamítne w .

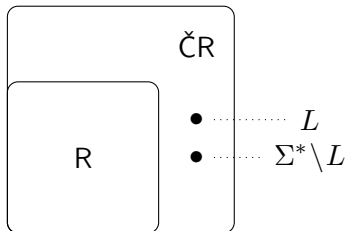
Takový stroj nebude nikdy cyklit a bude přijímat L .



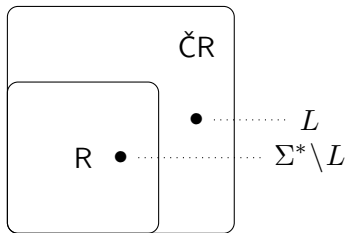
Poznámka

Taky pak platí, že $\Sigma^* \setminus L \in R$.

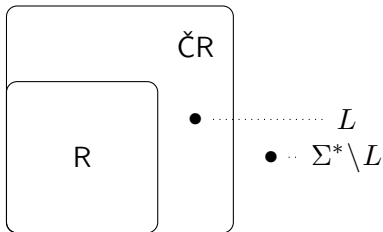
Jinými slovy (nebo spíše obrázkem), nemůže nastat tato situace



A taky nemůže nastat tato situace, protože R je uzavřená na doplněk.



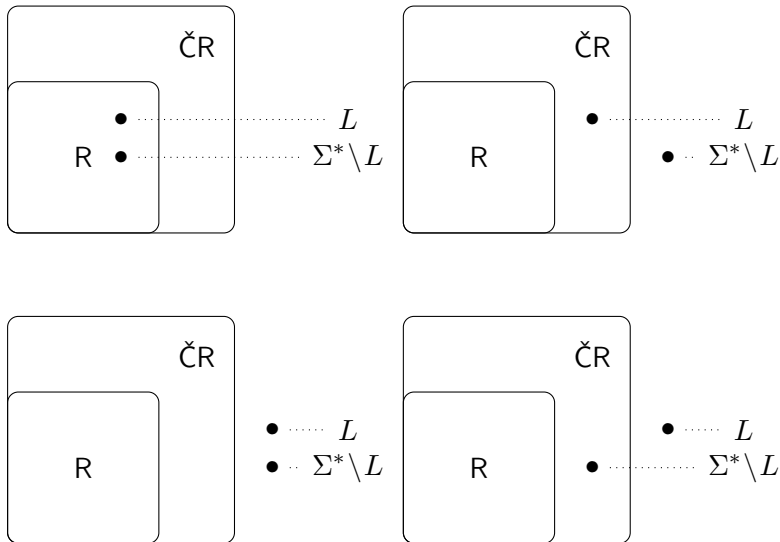
Pokud $L \in \check{C}R$ a $L \notin R$ může nastat jenom následující:



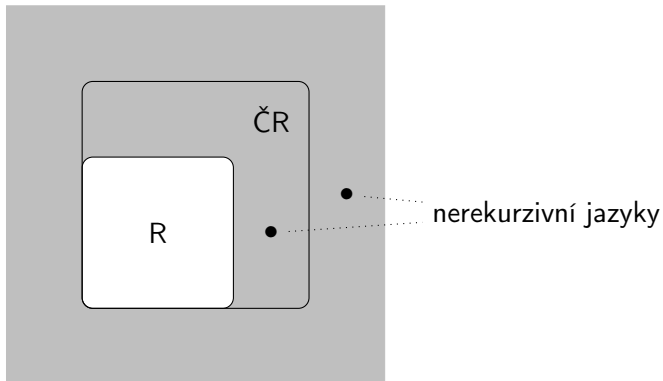
Důsledek

Pokud $L \in \check{C}R$ a $L \notin R$, pak $\Sigma^ \setminus L \notin \check{C}R$.*

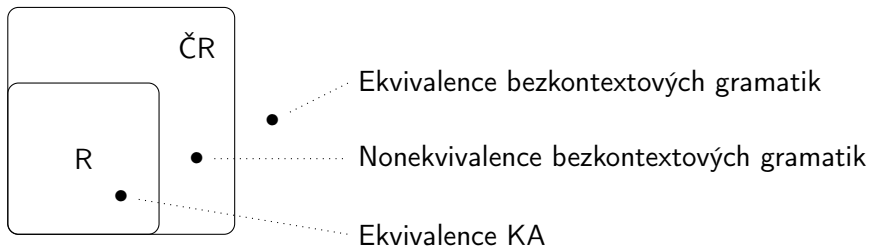
Jazyky TS a jejich doplňky



K názvosloví: neřešitelný problém, nerekurzivní problém.



Z PŘEDNÁŠKY 1 známe pár příkladů (zatím tomu jenom věříme).



Jazyky, které jsou částečně rekurzivní, ale nejsou rekurzivní

Definice

Univerzální jazyk L_U je definován takto:

$$L_U = \{[T, w] \mid T \text{ je TS, který přijímá } w\}.$$

Definice

Problém přijetí slova Turingovým strojem je definován jako:

Problém přijetí

Instance: T, w , kde T je TS a w je slovo nad jeho vstupní abecedou.

Otázka: *Přijme TS T slovo w ?*

Poznámka

Je to to samé.

Lemma

L_U je částečně rekurzivní.

Důkaz.

Univerzální TS ho přijímá. ☐

Lemma

L_U není rekurzivní.

(důkaz na další stránce)

Důkaz.

Sporem: předpokládejme, že rekurzivní je. A tedy existuje TS U který rozhoduje L_U .

Sestavíme následující TS D :

TS D pro $[T]$, kde T je TS:

- 1 Spustí U pro $[T, [T]]$,
pokud
 - U přijme $[T, [T]]$, D zamítne.
 - U zamítne $[T, [T]]$, D přijme.

Zjevně tento stroj nikdy necyklí.

Co se stane když stroji D dáme na vstup jeho vlastní kód $[D]$?

Předpokládejme, že $[D] \in L(D)$:

D pro $[D]$: spustí U pro $[D, [D]]$, ten přijme, protože $[D] \in L(D)$, a D zamítne. A tedy $[D] \notin L(D)$. **SPOR**

Předpokládejme, že $[D] \notin L(D)$:

D pro $[D]$: spustí U pro $[D, [D]]$, ten zamítne, protože $[D] \notin L(D)$, a D přijme. A tedy $[D] \in L(D)$. **SPOR** □

Definice

Problém zastavení Turingova stroje je definován jako:

Problém zastavení

Instance: T, w , kde T je TS a w je slovo nad jeho vstupní abecedou.

Otázka: Zastaví TS T pro slovo w ?

Totéž jako jazyk:

Definice

Jazyk L_{HALT} je definován takto:

$$L_{\text{HALT}} = \{[T, w] \mid T \text{ je TS, který zastaví pro } w\}.$$

Poznámka

Napohled skoro totéž jako L_u .

Věta

$L_{\text{HALT}} \notin R, L_{\text{HALT}} \in \check{R}.$

Idea důkazu.

V podstatě stejné jako v případě L_U . ☐

Jak převést jeden problém na druhý (a naopak)? NA TABULI.

Poznámka

Tomuto principu se říká *redukce*, budeme se jím zabývat na PŘEDNÁŠCE 5.

Věta

L_U je částečně rekurzivní jazyk, který není rekurzivní.

Důkaz.

Z předchozích dvou lemmat. ☐

Důsledek

Třída rekurzivních jazyků je vlastní podtřídou částečně rekurzivních jazyků.

Důkaz.

$L_U \in \text{ČR} \setminus \text{R}$. ☐

Definice

$$L_{NE} = \{ [T] \mid L(T) \neq \emptyset \}$$

A jako problém:

Definice

Problém 'neprázdnosti' jazyka Turingova stroje je definován jako:

<i>Problém 'neprázdnosti' jazyka</i>
Instance: T , kde T je TS.
Otázka: <i>Přijme TS T alespoň jedno slovo?</i>

Věta

$$L_{NE} \in \check{R}, L_{NE} \notin R.$$

Idea důkazu (první části).

Setavíme TS, který bude (stejnou fintou jako u enumeratoru, který byl ekvivalentní TS) zkoušet spouštět T pro všechna slova, jakmile bude jedno přijato, přijme. □

Důkaz (druhé části).

(sporem) Předpokládejme, že existuje takový TS T_{NE} , který rozhoduje L_{NE} .

Nejdříve si uveďme následující TS:

TS $S_{T,w}$ pro x

- ❶ porovná x a w , pokud jsou různé, zamítne.
- ❷ pak spustí T_{NE} pro w , pokud
 - T_{NE} přijal w , $S_{T,w}$ přijme x .
 - T_{NE} zamítl w , $S_{T,w}$ zamítne x .

Sestrojíme následující TS U :

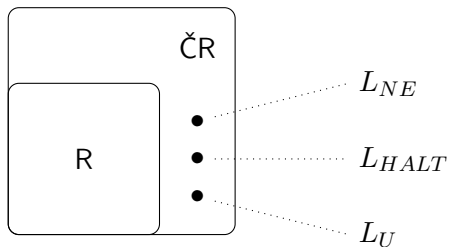
TS U pro $[T, w]$, kde T je TS a w je vstup pro T :

- ❶ Sestaví TS S_T ;
- ❷ Simuluje činnost $S_{T,w}$ pro w , pokud
 - $S_{T,w}$ přijal w , U přijme $[T, w]$.
 - $S_{T,w}$ zamítl w , U zamítne $[T, w]$.

No jo, ale U nám teď rozhoduje L_U . **SPOR.**



Shrnutí předchozích 9 slajdů



Jazyky, které nejsou částečně rekurzivní

Definice

Diagonální jazyk L_d je definován jako

$$L_d = \{[T] \mid T \text{ je TS, který nepřijímá svůj kód.}\}$$

Věta

L_d není částečně rekurzivní.

Důkaz.

Sporem. Předpokládáme, že existuje TS D , který ho přijímá. Patří $[D]$ do L_d ?

Pokud $[D] \in L_d$, pak D nepřijímá $[D]$, pak ale $[D] \notin L(D) = L_d$. **SPOR**

Pokud $[D] \notin L_d$, pak D přijímá $[D]$, pak ale $[D] \in L(D) = L_d$. **SPOR**



Jiný důkaz téhož.

Důkaz.

Protože TS lze zakódovat do řetězců a řetězce lze očíslovat (jsou spočetné, existuje bijekce na $\mathbb{N}$), můžu seřadit stroje podle jejich kódů.

Můžu tedy uvažovat tabulku, jejíž řádky budou TS, sloupce budou kódy TS, tak aby když je v i -tém řádku TS T_i , tak v i -tém sloupci $[T_i]$.

	$[T_1]$	$[T_2]$	$[T_3]$	$[T_4]$	$\dots$
T_1	1				
T_2			1		
T_3		1			
T_4				1	
$\vdots$					

Tabulka obsahuje 1 na pozici $\langle i, j \rangle$ pokud T_i přijímá $[T_j]$. Pokud existuje TS D rozhodující L_d , musí být někde v té tabulce, řekněme na řádku x . Co ale bude na pozici $\langle x, x \rangle$?

Ta hodnota se musí sama od sebe lišit. **SPOR**



Odsud název „diagonální jazyk“.

Ještě jednou důkaz toho, že $L_U \notin R$

Důkaz.

Sporem: předpokládejme, že rekurzivní je. A tedy existuje TS U který rozhoduje L_U .

Sestavíme následující TS D :

TS D pro $[T]$, kde T je TS:

- 1 Spustí U pro $[T, [T]]$,
pokud
 - U přijme $[T, [T]]$, D zamítne.
 - U zamítne $[T, [T]]$, D přijme.

Takto sestrojený TS rozhoduje D . **SPOR.**



Jazyk Turingových strojů přijímajících prázdný jazyk

Definice

$$L_{\emptyset} = \{[T] \mid T \text{ je TS a } L(T) = \emptyset\}.$$

Resp, jako problém

Problém odpovídající jazyku $L_{\emptyset}$

Instance: TS T

Otázka: *Nepřijímá TS T žádné slovo?*

Věta

$$L_{\emptyset} \notin \text{ČR}.$$

Idea důkazu.

Dá se ukázat, že jeho doplněk je v ČR (vlastně už jsme dokázali).



Jazyk Turingových strojů, které se nezacyklí

Definice

$L_{nocycle} = \{[T] \mid T \text{ je TS a necyklí pro žádné slovo.}\}.$

Resp, jako problém

Problém odpovídající jazyku $L_{nocycle}$

Instance: TS T

Otázka: Necyklí TS T žádné slovo?

Věta

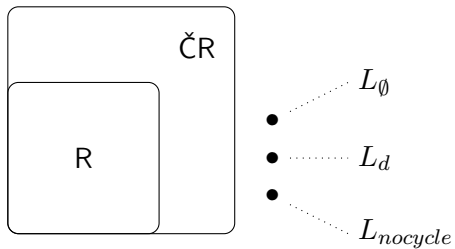
$L_{nocycle} \notin \text{ČR.}$

Idea důkazu.

Podobný vztah jako mezi L_U a L_{HALT} .



Shrnutí předchozích 6 slajdů

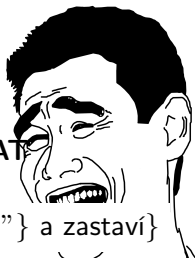




MAKE A SOFTWARE TO RECOGNIZES
'HELLO WORLD' PROGRAMS



BITCH PLEASE, NOONE CAN DO THAT



$L_{hello} = \{[E] \mid E \text{ je enumerator a } L(E) = \{\text{"hello world"}\} \text{ a zastaví}\}$

Definice

Jazyk HALT_{LBA} je definován takto.

$$\text{HALT}_{\text{LBA}} = \{[M, w] \mid M \text{ je LBA, který zastaví pro } w\}.$$

Věta

Jazyk HALT_{LBA} je rekurzivní.

Důkaz.

Názna: stačí sestavit TS, který simuluje činnost LBA a navíc počítá simulované kroky. Protože LBA má pro použití konečný pevně daný počet políček pásky, které může pro zpracování w použít, existuje konečný počet konfigurací, ve kterých se může nacházet. Pokud je výpočet delší, než tento počet, LBA cyklí (musel být v nějaké konfiguraci alespoň dvakrát). TS, který simuluje činnost LBA může tedy snadno detekovat zacyklení. $\square$

Jako důsledek této věty dostáváme následující větu:

Věta

Každý jazyk přijímaný LBA je rekurzivní.

Důkaz.

Nechť X je TS, t.ž. $L(X) = \text{HALT}_{\text{LBA}}$ a L je jazyk přijímaný LBA A .
Sestrojíme následující TS:

TS M_{LBA} pro vstupní slovo w :

- ❶ *spustí TS X pro vstupní slovo $[A, w]$, pokud X zamítne $[A, w]$, M_{LBA} zamítne w .*
- ❷ *simuluje činnost A pro w ; pokud A přijme w , pak M_{LBA} přijme w , jinak M_{LBA} zamítne w .*

