Základy složitosti a vyčíslitelnosti

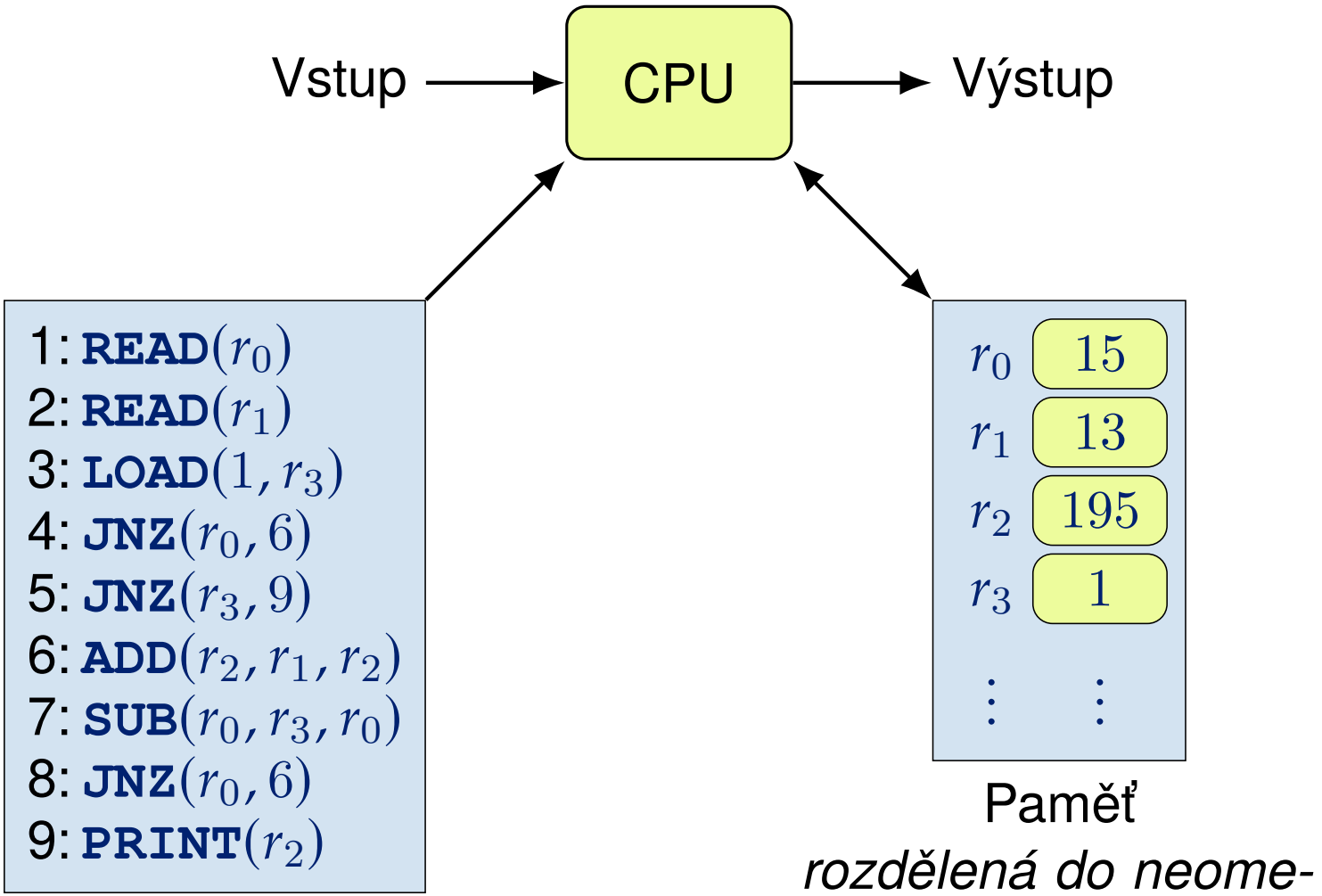
NTIN090

Petr Kučera

2022/23 (2. přednáška)

Random Access Machine

Random Access Machine (RAM)



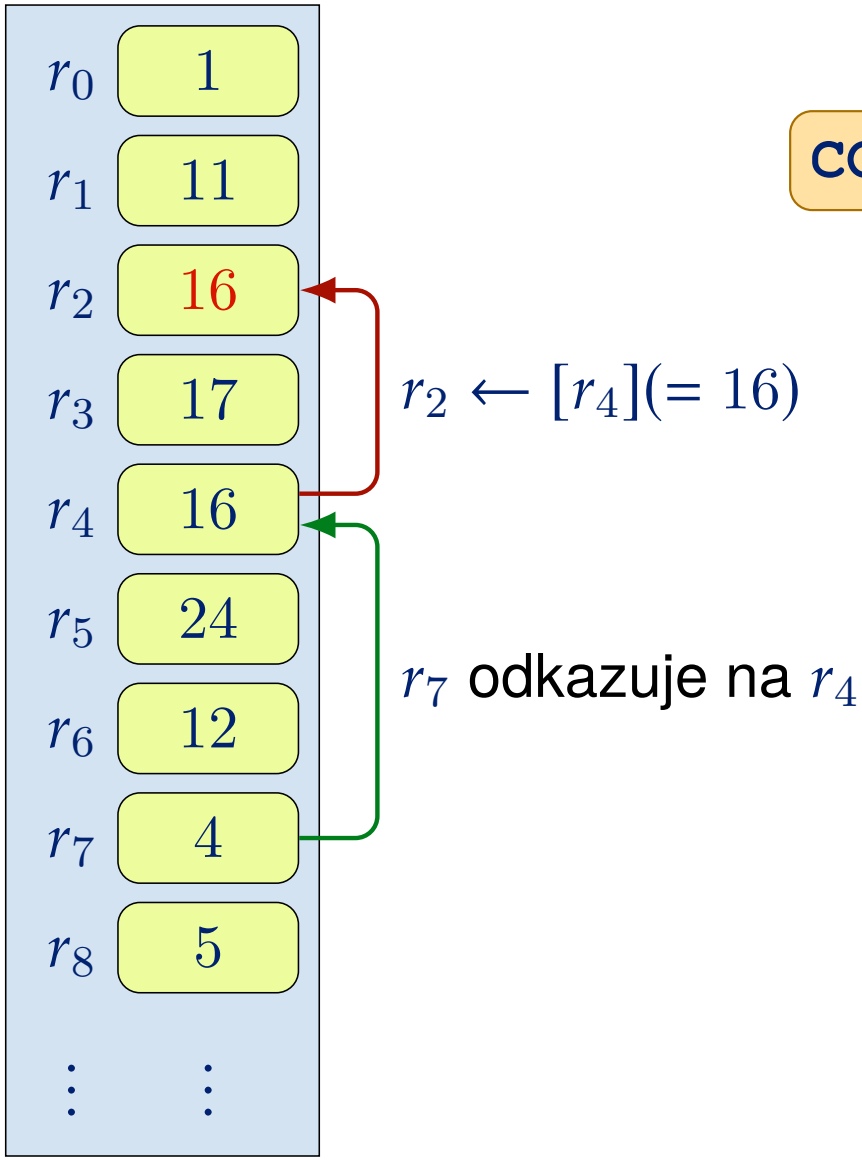
Random Access Machine (definice)

- Random Access Machine (RAM) se skládá z
  - řídící jednotky (procesoru, CPU) a
  - neomezené paměti
- Paměť RAMu je rozdělená do registrů r_i, $i \in \mathbb{N}$.
- V každém registru může být libovolné přirozené číslo
  - na začátku obsahují 0
- $[r_i]$ označuje obsah registru r_i
- Nepřímá adresace: $[\![r_i]\!] = [r_{[r_i]}]$
- Programem pro RAM je konečná posloupnost instrukcí $P = I_0, I_1, \ldots, I_\ell$
- Instrukce jsou vykonávané v pořadí daném programem

Instrukce RAM

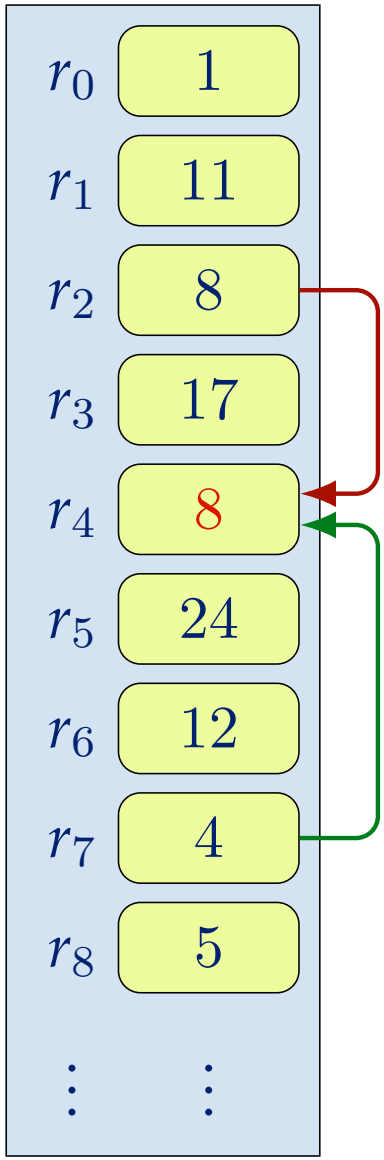
Instrukce	Efekt
LOAD(C, r_i)	$r_i \leftarrow C$
ADD(r_i, r_j, r_k)	$r_k \leftarrow [r_i] + [r_j]$
SUB(r_i, r_j, r_k)	$r_k \leftarrow \max([r_i] - [r_j], 0)$
COPY$([r_p], r_d)$	$r_d \leftarrow [\![r_p]\!]$
COPY$(r_s, [r_d])$	$r_{[r_d]} \leftarrow [r_s]$
JNZ(r_i, I_z)	**if** $[r_i] > 0$ **then goto** z
READ(r_i)	$r_i \leftarrow$ input
PRINT(r_i)	output $\leftarrow [r_i]$

Nepřímá adresace zdroje



COPY$([r_7], r_2)$ provede $r_2 \leftarrow [\![r_7]\!]$

Nepřímá adresace cíle



$r_4 \leftarrow [r_2](= 8)$

r_7 odkazuje na r_4

COPY$(r_2, [r_7])$ provede $r_{[r_7]} \leftarrow [r_2]$

RAM pro součin čísel

- Načte dvě čísla x a y ze vstupu a
- vypíše na výstup jejich součin $x \cdot y$

Algoritmus RAM pro výpočet součinu

```
1 READ(x)
2 READ(y)
3 z ← 0
4 while x > 0 do
5 |   z ← z + y
6 |   x ← x − 1
7 PRINT(z)
```

RAM pro součin čísel

Proměnným přiřadíme registry $r_0 \leftarrow x$, $r_1 \leftarrow y$, $r_2 \leftarrow z$.

Algoritmus RAM pro výpočet součinu

```
1 READ(r_0)  // x
2 READ(r_1)  // y
3 r_2 ← 0  // z ← 0
4 while [r_0] > 0 do
5 |   r_2 ← [r_2] + [r_1]
6 |   r_0 ← [r_0] − 1
7 PRINT(r_2)
```

RAM pro součin čísel

Nahradíme **while** cyklus podmínkami a skoky

Algoritmus RAM pro výpočet součinu

```
1 READ(r_0)
2 READ(r_1)
3 r_2 ← 0
4 if [r_0] > 0 then
5 |   r_2 ← [r_2] + [r_1]
6 |   r_0 ← [r_0] − 1
7 |   goto 4
8 PRINT(r_2)
```

RAM pro součin čísel

Přepíšeme jen s pomocí podmíněných a nepodmíněných skoků

Algoritmus RAM pro výpočet součinu

1 **READ**(r_0)
2 **READ**(r_1)
3 $r_2 \leftarrow 0$
4 **if** $[r_0] > 0$ **then goto** 6
5 **goto** 9
6 $r_2 \leftarrow [r_2] + [r_1]$
7 $r_0 \leftarrow [r_0] - 1$
8 **if** $[r_0] > 0$ **then goto** 6
9 **PRINT**(r_2)

RAM pro součin čísel

Přidáme pomocný registr s konstantou 1

Algoritmus RAM pro výpočet součinu

```
1  READ(r_0)
2  READ(r_1)
3  r_2 ← 0
4  r_3 ← 1
5  if [r_0] > 0 then goto 7
6  goto 10
7  r_2 ← [r_2] + [r_1]
8  r_0 ← [r_0] − [r_3]                    // r_0 ← [r_0] − 1
9  if [r_0] > 0 then goto 7
10 PRINT(r_2)
```

RAM pro součin čísel

Nahradíme nepodmíněný skok podmíněnými

Algoritmus RAM pro výpočet součinu

```
1  READ(r_0)
2  READ(r_1)
3  r_2 ← 0
4  r_3 ← 1
5  if [r_0] > 0 then goto 7
6  if [r_3] > 0 then goto 10                    // goto 10
7  r_2 ← [r_2] + [r_1]
8  r_0 ← [r_0] − [r_3]                          // r_0 ← [r_0] − 1
9  if [r_0] > 0 then goto 7
10 PRINT(r_2)
```

RAM pro součin čísel

Přepíšeme pomocí instrukcí RAM

Program RAM pro výpočet součinu

```
1 READ(r_0)
2 READ(r_1)
  // r_2 ← 0 není třeba
3 LOAD(1, r_3)                          // r_3 ← 1
4 JNZ(r_0, 6)              // if [r_0] > 0 then goto 6
5 JNZ(r_3, 9)                           // goto 9
6 ADD(r_2, r_1, r_2)              // r_2 ← [r_2] + [r_1]
7 SUB(r_0, r_3, r_0)                  // r_0 ← [r_0] − 1
8 JNZ(r_0, 6)              // if [r_0] > 0 then goto 6
9 PRINT(r_2)
```

Programování na RAMu

Programy pro RAM odpovídají procedurálnímu jazyku:

proměnné	skalární i neomezená pole
cykly	`for` i `while` s pomocí podmíněného skoku, případně čítače v proměnné
nepodmíněný skok	`goto` s použitím pomocného registru, kam uložíme `1` a použijeme podmíněný skok
podmíněný příkaz	s pomocí podmíněného skoku
funkce a procedury	inline, do místa použití funkce rovnou v programu napíšeme tělo funkce
nepřímá adresace	(**`COPY`**) umožňuje přístup k libovolně velké části paměti v závislosti na vstupu

Chybí rekurzivní volání funkcí, která lze implementovat pomocí cyklu `while` a zásobníku

Proměnné v programu pro RAM

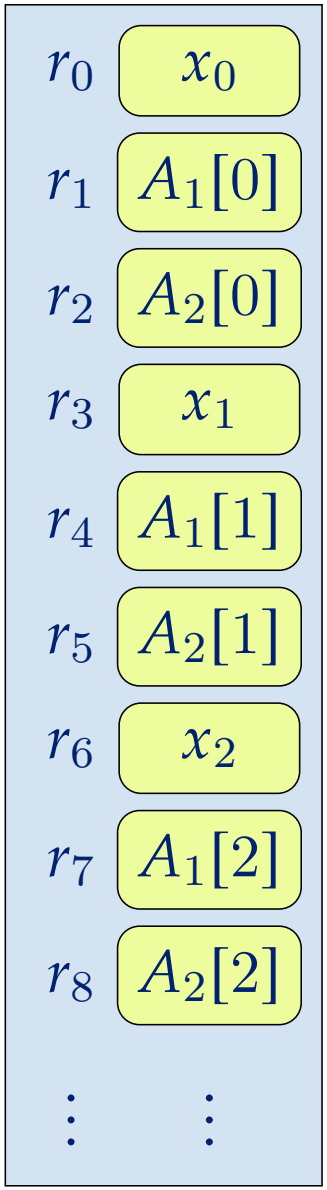
Předpokládejme, že v programu používáme pole $A_1, \ldots, A_p$ a skalární proměnné $x_0, \ldots, x_s$.

- Pole indexujeme přirozenými čísly (od 0)
- Prvek $A_i[j]$ umístíme do registru $r_{i+j*(p+1)}$
- Proměnnou x_i umístíme do registru $r_{i*(p+1)}$
- Prvky pole A_i jsou v registrech

$$r_i, r_{i+p+1}, r_{i+2(p+1)}, \ldots$$

- Skalární proměnné jsou v registrech

$$r_0, r_{p+1}, r_{2(p+1)}, \ldots$$



Jazyky rozhodnutelné RAMem

- Uvažme abecedu $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$.
- RAM R čte slovo $w = \sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_n}$ jako posloupnost čísel $i_1, \ldots, i_n$ zakončenou 0
- RAM R přijme slovo w, pokud $R(w)\downarrow$ a první číslo, které R zapíše na výstup je 1
- RAM R odmítne slovo w, pokud $R(w)\downarrow$ a R buď na výstup nezapíše nic, nebo první zapsané číslo je jiné než 1
- Jazyk slov přijímaných RAMem R označíme pomocí $L(R)$
- (RAMem) částečně rozhodnutelný jazyk = přijímán nějakým RAMem
- (RAMem) rozhodnutelný jazyk = přijímán nějakým RAMem, který se zastaví pro každý vstup
  - každé slovo buď přijme, nebo odmítne

Funkce vyčíslitelné na RAMu

O RAMu R řekneme, že počítá částečnou aritmetickou funkci $f : \mathbb{N}^n \to \mathbb{N}$, $n \geq 0$, pokud se vstupem $x_1, \ldots, x_n$ platí:

- Je-li $f(x_1, \ldots, x_n)\downarrow$, pak $R(x_1, \ldots, x_n)\downarrow$ a R vypíše na výstup hodnotu $f(x_1, \ldots, x_n)$
- Je-li $f(x_1, \ldots, x_n)\uparrow$, pak $R(x_1, \ldots, x_n)\uparrow$

Řetězcové funkce vyčíslitelné na RAMu

RAM R počítá částečnou funkci $f : \Sigma^* \to \Sigma^*$, kde $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$, pokud platí:

- Vstupní řetězec $w = \sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_n}$ je předaný jako posloupnost čísel $i_1, \ldots, i_n$ ukončený 0
- Pokud je $f(w)\downarrow = \sigma_{j_1}\sigma_{j_2}\ldots\sigma_{j_m}$, pak $R(w)\downarrow$ a na výstup je zapsaná posloupnost čísel $j_1, j_2, \ldots, j_m, 0$
- Pokud $f(w)\uparrow$, pak $R(w)\uparrow$

Turingův stroj $\longrightarrow$ RAM

Věta

Ke každému Turingovu stroji M existuje ekvivalentní RAM R.

- R simuluje práci M instrukci po instrukci
- R počítá touž funkci jako M
- R přijímá týž jazyk

Předpokládáme, že M má pásku neomezenou pouze doprava

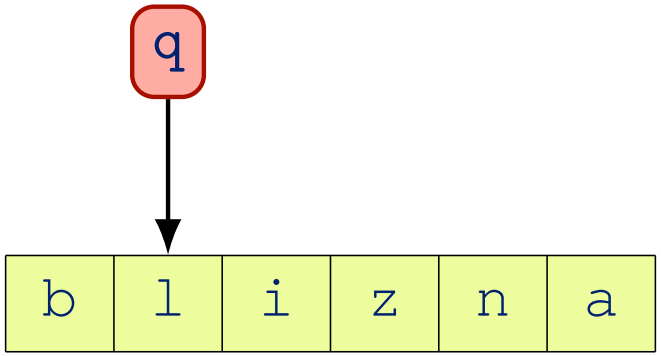
- M nikdy nepohne hlavou nalevo od nejlevějšího symbolu vstupu
- Lze předpokládat bez újmy na obecnosti — každý TS lze převést do této podoby (viz cvičení)

Konfigurace Turingova stroje

RAM R musí ve své paměti reprezentovat konfiguraci M.

Konfigurace zachycuje stav výpočtu Turingova stroje

- stav řídící jednotky
- slovo na pásce
  - od nejlevějšího do nejpravějšího neprázdného políčka
- pozici hlavy na pásce
  - v rámci slova na pásce



Technické předpoklady

$$M = (Q, \Sigma, \delta, q_0, F)$$

- $Q = \{q_0, q_1, \ldots, q_r\}$ pro nějaké $r \geq 0$, kde q_0 je počáteční stav
- $\Sigma = \{\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_s\}$ pro nějaké $s \geq 1$, kde $\sigma_0 = \lambda$ označuje znak prázdného políčka
- Nula ukončující vstup RAMu tedy odpovídá prázdnému políčku

Reprezentace konfigurace

obsah pásky je uložen v poli T

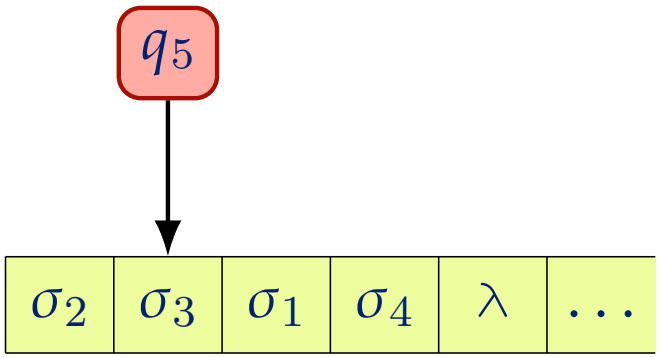
- Dle předpokladu je páska neomezená jen doprava
- $T[0]$ obsahuje první znak vstupu (po jeho načtení)

poloha hlavy v proměnné h

- $T[h]$ obsahuje symbol pod hlavou

stav v proměnné q

Konfigurace M



Reprezentace v R

$$q = 5$$

$$h = 1$$

$$T = \{2, 3, 1, 4, 0, \ldots\}$$

Algoritmus R

1. Načti vstup do pole T
   - Vstup je ukončený 0, která reprezentuje prázdné políčko λ
2. Polož $q = 0$, $h = 0$
3. Dokud $\delta(q, T[h])$ je definovaná, odsimuluj krok určený přechodovou funkcí
   - Přechodová funkce je uložená v programu R
   - Určení přechodu je provedeno posloupností podmíněných příkazů
   - Simulace kroku spočívá v aktualizaci $T[h]$, h a q dle instrukce
4. *Pokud nás zajímá přijetí slova*
   - je-li v q číslo přijímajícího stavu, zapiš 1 na výstup, jinak zapiš 0
5. *Pokud nás zajímá obsah pásky*
   - opiš obsah pásky na výstup

Přepis přechodové funkce do programu

Přechodová funkce M

q, c	$\rightarrow$	q', c', Z
q_0, σ_2	$\rightarrow$	q_3, σ_1, R
q_3, σ_1	$\rightarrow$	q_2, σ_0, L
q_2, σ_0	$\rightarrow$	q_0, σ_2, N

Odpovídající část programu R

if $q = 0$ **and** $T[h] = 2$ **then**
    $q \leftarrow 3$
    $T[h] \leftarrow 1$
    $h \leftarrow h + 1$
else if $q = 3$ **and** $T[h] = 1$ **then**
    $q \leftarrow 2$
    $T[h] \leftarrow 0$
    $h \leftarrow h - 1$
else if $q = 2$ **and** $T[h] = 0$ **then**
    $q \leftarrow 0$
    $T[h] \leftarrow 2$
else
    Konec simulace

RAM $\longrightarrow$ Turingův stroj

Věta

Ke každému RAMu R existuje ekvivalentní Turingův stroj M.

Obsah paměti R reprezentujeme na pásce M takto:

Jsou-li aktuálně využité registry $r_{i_1}, r_{i_2}, \ldots, r_{i_m}$, kde $i_1 < i_2 < \cdots < i_m$, pak je na pásce reprezentující paměť RAM R řetězec:

$$(i_1)_B|([r_{i_1}])_B\#(i_2)_B|([r_{i_2}])_B\#\ldots\#(i_m)_B|([r_{i_m}])_B$$

RAM $\longrightarrow$ Turingův stroj (struktura TS)

TS M bude mít 4 pásky

Vstupní páska posloupnost čísel, která má dostat R na vstup

- Čísla jsou zakódovaná binárně a oddělená znakem $\#$
- Z této pásky M jen čte

Výstupní páska sem zapisuje M čísla, která R zapisuje na výstup

- Čísla jsou zakódovaná binárně a oddělená znakem $\#$
- Na tuto pásku M jen zapisuje

Paměť RAM obsah paměti stroje R

Pomocná páska pro výpočty součtu, rozdílu, nepřímých adres, posunu části paměťové pásky a podobně

RAM $\longrightarrow$ Turingův stroj (přechodová funkce)

- Číslo prováděné instrukce (pořadí v programu) je uloženo ve stavu
- Každá instrukce R je provedena řetězcem instrukcí M
  - Nalezení registrů s operandy instrukce
  - Provedení aritmetických operací s operandy
  - Výpočet adres nepřímé adresace
  - Úprava paměti podle výsledku instrukce
    - Přidání nového registru do seznamu
    - Přepis obsahu některého registru
    - Může být nutné posunout obsah pásky s pamětí
- Následuje přechod do stavu, jímž začíná provádění další instrukce
- Pokud už další instrukce nenásleduje, simulace končí
  - Přijetí je dáno tím, jestli na výstupní pásku bylo zapsáno jen číslo 1
  - Toto je možné pamatovat si ve stavu

Číslování Turingových strojů

Definice

Definice

Jazyk $L \subseteq \Sigma^*$ je …

částečně rozhodnutelný je-li přijímán nějakým Turingovým strojem M

- $L = L(M)$

rozhodnutelný je-li přijímán nějakým Turingovým strojem M, jehož výpočet s každým vstupem se zastaví

- $L = L(M)$ a
- $(\forall x \in \Sigma^*)[M(x)\downarrow]$

- Částečně rozhodnutelný jazyk = rekurzivně spočetný jazyk.
- Rozhodnutelný jazyk = rekurzivní jazyk.

Kolik je částečně rozhodnutelných jazyků?

Jsou všechny jazyky nad konečnou abecedou Σ částečně rozhodnutelné?

Kolik je jazyků nad abecedou Σ?

Kolik je částečně rozhodnutelných jazyků nad abecedou Σ?

Shortlex uspořádání řetězců

- Uvažme abecedu Σ
- Předpokládejme, že $<$ je ostré uspořádání na znacích Σ
- $|u|$ označuje délku řetězce $u \in \Sigma^*$
- Řetězec $u \in \Sigma^*$ je menší než $v \in \Sigma^*$ v shortlex uspořádání, pokud
  1. $|u| < |v|$ (u je kratší než v), nebo
  2. $|u| = |v|$ a je-li i první index s $u[i] \neq v[i]$, pak $u[i] < v[i]$
- Tento fakt označíme pomocí $u \prec v$.
- Tím je dané i značení $u \preceq v$, $u \succ v$ a $u \succeq v$

Příklad

$$\texttt{Bob} \prec \texttt{Alena} \prec \texttt{Alice} \prec \texttt{Cyril} \prec \texttt{Andrea}$$

Číslování řetězců

- Každému řetězci $w \in \Sigma^*$ přiřadíme číslo

$$\mathrm{index}(w) = |\{u \in \Sigma^* \mid u \prec w\}|$$

- Porovnáváme nejprve délku $\Longrightarrow$ vždy konečné číslo
- $\mathrm{index}(w)$ je počet řetězců před w v shortlex uspořádání
- index je bijekcí mezi Σ^* a $\mathbb{N}$

Číslování binárních řetězců

- Uvažme binární abecedu $\Sigma = \{0, 1\}$ a řetězec $w \in \Sigma^*$
- $\mathrm{index}(w) = i$, kde

$$\underbrace{(i+1)_B}_{\text{binární zápis } i+1} = \underbrace{1w}_{\text{konkatenace } 1 \text{ a } w}$$

w	$1w$	$\mathrm{index}(w)+1$	$\mathrm{index}(w)$
ε	1	1	0
0	10	2	1
1	11	3	2
00	100	4	3
⋮	⋮	⋮	
001011	1001011	75	74
⋮	⋮	⋮	

Lze spočítat jazyky?

Definice

Množina A je spočetná, pokud existuje prostá funkce $f : A \to \mathbb{N}$, tj. pokud lze prvky A očíslovat.

- Jazyk $L \subseteq \Sigma^*$ odpovídá množině přirozených čísel

$$A = \{\mathrm{index}(w) \mid w \in L\}$$

- $\mathcal{P}(\mathbb{N})$ je nespočetná množina
  - Cantorova věta $\mathcal{P}(A)$ má větší mohutnost než A pro každou množinu A

Jazyků nad konečnou abecedou Σ není spočetně mnoho

Číslování Turingových strojů

Každému Turingovu stroji přiřadíme přirozené číslo

1. Turingův stroj popíšeme řetězcem nad malou abecedou
2. Řetězec nad touto abecedou převedeme do binární abecedy
3. Každému binárnímu řetězci w přiřadíme číslo $\mathrm{index}(w)$
4. Každému Turingovu stroji takto přiřadíme Gödelovo číslo

Převod do binární abecedy není pro číslování nutný, ale chceme, aby Univerzální Turingův stroj byl schopen simulovat sám sebe.

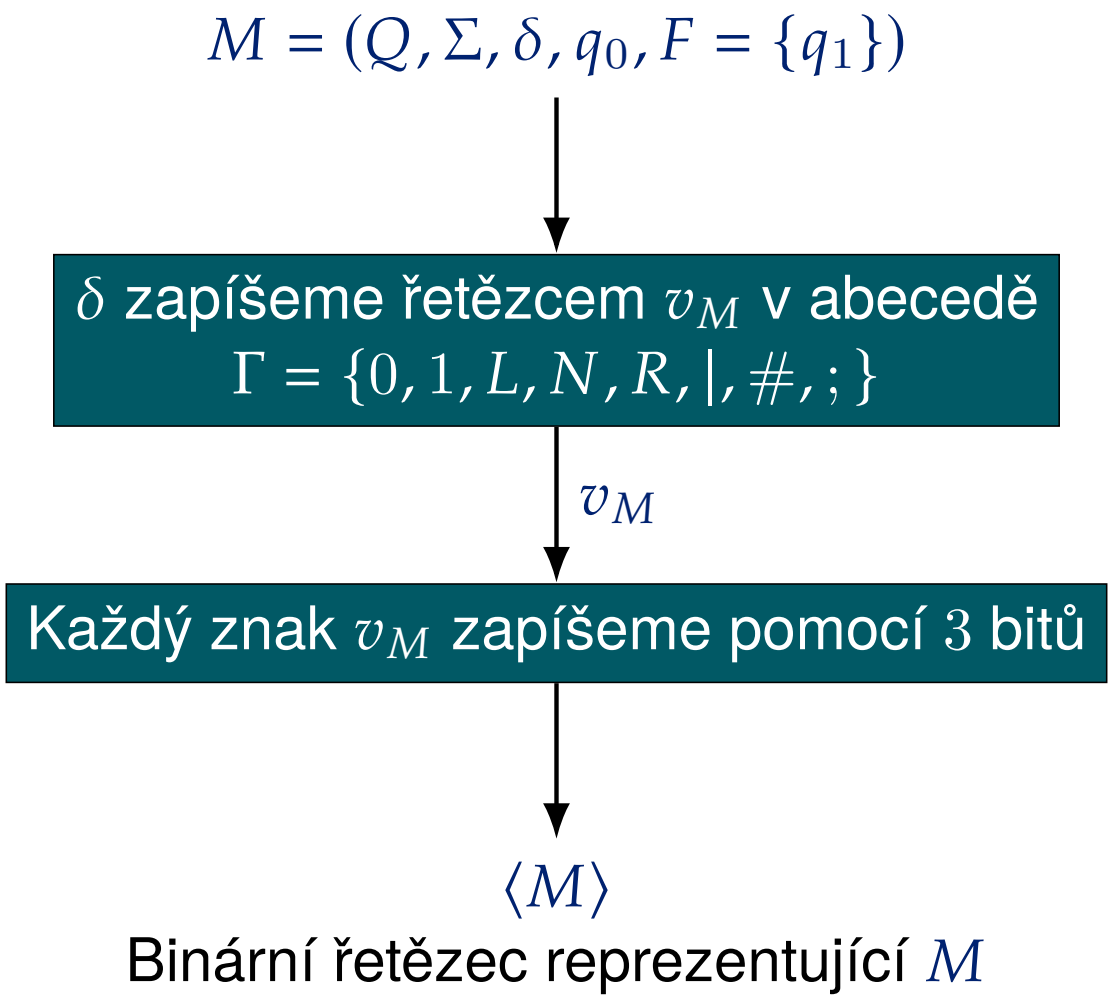
Pár technických omezení

Omezíme se na Turingovy stroje, které

1. mají jediný přijímající stav a
2. mají pouze binární vstupní abecedu $\Sigma_{in} = \{0, 1\}$.

- Vstupní řetězce budou zapsány jen pomocí znaků 0 a 1
- Pracovní abecedu nijak neomezujeme
  - Turingův stroj může během výpočtu zapisovat na pásku libovolné symboly
- Jakoukoli konečnou abecedu lze zakódovat do binární abecedy
- Každý TS M lze upravit tak, aby splňoval obě omezení

Zakódování přechodové funkce



Zápis přechodové funkce v abecedě Γ

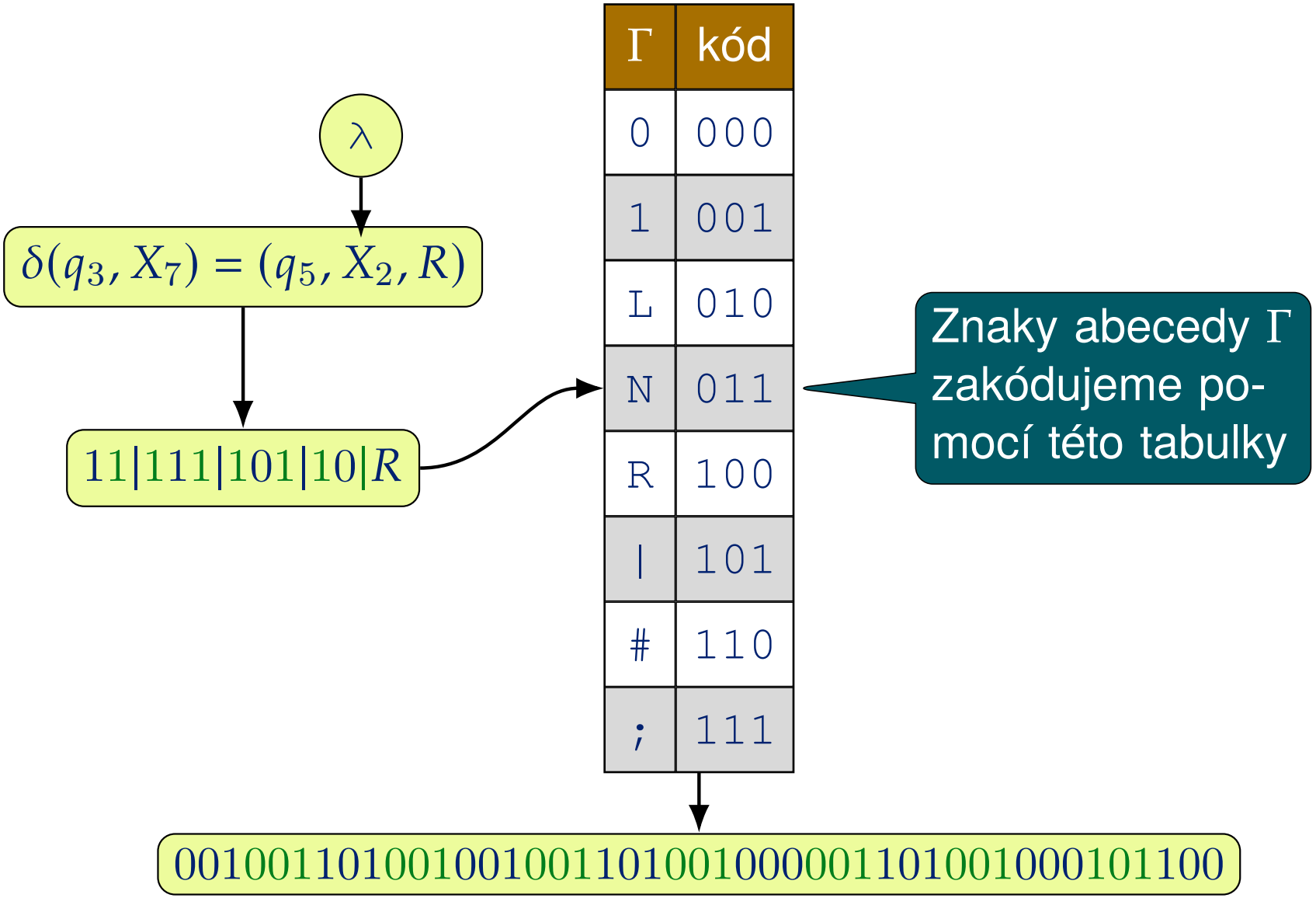
- Předpokládejme, že
  - $Q = \{q_0, q_1, \ldots, q_r\}$ pro nějaké $r \geq 1$, kde q_0 je počáteční stav a q_1 je jediný přijímající stav.
  - $\Sigma = \{X_0, X_1, X_2, \ldots, X_s\}$ pro nějaké $s \geq 2$, kde $X_0 = 0$, $X_1 = 1$ a $X_2 = \lambda$
- Instrukci $\delta(q_i, X_j) = (q_k, X_l, Z)$, kde $Z \in \{L, N, R\}$ zakódujeme řetězcem

$$(i)_B|(j)_B|(k)_B|(l)_B|Z$$

- Jsou-li $C_1, \ldots, C_n$ kódy instrukcí TS M, pak přechodovou funkci δ zakódujeme řetězcem

$$C_1 \# C_2 \# \ldots \# C_n$$

Převod do binární abecedy



Gödelovo číslo

$\langle M \rangle$ binární řetězec kódující TS M

Gödelovo číslo jednoznačně přiřazené danému Turingovu stroji

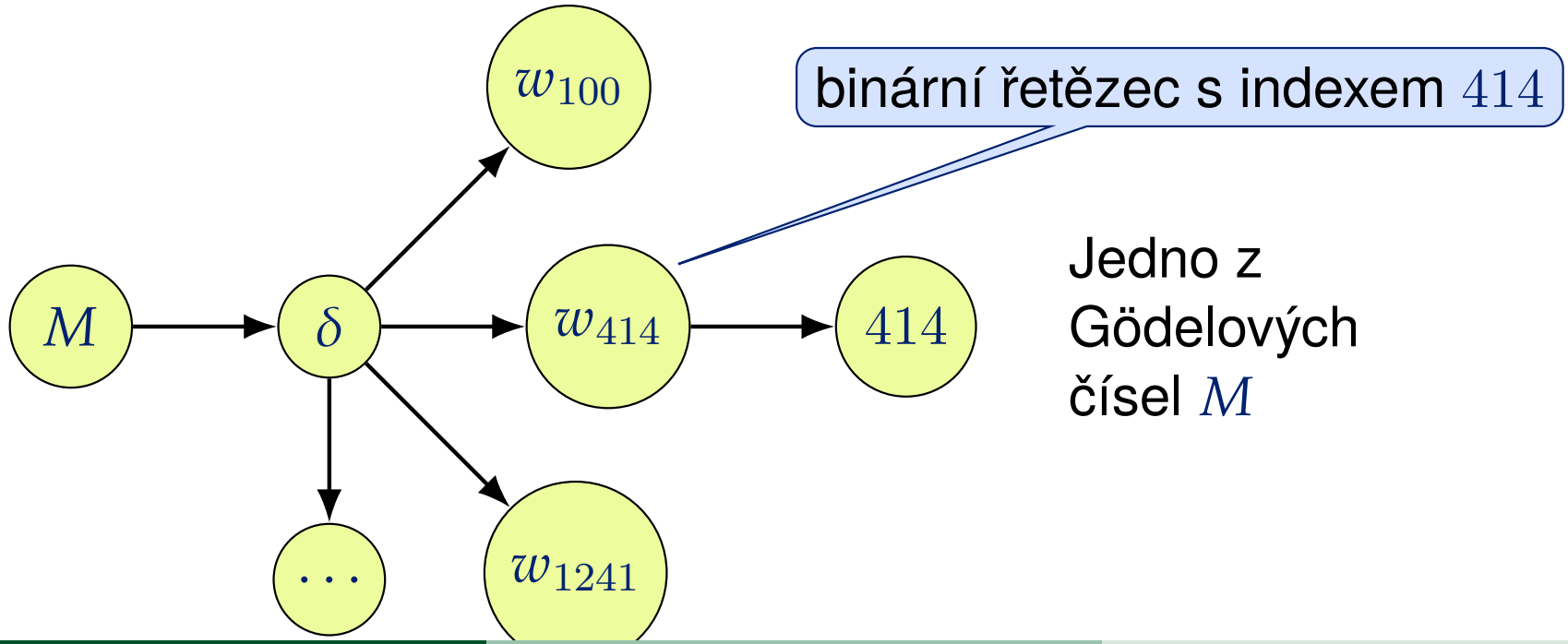
- Definujeme jako $\text{index}(\langle M \rangle)$

Je-li w řetězec, který není syntakticky správným kódem Turingova stroje, přiřadíme mu Turingův stroj M s prázdnou přechodovou funkcí.

- přechodová funkce M není definovaná pro žádný vstup a
- M okamžitě odmítne každý vstup, tedy $L(M) = \emptyset$

Nejednoznačnost kódu TS

- Kód TS není jednoznačný, protože nezáleží na
  - pořadí instrukcí,
  - na očíslování stavů kromě počátečního a přijímajícího,
  - znaků páskové abecedy kromě 0, 1, λ, a
  - binární zápis čísla stavu nebo znaku může být uvozen libovolným počtem 0.
- Každý TS má nekonečně mnoho různých kódů a potažmo nekonečně mnoho Gödelových čísel.



Kolik je částečně rozhodnutelných jazyků?

- Je jen spočetně mnoho Turingových strojů
  - každý má Gödelovo číslo
  - každé číslo odpovídá jedinému Turingovu stroji
- Každý částečně rozhodnutelný jazyk je přijímán nějakým Turingovým strojem

Lemma

Částečně rozhodnutelných jazyků je spočetně mnoho.

- Všech jazyků nad konečnou abecedou je nespočetně mnoho

Musí proto existovat jazyky nad abecedou $\{0,1\}$, které nejsou částečně rozhodnutelné.

Kódování objektů (značení)

- Konečné objekty (např. číslo, řetězec, Turingův stroj, RAM, graf nebo formuli) můžeme kódovat binárními řetězci
- Podobně můžeme zakódovat i n-tice objektů

Definice

$\langle X \rangle$ binární řetězec kódující objekt X

$\langle X_1, \ldots, X_n \rangle$ binární řetězec kódující n-tici objektů $X_1, \ldots, X_n$

Příklad

$\langle M \rangle$ kód Turingova stroje M

$\langle M, x \rangle$ kód dvojice tvořené Turingovým strojem M a řetězcem x