

# Algorytmy Równoległe i Rozproszone

## Część X - Algorytmy samostabilizujące.

Łukasz Kuszner  
pokój 209, WETI

<http://www.sphere.pl/~kuszner/>  
kuszner@sphere.pl

Oficjalna strona wykładu  
<http://www.sphere.pl/~kuszner/ARiR/>  
Wykład 15 godzin, Projekt 15 godzin

2006/07

Definicje

Uwagi do założeń

Rodziny układów

Redukcje

Problemy P-trudne

Strona główna

Strona tytułowa



Strona 1 z 18

Powrót

Full Screen

Zamknij

Koniec

# 1. Definicje

Boolowskie układy kombinacyjne (o takich będziemy mówić) są formalnym modelem dla układów logicznych. Układ reprezentujemy za pomocą digrafu, w którym krawędzie przenoszą jednokierunkowy sygnał logiczny a wierzchołki obliczają elementarne funkcje logiczne.

# 1. Definicje

Boolowskie układy kombinacyjne (o takich będziemy mówić) są formalnym modelem dla układów logicznych. Układ reprezentujemy za pomocą digrafu, w którym krawędzie przenoszą jednokierunkowy sygnał logiczny a wierzchołki obliczają elementarne funkcje logiczne.

Niech  $B_k = \{f \mid f : \{0, 1\}^k \rightarrow \{0, 1\}\}$  będzie zbiorem  $k$  argumentowych funkcji logicznych.

Definicje

Uwagi do założeń

Rodziny układów

Redukcje

Problemy P-trudne

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 3 z 18

Powrót

Full Screen

Zamknij

Koniec

Niech  $B_k = \{f \mid f : \{0, 1\}^k \rightarrow \{0, 1\}\}$  będzie zbiorem  $k$  argumentowych funkcji logicznych.

Przez *Boolowski układ kombinacyjny*, lub w skrócie *układ kombinacyjny*, rozumiemy etykietowany, skończony, acykliczny graf skierowany, w którym każdy z wierzchołków jest:

- *wejściem*, jeśli jego stopień wejściowy jest 0,

Niech  $B_k = \{f \mid f : \{0, 1\}^k \rightarrow \{0, 1\}\}$  będzie zbiorem  $k$  argumentowych funkcji logicznych.

Przez *Boolowski układ kombinacyjny*, lub w skrócie *układ kombinacyjny*, rozumiemy etykietowany, skończony, acykliczny graf skierowany, w którym każdy z wierzchołków jest:

- *wejściem*, jeśli jego stopień wejściowy jest 0,
- *wyjściem*, jeśli jego stopień wyjściowy jest 0,

Niech  $B_k = \{f \mid f : \{0, 1\}^k \rightarrow \{0, 1\}\}$  będzie zbiorem  $k$  argumentowych funkcji logicznych.

Przez *Boolowski układ kombinacyjny*, lub w skrócie *układ kombinacyjny*, rozumiemy etykietowany, skończony, acykliczny graf skierowany, w którym każdy z wierzchołków jest:

- *wejściem*, jeśli jego stopień wejściowy jest 0,
- *wyjściem*, jeśli jego stopień wyjściowy jest 0,
- *bramką*, jeśli jest funkcją typu  $B_1$  lub  $B_2$ .

Wierzchołek typu  $B_i$  musi mieć stopień wejściowy (indeg) równy  $i$ , natomiast stopień wyjściowy (outdeg) nie jest ograniczony.

Definicje

Uwagi do założeń

Rodziny układów

Redukcje

Problemy P-trudne

Zauważmy, że:

- wejścia i bramki mogą być jednocześnie wyjściami,

Strona główna

Strona tytułowa



Strona 4 z 18

Powrót

Full Screen

Zamknij

Koniec

Zauważmy, że:

- wejścia i bramki mogą być jednocześnie wyjściami,
- układ kombinacyjny oblicza funkcję  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ , gdzie  $n$  jest liczbą wejść, a  $m$  liczbą wyjść w układzie.

Interesującymi nas wielkościami będą rozmiar i głębokość układu. *Rozmiarem* układu  $\alpha$ , nazywamy liczbę jego wierzchołków  $size(\alpha)$ .

Zauważmy, że:

- wejścia i bramki mogą być jednocześnie wyjściami,
- układ kombinacyjny oblicza funkcję  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ , gdzie  $n$  jest liczbą wejść, a  $m$  liczbą wyjść w układzie.

Interesującymi nas wielkościami będą rozmiar i głębokość układu. *Rozmiarem* układu  $\alpha$ , nazywamy liczbę jego wierzchołków  $size(\alpha)$ .

*Głębokością* układu  $\alpha$ , nazywamy długość (liczbę krawędzi) najdłuższej ścieżki od wejścia do wyjścia w  $\alpha$  i oznaczamy  $depth(\alpha)$ .

[Strona główna](#)[Strona tytułowa](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)[Strona 4 z 18](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

## 2. Uwagi do przyjętych założeń

Dopuszczenie bramek o dowolnie dużej liczbie wyjść może się wydawać nierealistyczne, jednak pokazano, że ograniczenie liczby wyjść powoduje jedynie albo liniowy wzrost rozmiaru albo głębokości układu.

Definicje

Uwagi do założeń

Rodziny układów

Redukcje

Problemy P-trudne

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 5 z 18

Powrót

Full Screen

Zamknij

Koniec

## 2. Uwagi do przyjętych założeń

Dopuszczenie bramek o dowolnie dużej liczbie wyjść może się wydawać nierealistyczne, jednak pokazano, że ograniczenie liczby wyjść powoduje jedynie albo liniowy wzrost rozmiaru albo głębokości układu.

Dopuszczenie bramek o nieograniczonym rozmiarze wejścia może zmniejszyć głębokość układu o czynnik  $\log \log size$  ale za cenę wielomianowego zwiększenia rozmiaru (wtedy za rozmiar przyjmuje się liczbę krawędzi w układzie) lub pozwala skonstruować układ o głębokości 2 (korzystając z potraci kanonicznych) jednak za cenę wykładniczego wzrostu rozmiaru.

Definicje

Uwagi do założeń

Rodziny układów

Redukcje

Problemy P-trudne

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 5 z 18

Powrót

Full Screen

Zamknij

Koniec

## 2. Uwagi do przyjętych założeń

Dopuszczenie bramek o dowolnie dużej liczbie wyjść może się wydawać nierealistyczne, jednak pokazano, że ograniczenie liczby wyjść powoduje jedynie albo liniowy wzrost rozmiaru albo głębokości układu.

Dopuszczenie bramek o nieograniczonym rozmiarze wejścia może zmniejszyć głębokość układu o czynnik  $\log \log size$  ale za cenę wielomianowego zwiększenia rozmiaru (wtedy za rozmiar przyjmuje się liczbę krawędzi w układzie) lub pozwala skonstruować układ o głębokości 2 (korzystając z potraci kanonicznych) jednak za cenę wykładniczego wzrostu rozmiaru.

Z drugiej strony konwersja układu z bramkami o nieograniczonej liczbie wejść do układu w przedstawionym modelu może spowodować kwadratowy wzrost rozmiaru i wzrost głębokości o czynnik proporcjonalny do  $\log size$ .

Definicje

Uwagi do założeń

Rodziny układów

Redukcje

Problemy P-trudne

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 5 z 18

Powrót

Full Screen

Zamknij

Koniec

# 3. Rodziny układów kombinacyjnych

Pojedynczy układ oblicza nam funkcje logiczne o zadanym rozmiarze wejścia. Nie jest to sytuacja do jakiej jesteśmy przyzwyczajeni myśląc o algorytmach i obliczeniach. Jeden algorytm powinien rozwiązywać problem bez względu na rozmiar wejścia. Również rozmiar danych wyjściowych nie jest stały. Tę sytuację modelujemy funkcją

$$f_\alpha : \{0, 1\}^* \rightarrow \{0, 1\}^*$$

i nieskończonym zbiorem układów  $\{\alpha_n\}$ , w którym układ  $\alpha_n$  oblicza funkcję

$$f_\alpha : \{0, 1\}^n \rightarrow \{0, 1\}^{m(n)},$$

gdzie  $m(n)$  jest największym rozmiarem wyjścia dla danych o rozmiarze  $n$ .

Definicje

Uwagi do założeń

Rodziny układów

Redukcje

Problemy P-trudne

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 6 z 18

Powrót

Full Screen

Zamknij

Koniec

### 3.1. Opis układów kombinacyjnych

Jeśli nie ograniczymy sposobów definiowania rodzin układów możemy definiować bardzo dziwne układy, które np. rozwiązują problemy nierozwiązywalne. (zob. LtPC str. 31).

Dlatego ograniczamy możliwość definiowania układów do takich, które dają się szybko obliczyć.

Definicje

Uwagi do założeń

Rodziny układów

Redukcje

Problemy P-trudne

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 8 z 18

Powrót

Full Screen

Zamknij

Koniec

*Kodem standardowym  $\overline{\alpha_n}$  dla układu  $\alpha$  nazywamy unikalny ciąg ze zbioru  $\{0, 1\}^*$  definiowany w sposób jednoznaczny dla układu  $\alpha_n$  (zob. LtPC str. 29).*

*Kodem standardowym  $\overline{\alpha_n}$  dla układu  $\alpha$  nazywamy unikalny ciąg ze zbioru  $\{0, 1\}^*$  definiowany w sposób jednoznaczny dla układu  $\alpha_n$  (zob. LtPC str. 29).*

**Definicja 1** *Rodzinę układów kombinacyjnych  $\{\alpha_n\}$  nazywamy jednorodną, jeśli przekształcenie  $1^n \rightarrow \overline{\alpha_n}$  może być obliczone na deterministycznej maszynie Turinga o złożoności pamięciowej  $O(\log \text{size}(\alpha_n))$ .*

Przypadek, w którym funkcja  $m(n) = 1$  jest użyteczny dla definiowania języków formalnych.

**Definicja 2** Niech  $\{\alpha_n\}$  oblicza funkcję

$$f_\alpha : \{0, 1\}^* \rightarrow \{0, 1\}$$

Językiem rozpoznawanym przez rodzinę układów  $\alpha_n$  nazywamy zbiór

$$L_\alpha = \{x \in \{0, 1\}^* \mid f_\alpha(x) = 1\}.$$

**Definicja 3** Dla każdego  $k \geq 1$  klasą  $\mathbf{NC}^k$  nazywamy zbiór wszystkich języków rozpoznawanych przez jednorodną rodzinę  $\{\alpha_n\}$  o rozmiarze  $size(\alpha_n) = n^{O(1)}$  i głębokości  $size(\alpha_n) = O(\log^k n)$

Definicje

Uwagi do założeń

Rodziny układów

Redukcje

Problemy P-trudne

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 10 z 18

Powrót

Full Screen

Zamknij

Koniec

**Definicja 3** Dla każdego  $k \geq 1$  klasą  $\mathbf{NC}^k$  nazywamy zbiór wszystkich języków rozpoznawanych przez jednorodną rodzinę  $\{\alpha_n\}$  o rozmiarze  $size(\alpha_n) = n^{O(1)}$  i głębokości  $size(\alpha_n) = O(\log^k n)$

**Definicja 4** Klasą  $\mathbf{NC}$  nazywamy zbiór

$$\mathbf{NC} = \bigcup_{k \geq 1} \mathbf{NC}^k$$

Podobnie jak klasy języków definiujemy  $\mathbf{FNC}^k$  jako klasy funkcji z  $\{0, 1\}^*$  w  $\{0, 1\}^*$  obliczalnych przez układy o głębokości  $O(\log^k n)$

Celem powyższych definicji jest możliwość rozróżnienia problemów dla których możemy znaleźć „dobre, równoległe” rozwiązanie i takich, dla których takiego rozwiązania prawdopodobnie nie ma.

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 10 z 18

Powrót

Full Screen

Zamknij

Koniec

**Definicja 5** Klasą **P** nazywamy zbiór wszystkich języków rozpoznawanych przez jednorodną rodzinę  $\{\alpha_n\}$  o rozmiarze  $size(\alpha_n) = n^{O(1)}$  i głębokości  $depth(\alpha_n) = n^{O(1)}$ .

Podobnie definiujemy **FP** jako klasę funkcji z  $\{0, 1\}^*$  w  $\{0, 1\}^*$  obliczalnych przez układy o głębokości  $n^{O(1)}$ .

## 4. Redukcje

### 4.1. Redukcje typu wiele-do-jeden

**Definicja 6** Niech  $L, L'$  będą językami, piszemy  $L \leq_m L'$ , jeśli istnieje funkcja  $f$  taka, że  $x \in L$  wtedy i tylko wtedy, gdy  $f(x) \in L'$ . Podobnie:

- $L \leq_m^P L'$ , jeśli  $f \in \mathbf{FP}$
- $L \leq_m^{\mathbf{NC}^k} L'$ , jeśli  $f \in \mathbf{FNC}^k$
- $L \leq_m^{\mathbf{NC}} L'$ , jeśli  $f \in \mathbf{FNC}$

**Twierdzenie 1** Relacje  $\leq_m, \leq_m^P, \leq_m^{\mathbf{NC}^k}, L \leq_m^{\mathbf{NC}}$ , są przechodnie.

### Ćwiczenie 1

Udowodnij twierdzenie 1.

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)

Strona 12 z 18

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

## 4.2. Redukcje z „wyrocznią”

**Definicja 7** Niech  $B$  będzie problemem optymalizacyjnym. Rodzinę układów kombinacyjnych  $\alpha_n$  nazywamy rodziną układów z  $B$  wyrocznią, jeśli  $\alpha_n$  jest układem wyposażonym dodatkowo w bramki typu wyrocznia dla  $B$ , które potrafią udzielać odpowiedzi na instancję problemu  $B$ . Jeśli bramka wyrocznia ma  $k$  wejść i  $l$  wyjść, to jej rozmiar liczymy jako  $k + l$ , a głębokość jako  $\lg(k + l)$ .

**Definicja 8** Niech  $B, B'$  będą problemami optymalizacyjnymi. Zachodzi  $B \leq_T^{\text{NC}^k} B'$ , wtedy i tylko wtedy, gdy istnieje jednorodna rodzina  $B'$  wyroczeni  $\{\alpha_n\}$ , która rozwiązuje  $B$  tak, że  $\text{size}(\alpha_n) = n^{O(1)}$  oraz  $\text{depth}(\alpha_n) = \log^k n$ .

**Definicja 9** Niech  $B, B'$  będą problemami optymalizacyjnymi. Zachodzi  $B \leq_T^{\text{NC}} B'$ , wtedy i tylko wtedy, gdy istnieje  $k$ , takie że  $B \leq_T^{\text{NC}^k} B'$ .

## 5. Problemy P-trudne - czy $NC \neq P$ ?

**Definicja 10** *Problem optymalizacyjny  $B$  jest  $\mathbf{P}$ -trudny, jeśli  $L \leq_T^{NC} B$  dla wszystkich języków  $L \in \mathbf{P}$ . Problem decyzyjny  $B$  jest  $\mathbf{FP}$ -zupełny, jeśli  $B \in \mathbf{FP}$  i  $B$  jest  $\mathbf{P}$ -trudny.*

W sposób analogiczny definiujemy języki  $\mathbf{P}$ -trudne i  $\mathbf{P}$ -zupełne

### Ćwiczenie 2

Podaj odpowiednie definicje dla języków  $\mathbf{P}$ -trudnych i  $\mathbf{P}$ -zupełnych.

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)

Strona 15 z 18

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

**Definicja 11** *Problem GMSP (ang. generic machine simulation problem). Mając dane: opis maszyny Turinga  $M$ , oraz ciąg wejściowy  $t$  kodowany unarnie; stwierdzić, czy  $M$  akceptuje  $t$ .*

**Twierdzenie 2** *Problem symulacji maszyny Turinga jest P-zupełny.*

Dowód opiera się na dwóch faktach:

- Symulacja maszyny Turinga jest problemem wielomianowym
- Jeśli umiemy symulować działanie dowolnej maszyny Turinga, to tym bardziej jednej szczególnej.

Nie jest to oczywiście żadna rewelacja, podobnym problemem zupełnym w teorii **NP** zupełności byłoby symulowanie niedeterministycznej maszyny RAM na maszynie deterministycznej.

Udało się jednak udowodnić **P**-zupełność innego problemu, co można porównać z dowodem **NP**-zupełności problemu *3SAT*.

**Definicja 12** *Problem CVP (ang. Circuit Value Problem)*  
Mając dane: kod  $\bar{\alpha}$  układu  $\alpha$ , wejścia  $x_1, x_2, \dots, x_n$  i wyróżnioną bramkę  $y$ ; wyznaczyć wyjście wyróżnionej bramki.

**Twierdzenie 3** *Problem CVP jest P-zupełny.*

*Definicje*

*Uwagi do założeń*

*Rodziny układów*

*Redukcje*

*Problemy P-trudne*

*Strona główna*

*Strona tytułowa*



*Strona 18 z 18*

*Powrót*

*Full Screen*

*Zamknij*

*Koniec*