

IB107 Vyčísitelnost a složitost

standardní numerace, problém zastavení, věta o numeraci

Jan Strejček

Fakulta informatiky
Masarykova univerzita

Definice (numerace množiny)

*Numerace množiny M je surjektivní funkce $\nu : \mathbb{N} \rightarrow M$.
Je-li ν navíc totální, mluvíme o **totální numeraci množiny**.*

příklady

- numerace množiny $M = \{a, b, c\}$

- totální numerace množiny $M = \{a, b, c\}$

- numerace množiny $\{0, 2, 4, 6, 8, \dots\}$

- numerace množin \mathbb{Z} a \mathbb{Q}

- numerovat lze libovolnou nejvýše spočetnou množinu
- chceme definovat totální numeraci j -árních vyčíslitelných funkcí $\mathcal{P}^{(j)}$ (pro každé $j \geq 1$)

- každému programu přiřadíme číslo z \mathbb{N}
- zvolíme techniku, kterou použil Kurt Gödel
- předpokládáme, že programy používají pouze proměnné x_1, x_2, \dots
- definujeme funkci *code* : $\mathbb{P} \rightarrow \mathbb{N}$ pro všechna $i, j, m \geq 1$ a libovolné příkazy $\delta, \delta_1, \delta_2, \dots$

$$\text{code}(x_i := 0) = 2^i$$

$$\text{code}(x_i := x_j - 1) = 3^i 5^j$$

$$\text{code}(x_i := x_j + 1) = 7^i 11^j$$

$$\text{code}(\text{while } x_i \neq x_j \text{ do } \delta) = 13^i 17^j 19^{\text{code}(\delta)}$$

$$\text{code}(\text{begin } \delta_1; \delta_2; \dots \delta_m \text{ end}) = 23^{\text{code}(\delta_1)} 29^{\text{code}(\delta_2)} \dots (p_{8+m})^{\text{code}(\delta_m)}$$

$$\text{code}(\text{begin end}) = 1$$

kde p_i je i -té prvočíslo, přičemž první prvočíslo je 2

- *code*(P) nazýváme *index* programu P

$$\text{code}(x_i := 0) = 2^i$$

$$\text{code}(x_i := x_j - 1) = 3^i 5^j$$

$$\text{code}(x_i := x_j + 1) = 7^i 11^j$$

$$\text{code}(\text{while } x_i \neq x_j \text{ do } \delta) = 13^i 17^j 19^{\text{code}(\delta)}$$

$$\text{code}(\text{begin } \delta_1; \delta_2; \dots \delta_m \text{ end}) = 23^{\text{code}(\delta_1)} 29^{\text{code}(\delta_2)} \dots (p_{8+m})^{\text{code}(\delta_m)}$$

$$\text{code}(\text{begin end}) = 1$$

P_{empty} :

$$\text{code}(P_{\text{empty}}) =$$

begin

$x_1 := 0;$

$x_2 := x_1 + 1;$

while $x_1 \neq x_2$ **do begin end**

end

Lemma

Funkce code : $\mathbb{P} \rightarrow \mathbb{N}$ je injektivní a totální.

Důkaz:



Tvrzení

Funkce code : $\mathbb{P} \rightarrow \mathbb{N}$ není surjektivní.

Důkaz: např. číslo 12 není indexem žádného programu.



standardní numerace programů

P_{empty} : **begin**
 $x_1 := 0$;
 $x_2 := x_1 + 1$;
 while $x_1 \neq x_2$ **do begin end**
end

Definice (standardní numerace programů)

Standardní (kanonická) numerace programů je funkce
 $num : \mathbb{N} \mapsto \mathbb{P}$ *definovaná takto:*

$$num(n) = \begin{cases} code^{-1}(n) & \text{jestliže } n \in range(code) \\ P_{empty} & \text{jinak} \end{cases}$$

Namísto $num(0), num(1), \dots$ obvykle píšeme P_0, P_1, \dots

Definice (standardní numerace vyčíslitelných funkcí)

Nechť $j \geq 1$. *Standardní (kanonická) numerace j -árních vyčíslitelných funkcí je funkce $\varphi^{(j)} : \mathbb{N} \mapsto \mathcal{P}^{(j)}$ definovaná takto:*

$$\varphi^{(j)}(i) = \varphi_{\text{num}(i)}^{(j)}$$

Index vyčíslitelné funkce $f \in \mathcal{P}^{(j)}$ je číslo i splňující $f = \varphi^{(j)}(i)$.

Místo $\varphi^{(j)}(0), \varphi^{(j)}(1), \dots$ obvykle píšeme $\varphi_0^{(j)}, \varphi_1^{(j)}, \dots$

Pro $j = 1$ píšeme jen $\varphi_0, \varphi_1, \dots$

Vyčíslitelných j -árních funkcí je pouze spočetně mnoho.

Všech vyčíslitelných funkcí je pouze spočetně mnoho.

Lemma

Každá vyčíslitelná funkce má nekonečně mnoho různých indexů.

Důkaz:



- existují funkce, které nejsou vyčíslitelné
- neexistuje algoritmus rozhodující, zda daný program na svém indexu zastaví

Věta

Funkce $f : \mathbb{N} \mapsto \mathbb{N}$ definovaná vztahem

$$f(i) = \begin{cases} 1 & \text{jestliže } \varphi_i(i) \text{ je definováno} \\ 0 & \text{jestliže } \varphi_i(i) \text{ není definováno} \end{cases}$$

není vyčíslitelná.

důkaz nerozhodnutelnosti problému zastavení

$$f(i) = \begin{cases} 1 & \text{jestliže } \varphi_i(i) \text{ je definováno} \\ 0 & \text{jestliže } \varphi_i(i) \text{ není definováno} \end{cases}$$

Důkaz: (sporem) Nechť f je počítaná programem *halt*. Pak program *confuse*

```
begin  
  while  $halt(x_1) = 1$  do begin end;  
   $x_1 := 1$   
end
```

počítá funkci

$$\psi(i) = \begin{cases} \perp & \text{jestliže } f(i) = 1 \\ 1 & \text{jestliže } f(i) = 0 \end{cases}$$

Nechť e je index programu *confuse*. Pak

$$\psi(e) = \varphi_e(e) = \begin{cases} \perp \\ 1 \end{cases}$$

diagonalizační metoda

	0	1	2	...	n	...
P_0	↑	↓	↓	...	↑	...
P_1	↑	↓	↑	...	↓	...
P_2	↓	↓	↓	...	↓	...
⋮	⋮	⋮	⋮	⋱	⋮	
P_n	↓	↓	↑	...	↑	...
⋮	⋮	⋮	⋮		⋮	⋱
<i>confuse</i>	↓	↑	↑	...	↓	...

Věta (věta o numeraci)

Pro každé $j \geq 1$ existuje vyčíslitelná funkce $\Phi : \mathbb{N}^{j+1} \rightarrow \mathbb{N}$ taková, že pro všechna $e, a_1, \dots, a_j \in \mathbb{N}$ platí

$$\Phi(e, a_1, \dots, a_j) = \varphi_e^{(j)}(a_1, \dots, a_j).$$

- Φ nazýváme **univerzální** funkce pro standardní numeraci j -árních vyčíslitelných funkcí
- větě se také říká **utm-věta** (universal Turing machine)

intuitivní implementace univerzální funkce Φ

- 1 ze vstupu (e, a_1, \dots, a_j) dekoduj program P_e
- 2 simuluj program P_e na vstupu (a_1, \dots, a_j)
- 3 pokud simulovaný program P_e skončí, dej na výstup jeho výslednou hodnotu

úskalí intuitivní implementace

- simulovaný program může obsahovat libovolně velký (konečný) počet proměnných
 - implementace Φ má ale fixní počet proměnných
- kódování více proměnných do jedné

Definice (párující funkce)

Totálně vyčíslitelné bijekci $f : \mathbb{N}^2 \mapsto \mathbb{N}$ říkáme párující funkce.

(0, 0)	(0, 1)	(0, 2)	(0, 3)	...
(1, 0)	(1, 1)	(1, 2)	(1, 3)	...
(2, 0)	(2, 1)	(2, 2)	(2, 3)	...
(3, 0)	(3, 1)	(3, 2)	(3, 3)	...
⋮	⋮	⋮	⋮	⋮

párující funkce

(0, 0)	(0, 1)	(0, 2)	(0, 3)	...
(1, 0)	(1, 1)	(1, 2)	(1, 3)	...
(2, 0)	(2, 1)	(2, 2)	(2, 3)	...
(3, 0)	(3, 1)	(3, 2)	(3, 3)	...
⋮	⋮	⋮	⋮	⋱

Lemma

Funkce $\tau : \mathbb{N}^2 \mapsto \mathbb{N}$ definovaná vztahem

$$\tau(i, j) = \frac{(i+j)(i+j+1)}{2} + i$$

je párující funkce.

Důkaz: τ je bijekce (viz konstrukce) a totálně vyčíslitelná. ■

Lemma

Funkce $\pi_1 : \mathbb{N} \mapsto \mathbb{N}$ a $\pi_2 : \mathbb{N} \mapsto \mathbb{N}$ definované jako

$$\pi_1(\tau(i, j)) = i \quad \text{a} \quad \pi_2(\tau(i, j)) = j$$

jsou totálně vyčíslitelné funkce. Funkce nazýváme *projekce*.

Důkaz: program pro π_1

begin

$x_2 := 0; x_3 := 0;$

while $\tau(x_2, 0) < x_1$ **do** $x_2 := x_2 + 1;$

while $\tau(x_2, x_3) \neq x_1$ **do begin** $x_2 := x_2 - 1; x_3 := x_3 + 1$ **end;**

$x_1 := x_2$

end



Definice (standardní kódovací funkce)

Pro všechna $k \geq 1$ definujeme funkce $\tau_k : \mathbb{N}^k \mapsto \mathbb{N}$ následně:

$$\begin{aligned}\tau_1(i_1) &= i_1 \\ \tau_k(i_1, \dots, i_k) &= \tau(\tau_{k-1}(i_1, \dots, i_{k-1}), i_k) \quad \text{pro } k > 1\end{aligned}$$

Tyto funkce nazýváme **standardní kódovací funkce**.

Odpovídající **projekce** $\pi_{kl} : \mathbb{N} \mapsto \mathbb{N}$ jsou pro všechna $1 \leq l \leq k$ definovány takto:

$$\begin{aligned}\pi_{k1}(\tau_k(i_1, \dots, i_k)) &= i_1 \\ &\vdots \\ \pi_{kk}(\tau_k(i_1, \dots, i_k)) &= i_k\end{aligned}$$

Lemma

Existuje pevné číslo n takové, že pro každé $k, l \in \mathbb{N}$ splňující $1 \leq l \leq k$ je funkce π_{kl} vyčíslitelná programem, který používá nejvýše n proměnných.

Důkaz:

- nechť m je počet proměnných potřebný k výpočtu τ
- programy počítající π_1, π_2 vystačí s $m + 3$ proměnnými
- π_{kl} lze počítat pomocí π_1, π_2

$$\begin{aligned}\pi_{k1}(i) &= \pi_1^{k-1}(i) \\ \pi_{kl}(i) &= \pi_2(\pi_1^{k-l}(i)) \quad \text{pro } l > 1\end{aligned}$$

Lemma

Ke každému $j \geq 1$ existuje číslo r a totálně vyčíslitelná funkce *short* : $\mathbb{N} \mapsto \mathbb{N}$ taková, že

- $\varphi_e^{(j)} = \varphi_{short(e)}^{(j)}$
- a program $P_{short(e)}$ používá nejvýše $j + r$ proměnných

Důkaz: Necht program P_e používá proměnné x_1, \dots, x_k .

Program $P_{short(e)}$

- zakóduje tyto proměnné do proměnné U
- simuluje program P_e s využitím projekcí
- používá pouze
 - j vstupních proměnných
 - m proměnných pro výpočet τ
 - n proměnných pro výpočet projekcí (viz předchozí lemma)
 - pomocné proměnné U, V, W

program $P_{short(e)}$:

begin

$U := \tau(x_1, x_2);$

$U := \tau(U, x_3);$

\vdots

$U := \tau(U, x_j);$

$U := \tau(U, 0);$

\vdots

$U := \tau(U, 0);$

} $(k - j)$ -krát

modifikace programu P_e

$x_1 := \pi_{k1}(U)$

end

modifikace programu P_e :

```
begin  
   $V := \pi_{kl}(U);$   
   $V := V + 1;$   
   $W := \pi_{k1}(U);$   
   $W := \tau(W, \pi_{k2}(U));$   
   $W := \tau(W, \pi_{k3}(U));$   
  
   $x_j := x_l + 1$   
   $\vdots$   
   $W := \tau(W, V);$     //  $V$  místo  $\pi_{ki}(U)$   
   $\vdots$   
   $W := \tau(W, \pi_{kk}(U));$   
   $U := W$   
end
```

modifikace programu P_e :

```
begin  
   $V := \pi_{ki}(U)$ ;  
   $W := \pi_{kl}(U)$ ;  
  while  $V \neq W$  do begin  
    modifikovaná verze  $\delta$ ;  
     $V := \pi_{ki}(U)$ ;  
     $W := \pi_{kl}(U)$   
  end  
end
```

while $x_j \neq x_l$ do δ

- program $P_{short(e)}$ počítá tutéž j -arní funkci jako P_e
- funkce *short* spočítá z indexu programu P_e index programu $P_{short(e)}$, tj. provede zde uvedenou konstrukci ■

další úskalí implementace univerzální funkce Φ

- není snadno patrné, v jakém pořadí instrukce vykonávat (Pokračovat další instrukcí nebo se vrátit k testu? Ke kterému testu?)
- program proto převedeme na čtveřice (návěští, instrukce, návěští pro false, návěští pro true)

begin

while $x_1 \neq x_2$ **do**

while $x_1 \neq x_4$ **do**

$x_1 := x_1 + 1;$

$x_2 := x_4$

end

→

(1, $x_1 \neq x_2$, 4, 2)

(2, $x_1 \neq x_4$, 1, 3)

(3, $x_1 := x_1 + 1$, 2, 2)

(4, $x_2 := x_4$, 5, 5)

- konečné posloupnosti čtveřic lze kódovat přirozenými čísly

Lemma

Existuje totálně vyčíslitelná funkce *quad* : $\mathbb{N} \mapsto \mathbb{N}$ taková, že *quad*(*e*) je kód posloupnosti čtveřic reprezentující program P_e .

Nechť *LIST* je kód posloupnosti čtveřic. Důkaz věty o numeraci používá i tyto totálně vyčíslitelné funkce:

max(*LIST*) vrátí nejvyšší návěští v *LIST* (tj. konec programu)

fetch(*LIST*, *LBL*) najde v *LIST* čtveřici s návěštěm *LBL* a vrátí kód odpovídající instrukce

next(*LIST*, *LBL*, *BLN*) najde v *LIST* čtveřici s návěštěm *LBL* a vrátí návěští pro false pokud *BLN* = 0 nebo návěští pro true pokud *BLN* = 1

důkaz věty o numeraci

Důkaz: program počítající univerzální funkci $\Phi : \mathbb{N}^{j+1} \rightarrow \mathbb{N}$:

begin

$E := \text{short}(x_1); x_1 := x_2; x_2 := x_3; \dots x_j := x_{j+1}; x_{j+1} := 0;$

$LIST := \text{quad}(E);$

$LBL := 1;$

$MAX := \text{max}(LIST);$

while $LBL \neq MAX$ **do begin**

$INST := \text{fetch}(LIST, LBL);$

$BLN := 0;$

if $INST = \text{code}(x_1 := 0)$ **then** $x_1 := 0;$

\vdots

if $INST = \text{code}(x_k := 0)$ **then** $x_k := 0;$

if $INST = \text{code}(x_1 := x_1 + 1)$ **then** $x_1 := x_1 + 1;$

if $INST = \text{code}(x_1 := x_1 - 1)$ **then** $x_1 := x_1 - 1;$

if $INST = \text{code}(x_1 := x_2 + 1)$ **then** $x_1 := x_2 + 1;$

if $INST = \text{code}(x_1 := x_2 - 1)$ **then** $x_1 := x_2 - 1;$

\vdots

if $INST = \text{code}(x_k := x_k + 1)$ **then** $x_k := x_k + 1;$

if $INST = \text{code}(x_k := x_k - 1)$ **then** $x_k := x_k - 1;$

if $INST = \text{code}(x_1 \neq x_2) \wedge x_1 \neq x_2$ **then** $BLN := 1;$

if $INST = \text{code}(x_1 \neq x_3) \wedge x_1 \neq x_3$ **then** $BLN := 1;$

\vdots

if $INST = \text{code}(x_{k-1} \neq x_k) \wedge x_{k-1} \neq x_k$ **then** $BLN := 1;$

$LBL := \text{next}(LIST, LBL, BLN);$

end

end

