Paralelné algoritmy, časť č. 10

František Mráz

Kabinet software a výuky informatiky, MFF UK, Praha

Paralelné algoritmy, 2011/2012

Obsah

Outline

1. P-úplnosť a ťažko paralelizovateľné úlohy
   - Paralelizovateľnosť
   - Problém generovateľnosti
   - P-úplnosť problému generovateľnosti
   - Problém CVP
   - CFE – problém prázdneho slova v bezkontextovom jazyku

Kedy je problém paralelizovateľný?

(1) doba paralelného výpočtu klesá s rastúcim počtom procesorov p v istom rozsahu $1 \leq p \leq a(n)$, kde $a(n)$ je rastúca funkcia

budeme používať (2)

Kedy je problém paralelizovateľný?

(1) doba paralelného výpočtu klesá s rastúcim počtom procesorov p v istom rozsahu $1 \leq p \leq a(n)$, kde $a(n)$ je rastúca funkcia

- v predchádzajúcich prednáškach väčšinou $1 \leq p \leq \frac{n}{\log^k n}$, pre pevné k

budeme používať (2)

Kedy je problém paralelizovateľný?

(1) doba paralelného výpočtu klesá s rastúcim počtom procesorov p v istom rozsahu $1 \leq p \leq a(n)$, kde $a(n)$ je rastúca funkcia

- v predchádzajúcich prednáškach väčšinou $1 \leq p \leq \frac{n}{\log^k n}$, pre pevné k
- používať $p > a(n)$ procesorov nič neprináša

budeme používať (2)

Kedy je problém paralelizovateľný?

(1) doba paralelného výpočtu klesá s rastúcim počtom procesorov p v istom rozsahu $1 \leq p \leq a(n)$, kde $a(n)$ je rastúca funkcia

- v predchádzajúcich prednáškach väčšinou $1 \leq p \leq \frac{n}{\log^k n}$, pre pevné k
- používať $p > a(n)$ procesorov nič neprináša
- malé $a(n)$ ruší výhody paralelizácie

budeme používať (2)

Kedy je problém paralelizovateľný?

(1) doba paralelného výpočtu klesá s rastúcim počtom procesorov p v istom rozsahu $1 \leq p \leq a(n)$, kde $a(n)$ je rastúca funkcia
- v predchádzajúcich prednáškach väčšinou $1 \leq p \leq \frac{n}{\log^k n}$, pre pevné k
- používať $p > a(n)$ procesorov nič neprináša
- malé $a(n)$ ruší výhody paralelizácie

(2) problém je možné riešiť extrémne rýchle s rozumným počtom procesorov – polylogaritmický čas a polynomiálny počet procesorov – NC

budeme používať (2)

Pozor algoritmus z NC neznamená vždy zrýchlenie

- určenie stromu prehľadávania do **šírky** pre neorientovaný graf sa dá urobiť s použitím algoritmu na výpočet najkratších vzdialeností medzi všetkými dvojicami vrcholov – čas $O(\log^2 n)$ s $O(\frac{n^3}{\log n})$ procesormi na COMMON

Pozor algoritmus z NC neznamená vždy zrýchlenie

- určenie stromu prehľadávania do **šírky** pre neorientovaný graf sa dá urobiť s použitím algoritmu na výpočet najkratších vzdialeností medzi všetkými dvojicami vrcholov – čas $O(\log^2 n)$ s $O(\frac{n^3}{\log n})$ procesormi na COMMON
- čas $O(\frac{n^3 \log n}{p} + \log^2 n)$ s p procesormi

Pozor algoritmus z NC neznamená vždy zrýchlenie

- určenie stromu prehľadávania do **šírky** pre neorientovaný graf sa dá urobiť s použitím algoritmu na výpočet najkratších vzdialeností medzi všetkými dvojicami vrcholov – čas $O(\log^2 n)$ s $O(\frac{n^3}{\log n})$ procesormi na COMMON
- čas $O(\frac{n^3 \log n}{p} + \log^2 n)$ s p procesormi
- najlepší sekvenčný algoritmus s časovou zložitosťou $O(m + n)$ je však rýchlejší pre $p = o(n \log n)$!

Triedy zložitosti

obvykle definované cez jazyky; rozhodovacie problémy $w \overset{?}{\in} L$

NC trieda všetkých jazykov L ; $\exists$ algoritmus pre PRAM, ktorý pre dané slovo w dĺžky n rozhoduje či $w \in L$ v čase $O(\log^k n)$ s polynomiálnym počtom procesorov, k je konštanta nezávislá na n.

Triedy zložitosti

obvykle definované cez jazyky; rozhodovacie problémy $w \overset{?}{\in} L$

NC trieda všetkých jazykov L ; $\exists$ algoritmus pre PRAM, ktorý pre dané slovo w dĺžky n rozhoduje či $w \in L$ v čase $O(\log^k n)$ s polynomiálnym počtom procesorov, k je konštanta nezávislá na n.

P trieda všetkých jazykov rozpoznateľných (sekvenčným) Turingovým strojom v polynomiálnom čase

Triedy zložitosti

obvykle definované cez jazyky; rozhodovacie problémy $w \overset{?}{\in} L$

NC trieda všetkých jazykov L ; $\exists$ algoritmus pre PRAM, ktorý pre dané slovo w dĺžky n rozhoduje či $w \in L$ v čase $O(\log^k n)$ s polynomiálnym počtom procesorov, k je konštanta nezávislá na n.

P trieda všetkých jazykov rozpoznateľných (sekvenčným) Turingovým strojom v polynomiálnom čase

- P je robustná trieda – môže sa definovať rovnako i pre RAM a iné sekvenčné modely algoritmov

Triedy zložitosti

obvykle definované cez jazyky; rozhodovacie problémy $w \stackrel{?}{\in} L$

NC trieda všetkých jazykov L ; $\exists$ algoritmus pre PRAM, ktorý pre dané slovo w dĺžky n rozhoduje či $w \in L$ v čase $O(\log^k n)$ s polynomiálnym počtom procesorov, k je konštanta nezávislá na n.

P trieda všetkých jazykov rozpoznateľných (sekvenčným) Turingovým strojom v polynomiálnom čase

- P je robustná trieda – môže sa definovať rovnako i pre RAM a iné sekvenčné modely algoritmov
- triviálne NC $\subseteq$ P – sekvenčná simulácia NC-algoritmu

Triedy zložitosti

obvykle definované cez jazyky; rozhodovacie problémy $w \stackrel{?}{\in} L$

NC trieda všetkých jazykov L ; $\exists$ algoritmus pre PRAM, ktorý pre dané slovo w dĺžky n rozhoduje či $w \in L$ v čase $O(\log^k n)$ s polynomiálnym počtom procesorov, k je konštanta nezávislá na n.

P trieda všetkých jazykov rozpoznateľných (sekvenčným) Turingovým strojom v polynomiálnom čase

- P je robustná trieda – môže sa definovať rovnako i pre RAM a iné sekvenčné modely algoritmov
- triviálne NC $\subseteq$ P – sekvenčná simulácia NC-algoritmu
- P $\subseteq$ NC – otvorený problém

Triedy zložitosti

obvykle definované cez jazyky; rozhodovacie problémy $w \overset{?}{\in} L$

NC trieda všetkých jazykov L ; $\exists$ algoritmus pre PRAM, ktorý pre dané slovo w dĺžky n rozhoduje či $w \in L$ v čase $O(\log^k n)$ s polynomiálnym počtom procesorov, k je konštanta nezávislá na n.

P trieda všetkých jazykov rozpoznateľných (sekvenčným) Turingovým strojom v polynomiálnom čase

- P je robustná trieda – môže sa definovať rovnako i pre RAM a iné sekvenčné modely algoritmov
- triviálne NC $\subseteq$ P – sekvenčná simulácia NC-algoritmu
- P $\subseteq$ NC – otvorený problém
- usudzuje sa, že P $\subsetneq$ NC, ale nikto to nevie dokázať; aspoň boli nájdení kandidáti na problémy z NC $\setminus$ P

NC-prevediteľnosť

L_1, L_2 jazyky

- L_1 je NC-prevediteľný na L_2 ($L_1 \leq_{NC} L_2$)$\equiv_{df}$ $\exists$ NC-algoritmus, ktorý transformuje ľubovoľné slovo u_1 na slovo u_2 také, že

$$u_1 \in L_1 \Leftrightarrow u_2 \in L_2$$

NC-prevediteľnosť

L_1, L_2 jazyky

- L_1 je NC-prevediteľný na L_2 ($L_1 \leq_{NC} L_2$)$\equiv_{df}$ $\exists$ NC-algoritmus, ktorý transformuje ľubovoľné slovo u_1 na slovo u_2 také, že

$$u_1 \in L_1 \Leftrightarrow u_2 \in L_2$$

- ak $L_1 \leq_{NC} L_2$, tak každý algoritmus rozhodujúci L_2 sa dá transformovať na algoritmus rozhodujúci L_1

NC-prevediteľnosť

L_1, L_2 jazyky

- L_1 je NC-prevediteľný na L_2 ($L_1 \leq_{NC} L_2$)$\equiv_{df}$ $\exists$ NC-algoritmus, ktorý transformuje ľubovoľné slovo u_1 na slovo u_2 také, že

$$u_1 \in L_1 \Leftrightarrow u_2 \in L_2$$

- ak $L_1 \leq_{NC} L_2$, tak každý algoritmus rozhodujúci L_2 sa dá transformovať na algoritmus rozhodujúci L_1

NC-prevediteľnosť

L_1, L_2 jazyky

- L_1 je NC-prevediteľný na L_2 ($L_1 \leq_{NC} L_2$)$\equiv_{df}$ $\exists$ NC-algoritmus, ktorý transformuje ľubovoľné slovo u_1 na slovo u_2 také, že

$$u_1 \in L_1 \Leftrightarrow u_2 \in L_2$$

- ak $L_1 \leq_{NC} L_2$, tak každý algoritmus rozhodujúci L_2 sa dá transformovať na algoritmus rozhodujúci L_1

Lemma

Nech L_1, L_2 sú jazyky a $L_1 \leq_{NC} L_2$.Potom

$$L_2 \in \mathrm{NC} \Rightarrow L_1 \in \mathrm{NC} \; .$$

log-space prevediteľnosť

- $L_1 \leq_{\log} L_2$ – L_1 je trasformované na L_2 v logaritmickom priestore – $\exists f$: $\forall u$ sa dá spočítať $f(u)$ v priestore $O(\log n)$ tak, že

$$u \in L_1 \Leftrightarrow f(u) \in L_2$$

log-space preveditеľnosť

- $L_1 \leq_{\log} L_2$ – L_1 je trasformované na L_2 v logaritmickom priestore – $\exists f$: $\forall u$ sa dá spočítať $f(u)$ v priestore $O(\log n)$ tak, že

$$u \in L_1 \Leftrightarrow f(u) \in L_2$$

- paralelná téza zaručuje:
  $\leq_{\log}$-prevediteľnosť $\Rightarrow$ $\leq_{NC}$-prevediteľnosť

log-space prevediteľnosť

- $L_1 \leq_{\log} L_2$ – L_1 je trasformované na L_2 v logaritmickom priestore – $\exists f$: $\forall u$ sa dá spočítať $f(u)$ v priestore $O(\log n)$ tak, že

$$u \in L_1 \Leftrightarrow f(u) \in L_2$$

- paralelná téza zaručuje: $\leq_{\log}$-prevediteľnosť $\Rightarrow$ $\leq_{NC}$-prevediteľnosť
- problém L je P-úplný ak

log-space prevediteľnosť

- $L_1 \leq_{\log} L_2$ – L_1 je trasformované na L_2 v logaritmickom priestore – $\exists f$: $\forall u$ sa dá spočítať $f(u)$ v priestore $O(\log n)$ tak, že

$$u \in L_1 \Leftrightarrow f(u) \in L_2$$

- paralelná téza zaručuje:
  $\leq_{\log}$-prevediteľnosť $\Rightarrow$ $\leq_{NC}$-prevediteľnosť
- problém L je P-úplný ak
  1. $L \in P$

log-space prevediteľnosť

- $L_1 \leq_{\log} L_2$ – L_1 je trasformované na L_2 v logaritmickom priestore – $\exists f$: $\forall u$ sa dá spočítať $f(u)$ v priestore $O(\log n)$ tak, že

$$u \in L_1 \Leftrightarrow f(u) \in L_2$$

- paralelná téza zaručuje:
  $\leq_{\log}$-prevediteľnosť $\Rightarrow$ $\leq_{NC}$-prevediteľnosť
- problém L je P-úplný ak
  1. $L \in P$
  2. Každý $A \in P$ sa dá previesť na L pomocou polynomiálneho počtu procesorov v polylogaritmickom čase

log-space prevediteľnosť

- $L_1 \leq_{\log} L_2$ – L_1 je trasformované na L_2 v logaritmickom priestore – $\exists f$: $\forall u$ sa dá spočítať $f(u)$ v priestore $O(\log n)$ tak, že

$$u \in L_1 \Leftrightarrow f(u) \in L_2$$

- paralelná téza zaručuje:
  $\leq_{\log}$-prevediteľnosť $\Rightarrow$ $\leq_{NC}$-prevediteľnosť
- problém L je P-úplný ak
  1. $L \in P$
  2. Každý $A \in P$ sa dá previesť na L pomocou polynomiálneho počtu procesorov v polylogaritmickom čase
- budeme robiť NC-prevediteľnosť, ale v skutočnosti to pôjde log-space prevediteľnosťou, t.j. vždy, keď ukážeme, že problém je P-úplný, tak to bude platiť aj vzhľadom k log-space prevediteľnosti.

Outline

Problém generovateľnosti – GEN

Zadanie: Bázová množina X a binárny operátor $\circ$, iniciálna množina $T \subset X$ a cieľový prvok $x \in X$.

Problém generovateľnosti – GEN

Zadanie: Bázová množina X a binárny operátor $\circ$, iniciálna množina $T \subset X$ a cieľový prvok $x \in X$.

Otázka: Patrí x do uzáveru T vzhľadom k operácii $\circ$?

Problém generovateľnosti – GEN

Zadanie: Bázová množina X a binárny operátor $\circ$, iniciálna množina $T \subset X$ a cieľový prvok $x \in X$.

Otázka: Patrí x do uzáveru T vzhľadom k operácii $\circ$?

- Výpočet uzáveru ($S \circ S = \{a \circ b \mid a \in S, b \in S\}$):

function closure(T)
begin $T' := T$
    while $T' \neq T' \cup T' \circ T'$ do $T' := T' \cup T' \circ T'$
    closure := T'
end;

Problém generovateľnosti – GEN

Zadanie: Bázová množina X a binárny operátor $\circ$, iniciálna množina $T \subset X$ a cieľový prvok $x \in X$.

Otázka: Patrí x do uzáveru T vzhľadom k operácii $\circ$?

- Výpočet uzáveru ($S \circ S = \{a \circ b \mid a \in S, b \in S\}$):

  function closure(T)
  begin $T' := T$
  while $T' \neq T' \cup T' \circ T'$ do $T' := T' \cup T' \circ T'$
  closure := T'
  end;

- while-cyklus maximálne $|X \setminus T|$-krát a $|closure(T)| \leq |X|$
  $\Rightarrow$ GEN $\in$ P

Problém generovateľnosti – GEN'

- ak by operácia $\circ$ bola asociatívna, tak by sme to vedeli vyriešiť v logaritmickom čase s polynomiálnym počtom procesorov

Problém generovateľnosti – GEN'

- ak by operácia $\circ$ bola asociatívna, tak by sme to vedeli vyriešiť v logaritmickom čase s polynomiálnym počtom procesorov
- P-úplnosť ukážeme najprv pre problém GEN':

Problém generovateľnosti – GEN'

- ak by operácia $\circ$ bola asociatívna, tak by sme to vedeli vyriešiť v logaritmickom čase s polynomiálnym počtom procesorov
- P-úplnosť ukážeme najprv pre problém GEN':

Zadanie: $(X, next)$, X množina, $next$ ternárny operátor na X, $T \subset X$ a $x \in X$.

Problém generovateľnosti – GEN'

- ak by operácia $\circ$ bola asociatívna, tak by sme to vedeli vyriešiť v logaritmickom čase s polynomiálnym počtom procesorov
- P-úplnosť ukážeme najprv pre problém GEN':

Zadanie: $(X, next)$, X množina, $next$ ternárny operátor na X, $T \subset X$ a $x \in X$.

Otázka: Patrí x do uzáveru T vzhľadom k operácii $next$?

Problém generovateľnosti – GEN’

- ak by operácia $\circ$ bola asociatívna, tak by sme to vedeli vyriešiť v logaritmickom čase s polynomiálnym počtom procesorov
- P-úplnosť ukážeme najprv pre problém GEN’:

Zadanie: $(X, next)$, X množina, $next$ ternárny operátor na X, $T \subset X$ a $x \in X$.

Otázka: Patrí x do uzáveru T vzhľadom k operácii $next$?

- zrejme GEN’ $\in$ P

Outline

1. P-úplnosť a ťažko paralelizovateľné úlohy
   - Paralelizovateľnosť
   - Problém generovateľnosti
   - P-úplnosť problému generovateľnosti
   - Problém CVP
   - CFE – problém prázdneho slova v bezkontextovom jazyku

Páskový model

- Nech Turingov stroj M rieši problém L (rozhoduje jazyk L) v čase $P(n)$, kde n je veľkosť vstupu. Počas výpočtu použije maximálne $P(n) + n$ políčok pásky. Nech $T(n) = P(n) + n$

Páskový model

- Nech Turingov stroj M rieši problém L (rozhoduje jazyk L) v čase $P(n)$, kde n je veľkosť vstupu. Počas výpočtu použije maximálne $P(n) + n$ políčok pásky. Nech $T(n) = P(n) + n$
- **Páskový model** TS

Páskový model

- Nech Turingov stroj M rieši problém L (rozhoduje jazyk L) v čase $P(n)$, kde n je veľkosť vstupu. Počas výpočtu použije maximálne $P(n) + n$ políčok pásky. Nech $T(n) = P(n) + n$
- **Páskový model** TS
  1. dĺžka pásky je $T(n) + 2$, políčka $0, 1, \ldots, T(n) + 1$

Páskový model

- Nech Turingov stroj M rieši problém L (rozhoduje jazyk L) v čase $P(n)$, kde n je veľkosť vstupu. Počas výpočtu použije maximálne $P(n) + n$ políčok pásky. Nech $T(n) = P(n) + n$
- **Páskový model** TS
  1. dĺžka pásky je $T(n) + 2$, políčka $0, 1, \ldots, T(n) + 1$
  2. políčko obsahuje jeden znak, 0-té a $(T(n) + 1)$-vé obsahujú zarážku $

Páskový model

- Nech Turingov stroj M rieši problém L (rozhoduje jazyk L) v čase $P(n)$, kde n je veľkosť vstupu. Počas výpočtu použije maximálne $P(n) + n$ políčok pásky. Nech $T(n) = P(n) + n$
- **Páskový model** TS
  1. dĺžka pásky je $T(n) + 2$, políčka $0, 1, \ldots, T(n) + 1$
  2. políčko obsahuje jeden znak, 0-té a $(T(n) + 1)$-vé obsahujú zarážku $
  3. pre $1 \leq p, t \leq T(n)$ je $c(p, t)$ obsah políčka p v čase t

Páskový model

- Nech Turingov stroj M rieši problém L (rozhoduje jazyk L) v čase $P(n)$, kde n je veľkosť vstupu. Počas výpočtu použije maximálne $P(n) + n$ políčok pásky. Nech $T(n) = P(n) + n$
- **Páskový model** TS
  1. dĺžka pásky je $T(n) + 2$, políčka $0, 1, \ldots, T(n) + 1$
  2. políčko obsahuje jeden znak, 0-té a $(T(n) + 1)$-vé obsahujú zarážku \$
  3. pre $1 \leq p, t \leq T(n)$ je $c(p, t)$ obsah políčka p v čase t
  4. $c(p, t + 1)$ je jednoznačne určené prechodovou funkciou TS

$c(p-1, t)$	$c(p, t)$	$c(p+1, t)$
	$c(p, t+1)$	

$$trans : X \times X \times X \to X$$

Páskový model

- Nech Turingov stroj M rieši problém L (rozhoduje jazyk L) v čase $P(n)$, kde n je veľkosť vstupu. Počas výpočtu použije maximálne $P(n) + n$ políčok pásky. Nech $T(n) = P(n) + n$
- **Páskový model** TS
  1. dĺžka pásky je $T(n) + 2$, políčka $0, 1, \ldots, T(n) + 1$
  2. políčko obsahuje jeden znak, 0-té a $(T(n) + 1)$-vé obsahujú zarážku $
  3. pre $1 \leq p, t \leq T(n)$ je $c(p, t)$ obsah políčka p v čase t
  4. $c(p, t + 1)$ je jednoznačne určené prechodovou funkciou TS

$c(p-1, t)$	$c(p, t)$	$c(p+1, t)$
	$c(p, t+1)$	

     $trans : X \times X \times X \to X$
  5. vstup je v $c(1, 0), \ldots, c(n, 0)$; stav je kódovaný spolu so znakom na 1 políčku

Páskový model

- Nech Turingov stroj M rieši problém L (rozhoduje jazyk L) v čase $P(n)$, kde n je veľkosť vstupu. Počas výpočtu použije maximálne $P(n) + n$ políčok pásky. Nech $T(n) = P(n) + n$
- **Páskový model** TS
  1. dĺžka pásky je $T(n) + 2$, políčka $0, 1, \ldots, T(n) + 1$
  2. políčko obsahuje jeden znak, 0-té a $(T(n) + 1)$-vé obsahujú zarážku \$
  3. pre $1 \leq p, t \leq T(n)$ je $c(p, t)$ obsah políčka p v čase t
  4. $c(p, t + 1)$ je jednoznačne určené prechodovou funkciou TS

$c(p-1,t)$	$c(p,t)$	$c(p+1,t)$
	$c(p,t+1)$	

     $trans : X \times X \times X \to X$
  5. vstup je v $c(1, 0), \ldots, c(n, 0)$; stav je kódovaný spolu so znakom na 1 políčku
  6. výstup je

$$c(1, T(n)) = \begin{cases} \# & \text{áno} \\ \text{iný znak} & \text{nie} \end{cases}$$

Dôkaz P-úplnosti GEN'

Veta

GEN'*je* P*-úplný problém.*

Dôkaz:

- nech $L \in \mathrm{P} \Rightarrow$ páskový model riešiaci L v čase $p(n)$ s abecedou Σ a množinou stavov Q

Dôkaz P-úplnosti GEN'

Veta

GEN'*je* P*-úplný problém.*

Dôkaz:

- nech $L \in \mathrm{P} \Rightarrow$ páskový model riešiaci L v čase $p(n)$ s abecedou Σ a množinou stavov Q
- GEN' $\in$ P už máme

Dôkaz P-úplnosti GEN'

Veta

GEN' *je* P*-úplný problém.*

Dôkaz:

- nech $L \in \mathrm{P} \Rightarrow$ páskový model riešiaci L v čase $p(n)$ s abecedou Σ a množinou stavov Q
- GEN' $\in \mathrm{P}$ už máme
- nech (t, p, sym) označuje, že v p-tom políčku v čase t je symbol sym

Dôkaz P-úplnosti GEN'

Veta

GEN' *je* P*-úplný problém.*

Dôkaz:

- nech $L \in \mathrm{P} \Rightarrow$ páskový model riešiaci L v čase $p(n)$ s abecedou Σ a množinou stavov Q
- GEN' $\in$ P už máme
- nech (t, p, sym) označuje, že v p-tom políčku v čase t je symbol sym
- bázová množina $X = \{\ (t, p, sym) \mid \ 0 \le t \le p(n),\ 0 \le p \le p(n) + n + 2,\ sym \in \Sigma \cup (\Sigma \times Q)\ \}$

Dôkaz P-úplnosti GEN'

Veta

GEN'*je* P*-úplný problém.*

Dôkaz:

- nech $L \in \mathrm{P} \Rightarrow$ páskový model riešiaci L v čase $p(n)$ s abecedou Σ a množinou stavov Q
- GEN' $\in$ P už máme
- nech (t, p, sym) označuje, že v p-tom políčku v čase t je symbol sym
- bázová množina $X = \{\ (t, p, sym) \mid \ 0 \leq t \leq p(n),\ 0 \leq p \leq p(n) + n + 2,\ sym \in \Sigma \cup (\Sigma \times Q)\ \}$
- iniciálna množina $T = \{(0, p, sym) \in X\}$ zodpovedajúca počiatočnému stavu pásky pre daný vstup

Dôkaz P-úplnosti GEN'

Veta

GEN'*je* P*-úplný problém.*

Dôkaz:

- nech $L \in \mathrm{P} \Rightarrow$ páskový model riešiaci L v čase $p(n)$ s abecedou Σ a množinou stavov Q
- GEN' $\in$ P už máme
- nech (t, p, sym) označuje, že v p-tom políčku v čase t je symbol sym
- bázová množina $X = \{\ (t, p, sym) \mid \ 0 \le t \le p(n),\ 0 \le p \le p(n) + n + 2,\ sym \in \Sigma \cup (\Sigma \times Q)\ \}$
- iniciálna množina $T = \{(0, p, sym) \in X\}$ zodpovedajúca počiatočnému stavu pásky pre daný vstup
- cieľový prvok $(T(n), 1, \#)$

Dôkaz P-úplnosti GEN'

Lemma

(t, p, sym) sa dá vygenerovať z T $\Leftrightarrow$ v čase t políčko p obsahuje symbol sym.

- naša transformácia je NC redukcia, pretože $next(u, v, w)$ sa dá spočítať paralelne v polylogaritmickom čase s polynomiálnym počtom procesorov

Veta

Nech A je P-úplný problém, $B \in \mathrm{P}$ a $A \leq_{\mathrm{NC}} B$. Potom B je P-úplný problém.

Dôkaz P-úplnosti GEN

Veta

GEN *je* P*-úplný problém.*

Dôkaz:

- GEN $\in$ P máme

Dôkaz P-úplnosti GEN

Veta

GEN *je* P*-úplný problém.*

Dôkaz:

- GEN $\in$ P máme
- GEN' je P-úplný; zadanie pre GEN': $(X', next), T', x'$ skonštruujeme zadanie pre GEN

Dôkaz P-úplnosti GEN

Veta

GEN *je* P*-úplný problém.*

Dôkaz:

- GEN $\in$ P máme
- GEN' je P-úplný; zadanie pre GEN': $(X', next), T', x'$ skonštruujeme zadanie pre GEN
  - $X := X' \cup (X')^2$

Dôkaz P-úplnosti GEN

Veta

GEN *je* P*-úplný problém.*

Dôkaz:

- GEN $\in$ P máme
- GEN' je P-úplný; zadanie pre GEN': $(X', next), T', x'$ skonštruujeme zadanie pre GEN
  - $X := X' \cup (X')^2$
  - binárny operátor $\circ$:

$$\forall u, v, w \in X : \begin{array}{rcl} u \circ v & := & (u, v) \\ (u, v) \circ w & := & next(u, v, w) \end{array}$$

Dôkaz P-úplnosti GEN

Veta

GEN *je* P*-úplný problém.*

Dôkaz:

- GEN $\in$ P máme
- GEN' je P-úplný; zadanie pre GEN': $(X', next), T', x'$ skonštruujeme zadanie pre GEN
  - $X := X' \cup (X')^2$
  - binárny operátor $\circ$:

$$\forall u, v, w \in X : \quad \begin{array}{rcl} u \circ v & := & (u, v) \\ (u, v) \circ w & := & next(u, v, w) \end{array}$$

  - $T := T'$

Dôkaz P-úplnosti GEN

Veta

GEN *je* P*-úplný problém.*

Dôkaz:

- GEN $\in$ P máme
- GEN' je P-úplný; zadanie pre GEN': $(X', next), T', x'$ skonštruujeme zadanie pre GEN
  - $X := X' \cup (X')^2$
  - binárny operátor $\circ$:
    $$\forall u, v, w \in X : \quad \begin{array}{rcl} u \circ v & := & (u, v) \\ (u, v) \circ w & := & next(u, v, w) \end{array}$$
  - $T := T'$
  - $x := x'$

Dôkaz P-úplnosti GEN

Veta

GEN *je* P*-úplný problém.*

Dôkaz:

- GEN $\in$ P máme
- GEN' je P-úplný; zadanie pre GEN': $(X', next), T', x'$ skonštruujeme zadanie pre GEN
  - $X := X' \cup (X')^2$
  - binárny operátor $\circ$:

    $$\forall u, v, w \in X : \quad \begin{array}{rcl} u \circ v & := & (u, v) \\ (u, v) \circ w & := & next(u, v, w) \end{array}$$
  - $T := T'$
  - $x := x'$
- x je generovateľné z T v $X \Leftrightarrow x'$ je generovateľné z T' v X'

Outline

1. P-úplnosť a ťažko paralelizovateľné úlohy
   - Paralelizovateľnosť
   - Problém generovateľnosti
   - P-úplnosť problému generovateľnosti
   - Problém CVP
   - CFE – problém prázdneho slova v bezkontextovom jazyku

Problém výstupnej hodnoty obvodu – CVP

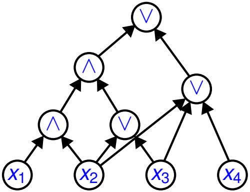
Circuit **V**alue **P**roblem

Zadanie: logický obvod (acyklický graf, vo vrcholoch hradlá) a vstupné hodnoty

Otázka: Je výstup logická 1?

B je báza, z ktorej berieme hradlá. Napr.

- $B =$ množina všetkých booleovských funkcií dvoch premenných
- $B = \{\wedge, \vee\}$ – monotónna báza (bez negácie) **M**onotone**CVP**
- $B = \{ \overbrace{\ldots}^{\wedge} \quad \overbrace{\ldots}^{\vee} \}$ – **U**nbounded**CVP**



P-úplnosť CVP

Veta

CVP *je* P*-úplný problém.*

Dôkaz:

- Nech $(X, T, \circ, s)$ je zadanie problému GEN. Logický obvod bude obsahovať X tak, že T budú vstupné uzly so vstupnou hodnotou 1 a ostatné budú mať vstup 0.

P-úplnosť CVP

Veta

CVP *je* P*-úplný problém.*

Dôkaz:

- Nech $(X, T, \circ, s)$ je zadanie problému GEN. Logický obvod bude obsahovať X tak, že T budú vstupné uzly so vstupnou hodnotou 1 a ostatné budú mať vstup 0.
- obvod bude mať $|X|$ vrstiev, prepojené budú iba po sebe idúce vrstvy

P-úplnosť CVP

Veta

CVP *je* P*-úplný problém.*

Dôkaz:

- Nech $(X, T, \circ, s)$ je zadanie problému GEN. Logický obvod bude obsahovať X tak, že T budú vstupné uzly so vstupnou hodnotou 1 a ostatné budú mať vstup 0.
- obvod bude mať $|X|$ vrstiev, prepojené budú iba po sebe idúce vrstvy
- pre všetky $x \in X \setminus T$ nájdeme všetky páry $(y_1, z_1), \ldots, (y_k, z_k)$ také, že $y_i \circ z_i = x$ a postavíme obvod realizujúci funkciu $(y_1 \wedge z_1) \vee (y_2 \wedge z_2) \vee \cdots \vee (y_k \wedge z_k)$ do uzlu x

P-úplnosť CVP

Veta

CVP *je* P*-úplný problém.*

Dôkaz:

- Nech $(X, T, \circ, s)$ je zadanie problému GEN. Logický obvod bude obsahovať X tak, že T budú vstupné uzly so vstupnou hodnotou 1 a ostatné budú mať vstup 0.
- obvod bude mať $|X|$ vrstiev, prepojené budú iba po sebe idúce vrstvy
- pre všetky $x \in X \setminus T$ nájdeme všetky páry $(y_1, z_1), \ldots, (y_k, z_k)$ také, že $y_i \circ z_i = x$ a postavíme obvod realizujúci funkciu $(y_1 \wedge z_1) \vee (y_2 \wedge z_2) \vee \cdots \vee (y_k \wedge z_k)$ do uzlu x
- do uzlu x sa dostane 1 $\Leftrightarrow$ existuje aspoň jedna dvojica ohodnotená 1

P-úplnosť CVP

Veta

CVP *je* P*-úplný problém.*

Dôkaz:

- Nech $(X, T, \circ, s)$ je zadanie problému GEN. Logický obvod bude obsahovať X tak, že T budú vstupné uzly so vstupnou hodnotou 1 a ostatné budú mať vstup 0.
- obvod bude mať $|X|$ vrstiev, prepojené budú iba po sebe idúce vrstvy
- pre všetky $x \in X \setminus T$ nájdeme všetky páry $(y_1, z_1), \ldots, (y_k, z_k)$ také, že $y_i \circ z_i = x$ a postavíme obvod realizujúci funkciu $(y_1 \wedge z_1) \vee (y_2 \wedge z_2) \vee \cdots \vee (y_k \wedge z_k)$ do uzlu x
- do uzlu x sa dostane 1 $\Leftrightarrow$ existuje aspoň jedna dvojica ohodnotená 1
- celkovo uzol s vo vrstve $|X|$ dostane 1 $\Leftrightarrow$ s je generovateľné z T.

P-úplnosť CVP

Veta

CVP *je* P*-úplný problém.*

Dôkaz:

- Nech $(X, T, \circ, s)$ je zadanie problému GEN. Logický obvod bude obsahovať X tak, že T budú vstupné uzly so vstupnou hodnotou 1 a ostatné budú mať vstup 0.
- obvod bude mať $|X|$ vrstiev, prepojené budú iba po sebe idúce vrstvy
- pre všetky $x \in X \setminus T$ nájdeme všetky páry $(y_1, z_1), \ldots, (y_k, z_k)$ také, že $y_i \circ z_i = x$ a postavíme obvod realizujúci funkciu $(y_1 \wedge z_1) \vee (y_2 \wedge z_2) \vee \cdots \vee (y_k \wedge z_k)$ do uzlu x
- do uzlu x sa dostane 1 $\Leftrightarrow$ existuje aspoň jedna dvojica ohodnotená 1
- celkovo uzol s vo vrstve $|X|$ dostane 1 $\Leftrightarrow$ s je generovateľné z T.
- konštrukcia je NC-redukcia

P-úplnosť CVP

Dôsledok

a) *MCVP je* P*-úplný problém*
b) *MCVP s topologicky usporiadanými uzlami je* P*-úplný problém (vstupné uzly majú menšie číslo, než uzly, do ktorých vedú vstupy – máme topologicky utriedený acyklický graf logického obvodu)*

Dôkaz:

a) zrejmé z predchádzajúcej konštrukcie

P-úplnosť CVP

Dôsledok

a) *MCVP je* P*-úplný problém*
b) *MCVP s topologicky usporiadanými uzlami je* P*-úplný problém (vstupné uzly majú menšie číslo, než uzly, do ktorých vedú vstupy – máme topologicky utriedený acyklický graf logického obvodu)*

Dôkaz:

a) zrejmé z predchádzajúcej konštrukcie
b) môžeme predpokladať, že prvky X sú očísľované a v dôkaze pre GEN môže očísľovanie zodpovedať lexikografickému usporiadaniu trojíc (t, p, sym)
$\Rightarrow$ z toho môžeme urobiť očísľovanie hradiel.

P-úplnosť CVP

Dôsledok

a) *MCVP je* P*-úplný problém*
b) *MCVP s topologicky usporiadanými uzlami je* P*-úplný problém (vstupné uzly majú menšie číslo, než uzly, do ktorých vedú vstupy – máme topologicky utriedený acyklický graf logického obvodu)*

Dôkaz:

a) zrejmé z predchádzajúcej konštrukcie
b) môžeme predpokladať, že prvky X sú očísľované a v dôkaze pre GEN môže očísľovanie zodpovedať lexikografickému usporiadaniu trojíc (t, p, sym)
$\Rightarrow$ z toho môžeme urobiť očísľovanie hradiel.

P-úplnosť CVP

Dôsledok

a) *MCVP je* P*-úplný problém*
b) *MCVP s topologicky usporiadanými uzlami je* P*-úplný problém (vstupné uzly majú menšie číslo, než uzly, do ktorých vedú vstupy – máme topologicky utriedený acyklický graf logického obvodu)*

Dôkaz:

a) zrejmé z predchádzajúcej konštrukcie
b) môžeme predpokladať, že prvky X sú očísľované a v dôkaze pre GEN môže očísľovanie zodpovedať lexikografickému usporiadaniu trojíc (t, p, sym)
$\Rightarrow$ z toho môžeme urobiť očísľovanie hradiel.

Pozn.: CVP je P-úplný aj pre planárne logické obvody, ale CVP nie je P-úplný pre planárne a zároveň monotónne obvody.

Outline

Prázdne slovo v bezkontextovom jazyku

Context-**F**ree Language **E**mpty Word Generation – CFE

Zadanie: bezkontextová gramatika G.

Veta

CFE *je* P*-úplný problém.*

Dôkaz:

Prázdne slovo v bezkontextovom jazyku

Context-**F**ree Language **E**mpty Word Generation – CFE

Zadanie: bezkontextová gramatika G.
Otázka: Obsahuje $L(G)$ prázdne slovo?

Veta

CFE *je* P*-úplný problém.*

Dôkaz:

Prázdne slovo v bezkontextovom jazyku

Context-**F**ree Language **E**mpty Word Generation – CFE

Zadanie: bezkontextová gramatika G.
Otázka: Obsahuje $L(G)$ prázdne slovo?

Veta

CFE *je* P*-úplný problém.*

Dôkaz:

- U CFE je veľkosť vstupu rovná veľkosti gramatiky. CFE $\in$ P.

Prázdne slovo v bezkontextovom jazyku

Context-**F**ree Language **E**mpty Word Generation – CFE

Zadanie: bezkontextová gramatika G.
Otázka: Obsahuje $L(G)$ prázdne slovo?

Veta

CFE *je* P*-úplný problém.*

Dôkaz:

- U CFE je veľkosť vstupu rovná veľkosti gramatiky. CFE $\in$ P.
- Nech $(X, T, \circ, x)$ je zadanie GEN.

Prázdne slovo v bezkontextovom jazyku

Context-**F**ree Language **E**mpty Word Generation – CFE

Zadanie: bezkontextová gramatika G.
Otázka: Obsahuje $L(G)$ prázdne slovo?

Veta

CFE *je* P*-úplný problém.*

Dôkaz:

- U CFE je veľkosť vstupu rovná veľkosti gramatiky. CFE $\in$ P.
- Nech $(X, T, \circ, x)$ je zadanie GEN.
- Skonštruujeme bezkontextovú gramatiku $G = (\Pi, \Sigma.S, P)$:

Prázdne slovo v bezkontextovom jazyku

Context-**F**ree Language **E**mpty Word Generation – CFE

Zadanie: bezkontextová gramatika G.
Otázka: Obsahuje $L(G)$ prázdne slovo?

Veta

CFE *je* P*-úplný problém.*

Dôkaz:

- U CFE je veľkosť vstupu rovná veľkosti gramatiky. CFE $\in$ P.
- Nech $(X, T, \circ, x)$ je zadanie GEN.
- Skonštruujeme bezkontextovú gramatiku $G = (\Pi, \Sigma.S, P)$:
  - $\Pi = X$,

Prázdne slovo v bezkontextovom jazyku

Context-**F**ree Language **E**mpty Word Generation – CFE

Zadanie: bezkontextová gramatika G.
Otázka: Obsahuje $L(G)$ prázdne slovo?

Veta

CFE *je* P*-úplný problém.*

Dôkaz:

- U CFE je veľkosť vstupu rovná veľkosti gramatiky. CFE $\in$ P.
- Nech $(X, T, \circ, x)$ je zadanie GEN.
- Skonštruujeme bezkontextovú gramatiku $G = (\Pi, \Sigma.S, P)$:
  - $\Pi = X$,
  - $\Sigma = \emptyset$,

Prázdne slovo v bezkontextovom jazyku

Context-**F**ree Language **E**mpty Word Generation – CFE

Zadanie: bezkontextová gramatika G.
Otázka: Obsahuje $L(G)$ prázdne slovo?

Veta

CFE *je* P*-úplný problém.*

Dôkaz:

- U CFE je veľkosť vstupu rovná veľkosti gramatiky. CFE $\in$ P.
- Nech $(X, T, \circ, x)$ je zadanie GEN.
- Skonštruujeme bezkontextovú gramatiku $G = (\Pi, \Sigma.S, P)$:
  - $\Pi = X$,
  - $\Sigma = \emptyset$,
  - $S = x$,

Prázdne slovo v bezkontextovom jazyku

Context-**F**ree Language **E**mpty Word Generation – CFE

Zadanie: bezkontextová gramatika G.
Otázka: Obsahuje $L(G)$ prázdne slovo?

Veta

CFE *je* P*-úplný problém.*

Dôkaz:

- U CFE je veľkosť vstupu rovná veľkosti gramatiky. CFE $\in$ P.
- Nech $(X, T, \circ, x)$ je zadanie GEN.
- Skonštruujeme bezkontextovú gramatiku $G = (\Pi, \Sigma.S, P)$:
  - $\Pi = X$,
  - $\Sigma = \emptyset$,
  - $S = x$,
  - $$\begin{aligned} x \to yz \quad &\equiv_{df} \quad y \circ z = x \\ x \to \lambda \quad &\equiv_{df} \quad x \in T \end{aligned}$$

Prázdne slovo v bezkontextovom jazyku

Context-**F**ree Language **E**mpty Word Generation – CFE

Zadanie: bezkontextová gramatika G.
Otázka: Obsahuje $L(G)$ prázdne slovo?

Veta

CFE *je* P*-úplný problém.*

Dôkaz:

- U CFE je veľkosť vstupu rovná veľkosti gramatiky. CFE $\in$ P.
- Nech $(X, T, \circ, x)$ je zadanie GEN.
- Skonštruujeme bezkontextovú gramatiku $G = (\Pi, \Sigma.S, P)$:
  - $\Pi = X$,
  - $\Sigma = \emptyset$,
  - $S = x$,
  - $$\begin{array}{lll} x \to yz & \equiv_{df} & y \circ z = x \\ x \to \lambda & \equiv_{df} & x \in T \end{array}$$
- G generuje prázdne slovo $\Leftrightarrow (X, T, \circ, x)$ má riešenie.

Prázdne slovo v bezkontextovom jazyku

- Ak gramatika G neobsahuje pravidlá s prázdnou pravou stranou a za veľkosť vstupu berieme súčet dĺžky vstupného slova w a veľkosti gramatiky, tak je problém $w \stackrel{?}{\in} L(G)$ v NC.

Prázdne slovo v bezkontextovom jazyku

- Ak gramatika G neobsahuje pravidlá s prázdnou pravou stranou a za veľkosť vstupu berieme súčet dĺžky vstupného slova w a veľkosti gramatiky, tak je problém $w \stackrel{?}{\in} L(G)$ v NC.
- Ak sa dovolia λ-pravidlá, tak je to P-úplné.

Prázdne slovo v bezkontextovom jazyku

- Ak gramatika G neobsahuje pravidlá s prázdnou pravou stranou a za veľkosť vstupu berieme súčet dĺžky vstupného slova w a veľkosti gramatiky, tak je problém $w \stackrel{?}{\in} L(G)$ v NC.
- Ak sa dovolia λ-pravidlá, tak je to P-úplné.
- problém $L(G) \stackrel{?}{=} \emptyset$:

Prázdne slovo v bezkontextovom jazyku

- Ak gramatika G neobsahuje pravidlá s prázdnou pravou stranou a za veľkosť vstupu berieme súčet dĺžky vstupného slova w a veľkosti gramatiky, tak je problém $w \overset{?}{\in} L(G)$ v NC.
- Ak sa dovolia λ-pravidlá, tak je to P-úplné.
- problém $L(G) \overset{?}{=} \emptyset$:
  - pre každé $x \in T$ pridáme pravidlo $x \to a$, kde a je nejaký terminál

Prázdne slovo v bezkontextovom jazyku

- Ak gramatika G neobsahuje pravidlá s prázdnou pravou stranou a za veľkosť vstupu berieme súčet dĺžky vstupného slova w a veľkosti gramatiky, tak je problém $w \overset{?}{\in} L(G)$ v NC.
- Ak sa dovolia λ-pravidlá, tak je to P-úplné.
- problém $L(G) \overset{?}{=} \emptyset$:
  - pre každé $x \in T$ pridáme pravidlo $x \to a$, kde a je nejaký terminál
- POZOR: Rozpoznávanie pevne daného bezkontextového jazyka je v NC!