

Die Komplexität von Problemen auf implizit repräsentierten Eingaben

Daniel Sawitzki

Lehrstuhl-2-Seminar

7. Februar 2006

Inhalt

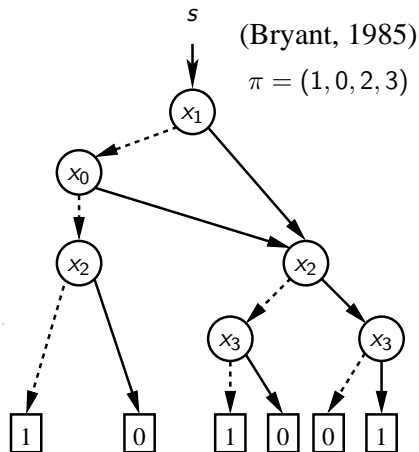
- 1** OBDDs und Motivation
- 2** Anzahl der OBDD-Operationen
- 3** Nichtexistenz von Festparameteralgorithmen
- 4** Konkrete exponentielle untere Schranken
- 5** Fazit und Ausblick

Inhalt

- 1** OBDDs und Motivation
- 2 Anzahl der OBDD-Operationen
- 3 Nichtexistenz von Festparameteralgorithmen
- 4 Konkrete exponentielle untere Schranken
- 5 Fazit und Ausblick

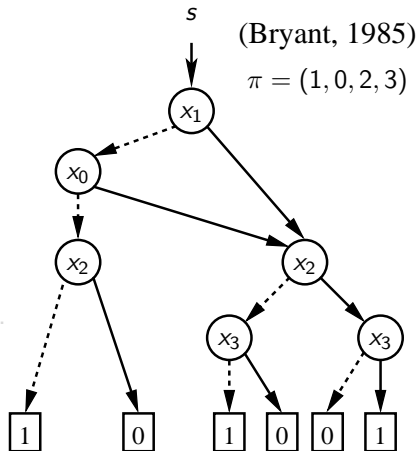
Ordered Binary Decision Diagrams (OBDDs)

- Datenstruktur für $f: \{0, 1\}^n \rightarrow \{0, 1\}$ mit Var. $x_0, \dots, x_{n-1} \in \{0, 1\}$
- OBDD \mathcal{G} ist azykl. Digraph mit **inneren Knoten** und **Senken**.
- Innere Knoten: Variablenmarkierung, 0- und 1-Kante
- Senke entspricht Wert $f(x_0, \dots, x_{n-1})$.
- Zeiger auf **Quelle** s
- Liest Var. bzgl. $\pi \in \Sigma_n$.



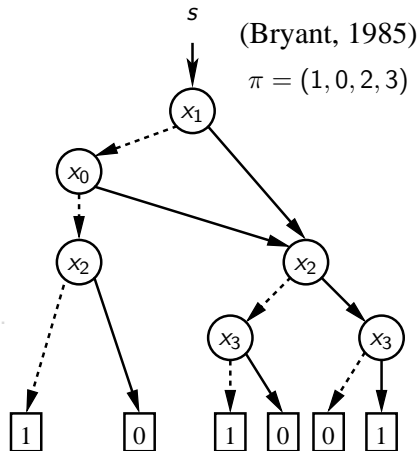
Ordered Binary Decision Diagrams (OBDDs)

- Datenstruktur für $f: \{0, 1\}^n \rightarrow \{0, 1\}$ mit Var. $x_0, \dots, x_{n-1} \in \{0, 1\}$
- OBDD \mathcal{G} ist azykl. Digraph mit **inneren Knoten** and **Senken**.
- Innere Knoten: Variablenmarkierung, 0- und 1-Kante
- Senke entspricht Wert $f(x_0, \dots, x_{n-1})$.
- Zeiger auf **Quelle** s
- Liest Var. bzgl. $\pi \in \Sigma_n$.



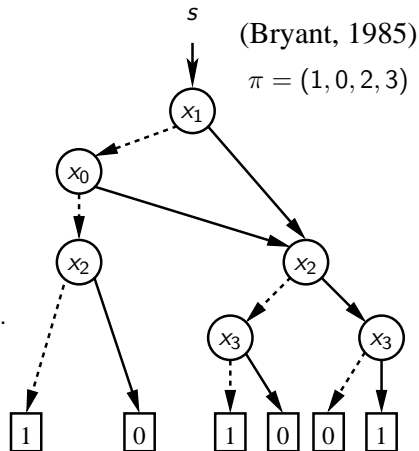
Ordered Binary Decision Diagrams (OBDDs)

- Datenstruktur für $f: \{0, 1\}^n \rightarrow \{0, 1\}$ mit Var. $x_0, \dots, x_{n-1} \in \{0, 1\}$
- OBDD \mathcal{G} ist azykl. Digraph mit **inneren Knoten** and **Senken**.
- Innere Knoten: Variablenmarkierung, 0- und 1-Kante
- Senke entspricht Wert $f(x_0, \dots, x_{n-1})$.
- Zeiger auf **Quelle** s
- Liest Var. bzgl. $\pi \in \Sigma_n$.



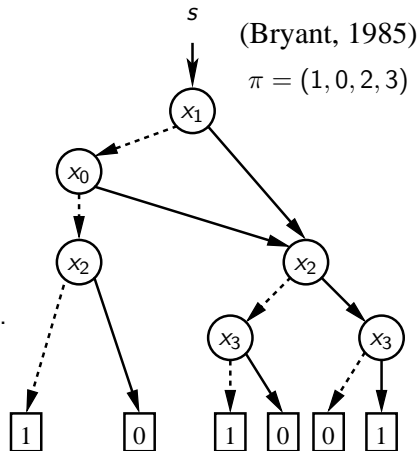
Ordered Binary Decision Diagrams (OBDDs)

- Datenstruktur für $f: \{0, 1\}^n \rightarrow \{0, 1\}$ mit Var. $x_0, \dots, x_{n-1} \in \{0, 1\}$
- OBDD \mathcal{G} ist azykl. Digraph mit **inneren Knoten** and **Senken**.
- Innere Knoten: Variablenmarkierung, 0- und 1-Kante
- Senke entspricht Wert $f(x_0, \dots, x_{n-1})$.
- Zeiger auf **Quelle** s
- Liest Var. bzgl. $\pi \in \Sigma_n$.



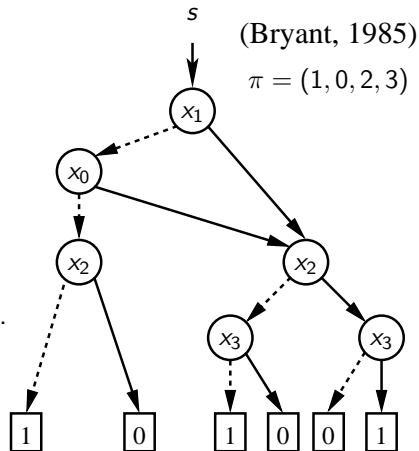
Ordered Binary Decision Diagrams (OBDDs)

- Datenstruktur für $f: \{0, 1\}^n \rightarrow \{0, 1\}$ mit Var. $x_0, \dots, x_{n-1} \in \{0, 1\}$
- OBDD \mathcal{G} ist azykl. Digraph mit **inneren Knoten** and **Senken**.
- Innere Knoten: Variablenmarkierung, 0- und 1-Kante
- Senke entspricht Wert $f(x_0, \dots, x_{n-1})$.
- Zeiger auf **Quelle** s
- Liest Var. bzgl. $\pi \in \Sigma_n$.



Ordered Binary Decision Diagrams (OBDDs)

- Datenstruktur für $f: \{0, 1\}^n \rightarrow \{0, 1\}$ mit Var. $x_0, \dots, x_{n-1} \in \{0, 1\}$
- OBDD \mathcal{G} ist azykl. Digraph mit **inneren Knoten** and **Senken**.
- Innere Knoten: Variablenmarkierung, 0- und 1-Kante
- Senke entspricht Wert $f(x_0, \dots, x_{n-1})$.
- Zeiger auf **Quelle** s
- Liest Var. bzgl. $\pi \in \Sigma_n$.



OBDDs – Eigenschaften und Operationen

- Jede Funktion f auf n Var. hat ein OBDD der Größe $(2 + o(1))2^n/n$.
- Hoffnung bei strukturierten Funktionen: OBDD-Größe $\text{poly}(n)$
- Effiziente Operationen für OBDDs \mathcal{G}_f und \mathcal{G}_h :
 - Erfüllbarkeit: $f \neq 0$ in Zeit $O(1)$
 - Äquivalenztest: $f = h$ in Zeit $O(|\mathcal{G}_f| + |\mathcal{G}_h|)$
 - Variablenersetzung: $f_{|x_i=0/1}$ in Zeit $O(|\mathcal{G}_f|)$
 - Binäre Synthese: $f \otimes h$ in Zeit $O(|\mathcal{G}_f| \cdot |\mathcal{G}_h|)$
 - Quantifizierung: $(\exists/\forall x_i)f$ in Zeit $O(|\mathcal{G}_f|^2)$

OBDDs – Eigenschaften und Operationen

- Jede Funktion f auf n Var. hat ein OBDD der Größe $(2 + o(1))2^n/n$.
- Hoffnung bei strukturierten Funktionen: OBDD-Größe $\text{poly}(n)$
- Effiziente Operationen für OBDDs \mathcal{G}_f und \mathcal{G}_h :
 - Erfüllbarkeit: $f \neq 0$ in Zeit $O(1)$
 - Äquivalenztest: $f = h$ in Zeit $O(|\mathcal{G}_f| + |\mathcal{G}_h|)$
 - Variablenersetzung: $f_{|x_i=0/1}$ Zeit $O(|\mathcal{G}_f|)$
 - Binäre Synthese: $f \otimes h$ in Zeit $O(|\mathcal{G}_f| \cdot |\mathcal{G}_h|)$
 - Quantifizierung: $(\exists/\forall x_i)f$ in Zeit $O(|\mathcal{G}_f|^2)$

OBDDs – Eigenschaften und Operationen

- Jede Funktion f auf n Var. hat ein OBDD der Größe $(2 + o(1))2^n/n$.
- Hoffnung bei strukturierten Funktionen: OBDD-Größe $\text{poly}(n)$
- Effiziente Operationen für OBDDs \mathcal{G}_f und \mathcal{G}_h :
 - **Erfüllbarkeit:** $f \neq 0$ in Zeit $\mathcal{O}(1)$
 - **Äquivalenztest:** $f = h$ in Zeit $\mathcal{O}(|\mathcal{G}_f| + |\mathcal{G}_h|)$
 - **Variablenersetzung:** $f_{|x_i=0/1}$ in Zeit $\mathcal{O}(|\mathcal{G}_f|)$
 - **Binäre Synthese:** $f \otimes h$ in Zeit $\mathcal{O}(|\mathcal{G}_f| \cdot |\mathcal{G}_h|)$
 - **Quantifizierung:** $(\exists/\forall x_i)f$ in Zeit $\mathcal{O}(|\mathcal{G}_f|^2)$

OBDDs – Eigenschaften und Operationen

- Jede Funktion f auf n Var. hat ein OBDD der Größe $(2 + o(1))2^n/n$.
- Hoffnung bei strukturierten Funktionen: OBDD-Größe $\text{poly}(n)$
- Effiziente Operationen für OBDDs \mathcal{G}_f und \mathcal{G}_h :
 - **Erfüllbarkeit:** $f \neq 0$ in Zeit $\mathcal{O}(1)$
 - **Äquivalenztest:** $f = h$ in Zeit $\mathcal{O}(|\mathcal{G}_f| + |\mathcal{G}_h|)$
 - **Variablenersetzung:** $f_{|x_i=0/1}$ Zeit $\mathcal{O}(|\mathcal{G}_f|)$
 - **Binäre Synthese:** $f \otimes h$ in Zeit $\mathcal{O}(|\mathcal{G}_f| \cdot |\mathcal{G}_h|)$
 - **Quantifizierung:** $(\exists/\forall x_i)f$ in Zeit $\mathcal{O}(|\mathcal{G}_f|^2)$

OBDDs – Eigenschaften und Operationen

- Jede Funktion f auf n Var. hat ein OBDD der Größe $(2 + o(1))2^n/n$.
- Hoffnung bei strukturierten Funktionen: OBDD-Größe $\text{poly}(n)$
- Effiziente Operationen für OBDDs \mathcal{G}_f und \mathcal{G}_h :
 - **Erfüllbarkeit:** $f \neq 0$ in Zeit $\mathcal{O}(1)$
 - **Äquivalenztest:** $f = h$ in Zeit $\mathcal{O}(|\mathcal{G}_f| + |\mathcal{G}_h|)$
 - **Variablenersetzung:** $f_{|x_i=0/1}$ Zeit $\mathcal{O}(|\mathcal{G}_f|)$
 - **Binäre Synthese:** $f \otimes h$ in Zeit $\mathcal{O}(|\mathcal{G}_f| \cdot |\mathcal{G}_h|)$
 - **Quantifizierung:** $(\exists/\forall x_i)f$ in Zeit $\mathcal{O}(|\mathcal{G}_f|^2)$

OBDDs – Eigenschaften und Operationen

- Jede Funktion f auf n Var. hat ein OBDD der Größe $(2 + o(1))2^n/n$.
- Hoffnung bei strukturierten Funktionen: OBDD-Größe $\text{poly}(n)$
- Effiziente Operationen für OBDDs \mathcal{G}_f und \mathcal{G}_h :
 - **Erfüllbarkeit:** $f \neq 0$ in Zeit $\mathcal{O}(1)$
 - **Äquivalenztest:** $f = h$ in Zeit $\mathcal{O}(|\mathcal{G}_f| + |\mathcal{G}_h|)$
 - **Variablenersetzung:** $f_{|x_i=0/1}$ Zeit $\mathcal{O}(|\mathcal{G}_f|)$
 - **Binäre Synthese:** $f \otimes h$ in Zeit $\mathcal{O}(|\mathcal{G}_f| \cdot |\mathcal{G}_h|)$
 - **Quantifizierung:** $(\exists/\forall x_i)f$ in Zeit $\mathcal{O}(|\mathcal{G}_f|^2)$

OBDDs – Eigenschaften und Operationen

- Jede Funktion f auf n Var. hat ein OBDD der Größe $(2 + o(1))2^n/n$.
- Hoffnung bei strukturierten Funktionen: OBDD-Größe $\text{poly}(n)$
- Effiziente Operationen für OBDDs \mathcal{G}_f und \mathcal{G}_h :
 - **Erfüllbarkeit:** $f \neq 0$ in Zeit $\mathcal{O}(1)$
 - **Äquivalenztest:** $f = h$ in Zeit $\mathcal{O}(|\mathcal{G}_f| + |\mathcal{G}_h|)$
 - **Variablenersetzung:** $f_{|x_i=0/1}$ Zeit $\mathcal{O}(|\mathcal{G}_f|)$
 - **Binäre Synthese:** $f \otimes h$ in Zeit $\mathcal{O}(|\mathcal{G}_f| \cdot |\mathcal{G}_h|)$
 - **Quantifizierung:** $(\exists/\forall x_i)f$ in Zeit $\mathcal{O}(|\mathcal{G}_f|^2)$

OBDDs – Eigenschaften und Operationen

- Jede Funktion f auf n Var. hat ein OBDD der Größe $(2 + o(1))2^n/n$.
- Hoffnung bei strukturierten Funktionen: OBDD-Größe $\text{poly}(n)$
- Effiziente Operationen für OBDDs \mathcal{G}_f und \mathcal{G}_h :
 - **Erfüllbarkeit:** $f \neq 0$ in Zeit $\mathcal{O}(1)$
 - **Äquivalenztest:** $f = h$ in Zeit $\mathcal{O}(|\mathcal{G}_f| + |\mathcal{G}_h|)$
 - **Variablenersetzung:** $f_{|x_i=0/1}$ Zeit $\mathcal{O}(|\mathcal{G}_f|)$
 - **Binäre Synthese:** $f \otimes h$ in Zeit $\mathcal{O}(|\mathcal{G}_f| \cdot |\mathcal{G}_h|)$
 - **Quantifizierung:** $(\exists/\forall x_i)f$ in Zeit $\mathcal{O}(|\mathcal{G}_f|^2)$

Implizite Darstellung mit OBDDs

- Fasse Daten auf als Teilmenge $S \subseteq \{0, \dots, N-1\}^k$ mit $N = 2^n$.
- Stelle S dar durch $\chi_S: \{0, 1\}^{nk} \rightarrow \{0, 1\}$ mit

$$\chi_S(x^{(1)}, \dots, x^{(k)}) = 1 \Leftrightarrow (|x^{(1)}|, \dots, |x^{(k)}|) \in S$$

für Vektoren $x^{(1)}, \dots, x^{(k)} \in \{0, 1\}^n$.

- Für Digraphen $G = (V, E)$ mit $V = \{v_0, \dots, v_{N-1}\}$:

$$\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E$$

- Mit Gewichten/Kapazitäten $c: E \rightarrow \mathbb{N}$:

$$\chi_G(x, y, a) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E \wedge c(v_{|x|}, v_{|y|}) = |a|$$

Implizite Darstellung mit OBDDs

- Fasse Daten auf als Teilmenge $S \subseteq \{0, \dots, N-1\}^k$ mit $N = 2^n$.
- Stelle S dar durch $\chi_S: \{0, 1\}^{nk} \rightarrow \{0, 1\}$ mit

$$\chi_S(x^{(1)}, \dots, x^{(k)}) = 1 \Leftrightarrow (|x^{(1)}|, \dots, |x^{(k)}|) \in S$$

für Vektoren $x^{(1)}, \dots, x^{(k)} \in \{0, 1\}^n$.

- Für Digraphen $G = (V, E)$ mit $V = \{v_0, \dots, v_{N-1}\}$:

$$\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E$$

- Mit Gewichten/Kapazitäten $c: E \rightarrow \mathbb{N}$:

$$\chi_G(x, y, a) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E \wedge c(v_{|x|}, v_{|y|}) = |a|$$

Implizite Darstellung mit OBDDs

- Fasse Daten auf als Teilmenge $S \subseteq \{0, \dots, N-1\}^k$ mit $N = 2^n$.
- Stelle S dar durch $\chi_S: \{0, 1\}^{nk} \rightarrow \{0, 1\}$ mit

$$\chi_S(x^{(1)}, \dots, x^{(k)}) = 1 \Leftrightarrow (|x^{(1)}|, \dots, |x^{(k)}|) \in S$$

für Vektoren $x^{(1)}, \dots, x^{(k)} \in \{0, 1\}^n$.

- Für Digraphen $G = (V, E)$ mit $V = \{v_0, \dots, v_{N-1}\}$:

$$\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E$$

- Mit Gewichten/Kapazitäten $c: E \rightarrow \mathbb{N}$:

$$\chi_G(x, y, a) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E \wedge c(v_{|x|}, v_{|y|}) = |a|$$

Implizite Darstellung mit OBDDs

- Fasse Daten auf als Teilmenge $S \subseteq \{0, \dots, N-1\}^k$ mit $N = 2^n$.
- Stelle S dar durch $\chi_S: \{0, 1\}^{nk} \rightarrow \{0, 1\}$ mit

$$\chi_S(x^{(1)}, \dots, x^{(k)}) = 1 \Leftrightarrow (|x^{(1)}|, \dots, |x^{(k)}|) \in S$$

für Vektoren $x^{(1)}, \dots, x^{(k)} \in \{0, 1\}^n$.

- Für Digraphen $G = (V, E)$ mit $V = \{v_0, \dots, v_{N-1}\}$:

$$\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E$$

- Mit Gewichten/Kapazitäten $c: E \rightarrow \mathbb{N}$:

$$\chi_G(x, y, a) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E \wedge c(v_{|x|}, v_{|y|}) = |a|$$

Angangssituation

- OBDDs finden viele erfolgreiche Anwendungen in CAD, Model Checking, Integer Programming, Relationale Algebra, ...
- Algorithmen sind reine Heuristiken.
- Alle Analysen sind experimentell oder zu grob.
- Behandelte Probleme sind anwendungsspezifisch.
- 1997: Hachtel, Somenzi et al. stellen OBDD-Algorithmen für bekannte Graphprobleme vor.
- ⇒ Untersuche praktischen Erfolg aus theoretischer Sicht!

Angangssituation

- OBDDs finden viele erfolgreiche Anwendungen in CAD, Model Checking, Integer Programming, Relationale Algebra, ...
- Algorithmen sind reine Heuristiken.
- Alle Analysen sind experimentell oder zu grob.
- Behandelte Probleme sind anwendungsspezifisch.
- 1997: Hachtel, Somenzi et al. stellen OBDD-Algorithmen für bekannte Graphprobleme vor.
- ⇒ Untersuche praktischen Erfolg aus theoretischer Sicht!

Ausgangssituation

- OBDDs finden viele erfolgreiche Anwendungen in CAD, Model Checking, Integer Programming, Relationale Algebra, ...
- Algorithmen sind reine Heuristiken.
- Alle Analysen sind experimentell oder zu grob.
- Behandelte Probleme sind anwendungsspezifisch.
- 1997: Hachtel, Somenzi et al. stellen OBDD-Algorithmen für bekannte Graphprobleme vor.
- ⇒ Untersuche praktischen Erfolg aus theoretischer Sicht!

Ausgangssituation

- OBDDs finden viele erfolgreiche Anwendungen in CAD, Model Checking, Integer Programming, Relationale Algebra, ...
- Algorithmen sind reine Heuristiken.
- Alle Analysen sind experimentell oder zu grob.
- Behandelte Probleme sind anwendungsspezifisch.
- 1997: Hachtel, Somenzi et al. stellen OBDD-Algorithmen für bekannte Graphprobleme vor.
- ⇒ Untersuche praktischen Erfolg aus theoretischer Sicht!

Ausgangssituation

- OBDDs finden viele erfolgreiche Anwendungen in CAD, Model Checking, Integer Programming, Relationale Algebra, ...
- Algorithmen sind reine Heuristiken.
- Alle Analysen sind experimentell oder zu grob.
- Behandelte Probleme sind anwendungsspezifisch.
- 1997: Hachtel, Somenzi et al. stellen OBDD-Algorithmen für bekannte Graphprobleme vor.
- ⇒ Untersuche praktischen Erfolg aus theoretischer Sicht!

Ausgangssituation

- OBDDs finden viele erfolgreiche Anwendungen in CAD, Model Checking, Integer Programming, Relationale Algebra, ...
- Algorithmen sind reine Heuristiken.
- Alle Analysen sind experimentell oder zu grob.
- Behandelte Probleme sind anwendungsspezifisch.
- 1997: Hachtel, Somenzi et al. stellen OBDD-Algorithmen für bekannte Graphprobleme vor.
- \Rightarrow Untersuche praktischen Erfolg aus theoretischer Sicht!

Implizite Graphalgorithmen

- **Implizite Graphalgorithmen** erhalten OBDD für χ_G .
- Kanten-/Knotenmengen, Ausgaben werden durch OBDDs dargestellt.
- Möglichst wenige Elemente werden einzeln behandelt.
- Möglichst viele Elemente werden von wenigen OBDD-Operationen verarbeitet.
- Ziele: Wenige Operationen und kleine OBDDs.
- Problem: Ziele oft gegenläufig.

Implizite Graphalgorithmen

- **Implizite Graphalgorithmen** erhalten OBDD für χ_G .
- Kanten-/Knotenmengen, Ausgaben werden durch OBDDs dargestellt.
- Möglichst wenige Elemente werden einzeln behandelt.
- Möglichst viele Elemente werden von wenigen OBDD-Operationen verarbeitet.
- Ziele: Wenige Operationen und kleine OBDDs.
- Problem: Ziele oft gegenläufig.

Implizite Graphalgorithmen

- **Implizite Graphalgorithmen** erhalten OBDD für χ_G .
- Kanten-/Knotenmengen, Ausgaben werden durch OBDDs dargestellt.
- Möglichst wenige Elemente werden einzeln behandelt.
- Möglichst viele Elemente werden von wenigen OBDD-Operationen verarbeitet.
- Ziele: Wenige Operationen und kleine OBDDs.
- Problem: Ziele oft gegenläufig.

Implizite Graphalgorithmen

- **Implizite Graphalgorithmen** erhalten OBDD für χ_G .
- Kanten-/Knotenmengen, Ausgaben werden durch OBDDs dargestellt.
- Möglichst wenige Elemente werden einzeln behandelt.
- Möglichst viele Elemente werden von wenigen OBDD-Operationen verarbeitet.
- Ziele: Wenige Operationen und kleine OBDDs.
- Problem: Ziele oft gegenläufig.

Implizite Graphalgorithmen

- **Implizite Graphalgorithmen** erhalten OBDD für χ_G .
- Kanten-/Knotenmengen, Ausgaben werden durch OBDDs dargestellt.
- Möglichst wenige Elemente werden einzeln behandelt.
- Möglichst viele Elemente werden von wenigen OBDD-Operationen verarbeitet.
- Ziele: Wenige Operationen und kleine OBDDs.
- Problem: Ziele oft gegenläufig.

Implizite Graphalgorithmen

- **Implizite Graphalgorithmen** erhalten OBDD für χ_G .
- Kanten-/Knotenmengen, Ausgaben werden durch OBDDs dargestellt.
- Möglichst wenige Elemente werden einzeln behandelt.
- Möglichst viele Elemente werden von wenigen OBDD-Operationen verarbeitet.
- Ziele: Wenige Operationen und kleine OBDDs.
- Problem: Ziele oft gegenläufig.

OBDD-Operationen in Graphalgorithmen

- Logische OBDD-Operationen simulieren Mengenoperationen:

$$A := B \cap C \quad \approx \quad \chi_A(x) := \chi_B(x) \wedge \chi_C(x)$$

- Wichtig: Quantorensequenzen über $\Omega(\log |V|)$ Bits:

$$\chi_A(x) := (\exists y) \chi_B(x, y), \quad y \in \{0, 1\}^n$$

- Beispiel: Ein impliziter BFS-Algorithmus

$i := 0; R_0(x) := (|x| = s)$
repeat

until $R_i = R_{i-1}$

OBDD-Operationen in Graphalgorithmen

- Logische OBDD-Operationen simulieren Mengenoperationen:

$$A := B \cap C \quad \approx \quad \chi_A(x) := \chi_B(x) \wedge \chi_C(x)$$

- Wichtig: Quantorensequenzen über $\Omega(\log |V|)$ Bits:

$$\chi_A(x) := (\exists y)\chi_B(x, y), \quad y \in \{0, 1\}^n$$

- Beispiel: Ein impliziter BFS-Algorithmus

$i := 0; R_0(x) := (|x| = s)$
repeat

until $R_i = R_{i-1}$

OBDD-Operationen in Graphalgorithmen

- Logische OBDD-Operationen simulieren Mengenoperationen:

$$A := B \cap C \quad \approx \quad \chi_A(x) := \chi_B(x) \wedge \chi_C(x)$$

- Wichtig: Quantorensequenzen über $\Omega(\log |V|)$ Bits:

$$\chi_A(x) := (\exists y)\chi_B(x, y), \quad y \in \{0, 1\}^n$$

- Beispiel: Ein impliziter BFS-Algorithmus

$i := 0; R_0(x) := (|x| = s)$

repeat

$N(x) := (\exists y)[\chi_C(y, x) \wedge R_i(y) \wedge \overline{R_i(x)}]$

$R_{i+1}(x) := R_i(x) \vee N(x)$

$i := i + 1$

until $R_i = R_{i-1}$

OBDD-Operationen in Graphalgorithmen

- Logische OBDD-Operationen simulieren Mengenoperationen:

$$A := B \cap C \quad \approx \quad \chi_A(x) := \chi_B(x) \wedge \chi_C(x)$$

- Wichtig: Quantorensequenzen über $\Omega(\log |V|)$ Bits:

$$\chi_A(x) := (\exists y)\chi_B(x, y), \quad y \in \{0, 1\}^n$$

- Beispiel: Ein impliziter BFS-Algorithmus

$i := 0; R_0(x) := (|x| = s)$

repeat

$$N(x) := (\exists y)[\chi_C(y, x) \wedge R_i(y) \wedge \overline{R_i(x)}]$$

$$R_{i+1}(x) := R_i(x) \vee N(x)$$

$i := i + 1$

until $R_i = R_{i-1}$

OBDD-Operationen in Graphalgorithmen

- Logische OBDD-Operationen simulieren Mengenoperationen:

$$A := B \cap C \quad \approx \quad \chi_A(x) := \chi_B(x) \wedge \chi_C(x)$$

- Wichtig: Quantorensequenzen über $\Omega(\log |V|)$ Bits:

$$\chi_A(x) := (\exists y)\chi_B(x, y), \quad y \in \{0, 1\}^n$$

- Beispiel: Ein impliziter BFS-Algorithmus

$$i := 0; R_0(x) := (|x| = s)$$

repeat

$$N(x) := (\exists y)[\chi_G(y, x) \wedge R_i(y) \wedge \overline{R_i(x)}]$$

$$R_{i+1}(x) := R_i(x) \vee N(x)$$

$$i := i + 1$$

until $R_i = R_{i-1}$

OBDD-Operationen in Graphalgorithmen

- Logische OBDD-Operationen simulieren Mengenoperationen:

$$A := B \cap C \quad \approx \quad \chi_A(x) := \chi_B(x) \wedge \chi_C(x)$$

- Wichtig: Quantorensequenzen über $\Omega(\log |V|)$ Bits:

$$\chi_A(x) := (\exists y)\chi_B(x, y), \quad y \in \{0, 1\}^n$$

- Beispiel: Ein impliziter BFS-Algorithmus

$$i := 0; R_0(x) := (|x| = s)$$

repeat

$$N(x) := (\exists y)[\chi_G(y, x) \wedge R_i(y) \wedge \overline{R_i(x)}]$$

$$R_{i+1}(x) := R_i(x) \vee N(x)$$

$$i := i + 1$$

until $R_i = R_{i-1}$

OBDD-Operationen in Graphalgorithmen

- Logische OBDD-Operationen simulieren Mengenoperationen:

$$A := B \cap C \quad \approx \quad \chi_A(x) := \chi_B(x) \wedge \chi_C(x)$$

- Wichtig: Quantorensequenzen über $\Omega(\log |V|)$ Bits:

$$\chi_A(x) := (\exists y)\chi_B(x, y), \quad y \in \{0, 1\}^n$$

- Beispiel: Ein impliziter BFS-Algorithmus

$$i := 0; R_0(x) := (|x| = s)$$

repeat

$$N(x) := (\exists y)[\chi_G(y, x) \wedge R_i(y) \wedge \overline{R_i(x)}]$$

$$R_{i+1}(x) := R_i(x) \vee N(x)$$

$$i := i + 1$$

until $R_i = R_{i-1}$

OBDD-Operationen in Graphalgorithmen

- Logische OBDD-Operationen simulieren Mengenoperationen:

$$A := B \cap C \quad \approx \quad \chi_A(x) := \chi_B(x) \wedge \chi_C(x)$$

- Wichtig: Quantorensequenzen über $\Omega(\log |V|)$ Bits:

$$\chi_A(x) := (\exists y)\chi_B(x, y), \quad y \in \{0, 1\}^n$$

- Beispiel: Ein impliziter BFS-Algorithmus

$$i := 0; R_0(x) := (|x| = s)$$

repeat

$$N(x) := (\exists y)[\chi_G(y, x) \wedge R_i(y) \wedge \overline{R_i(x)}]$$

$$R_{i+1}(x) := R_i(x) \vee N(x)$$

$$i := i + 1$$

until $R_i = R_{i-1}$

OBDD-Operationen in Graphalgorithmen

- Logische OBDD-Operationen simulieren Mengenoperationen:

$$A := B \cap C \quad \approx \quad \chi_A(x) := \chi_B(x) \wedge \chi_C(x)$$

- Wichtig: Quantorensequenzen über $\Omega(\log |V|)$ Bits:

$$\chi_A(x) := (\exists y)\chi_B(x, y), \quad y \in \{0, 1\}^n$$

- Beispiel: Ein impliziter BFS-Algorithmus

$$i := 0; R_0(x) := (|x| = s)$$

repeat

$$N(x) := (\exists y)[\chi_G(y, x) \wedge R_i(y) \wedge \overline{R_i(x)}]$$

$$R_{i+1}(x) := R_i(x) \vee N(x)$$

$$i := i + 1$$

until $R_i = R_{i-1}$

Fragestellungen

- Die OBDD-Größe von Graphen
- Entwicklung impliziter Graphalgorithmen
- Die Anzahl der OBDD-Operationen
- Die OBDD-Größe von Zwischenergebnissen und Ausgabe
- Sichtweisen: “Klein” bedeutet $o(|V|)$ / $o(|E|)$ / $\text{poly}(\log |V|)$.
- Zeit / Platz bezüglich $|V|$ oder OBDD-Größe der Eingabe χ_G
- Theoretische worst-case Aussagen / experimentelle average-case Aussagen

Fragestellungen

- Die OBDD-Größe von Graphen
- Entwicklung impliziter Graphalgorithmen
- Die Anzahl der OBDD-Operationen
- Die OBDD-Größe von Zwischenergebnissen und Ausgabe
- Sichtweisen: “Klein” bedeutet $o(|V|)$ / $o(|E|)$ / $\text{poly}(\log |V|)$.
- Zeit / Platz bezüglich $|V|$ oder OBDD-Größe der Eingabe χ_G
- Theoretische worst-case Aussagen / experimentelle average-case Aussagen

Fragestellungen

- Die OBDD-Größe von Graphen
- Entwicklung impliziter Graphalgorithmen
- Die Anzahl der OBDD-Operationen
- Die OBDD-Größe von Zwischenergebnissen und Ausgabe
- Sichtweisen: “Klein” bedeutet $o(|V|)$ / $o(|E|)$ / $\text{poly}(\log |V|)$.
- Zeit / Platz bezüglich $|V|$ oder OBDD-Größe der Eingabe χ_G
- Theoretische worst-case Aussagen / experimentelle average-case Aussagen

Fragestellungen

- Die OBDD-Größe von Graphen
- Entwicklung impliziter Graphalgorithmen
- Die Anzahl der OBDD-Operationen
- Die OBDD-Größe von Zwischenergebnissen und Ausgabe
- Sichtweisen: “Klein” bedeutet $o(|V|)$ / $o(|E|)$ / $\text{poly}(\log |V|)$.
- Zeit / Platz bezüglich $|V|$ oder OBDD-Größe der Eingabe χ_G
- Theoretische worst-case Aussagen / experimentelle average-case Aussagen

Fragestellungen

- Die OBDD-Größe von Graphen
- Entwicklung impliziter Graphalgorithmen
- Die Anzahl der OBDD-Operationen
- Die OBDD-Größe von Zwischenergebnissen und Ausgabe
- Sichtweisen: “Klein” bedeutet $o(|V|)$ / $o(|E|)$ / $\text{poly}(\log |V|)$.
- Zeit / Platz bezüglich $|V|$ oder OBDD-Größe der Eingabe χ_G
- Theoretische worst-case Aussagen / experimentelle average-case Aussagen

Fragestellungen

- Die OBDD-Größe von Graphen
- Entwicklung impliziter Graphalgorithmen
- Die Anzahl der OBDD-Operationen
- Die OBDD-Größe von Zwischenergebnissen und Ausgabe
- Sichtweisen: “Klein” bedeutet $o(|V|)$ / $o(|E|)$ / $\text{poly}(\log |V|)$.
- Zeit / Platz bezüglich $|V|$ oder OBDD-Größe der Eingabe χ_G
- Theoretische worst-case Aussagen / experimentelle average-case Aussagen

Fragestellungen

- Die OBDD-Größe von Graphen
- Entwicklung impliziter Graphalgorithmen
- Die Anzahl der OBDD-Operationen
- Die OBDD-Größe von Zwischenergebnissen und Ausgabe
- Sichtweisen: “Klein” bedeutet $o(|V|)$ / $o(|E|)$ / $\text{poly}(\log |V|)$.
- Zeit / Platz bezüglich $|V|$ oder OBDD-Größe der Eingabe χ_G
- Theoretische worst-case Aussagen / experimentelle average-case Aussagen

Inhalt

- 1 OBDDs und Motivation
- 2 Anzahl der OBDD-Operationen**
- 3 Nichtexistenz von Festparameteralgorithmen
- 4 Konkrete exponentielle untere Schranken
- 5 Fazit und Ausblick

Operationsanzahl als grobes Laufzeitmaß

- Hauptaspekte impliziter Algorithmen: **Operationsanzahl** und **OBDD-Größen**
- Kleine OBDDs nur in Spezialfällen, Analyse schwierig
- Übliches Gütemaß: **Anzahl** der Operationen
 - Gentilini, Piazza, Policriti: Starker Zusammenhang (SODA'03)
 - Gentilini, Policriti: Zweizusammenhang (ISAAC'03)
 - ...
- Wenig aussagekräftige untere Schranke der Laufzeit
- Lediglich erster Schritt zu effizienten Algorithmen
- Eigener Anspruch: Möglichst $\mathcal{O}(\text{poly}(\log |V|))$ Operationen
- \Rightarrow Sequenzielles Vorgehen vermeiden

Operationsanzahl als grobes Laufzeitmaß

- Hauptaspekte impliziter Algorithmen: **Operationsanzahl** und **OBDD-Größen**
- Kleine OBDDs nur in Spezialfällen, Analyse schwierig
- Übliches Gütemaß: **Anzahl** der Operationen
 - Gentilini, Piazza, Policriti: Starker Zusammenhang (SODA'03)
 - Gentilini, Policriti: Zweizusammenhang (ISAAC'03)
 - ...
- Wenig aussagekräftige untere Schranke der Laufzeit
- Lediglich erster Schritt zu effizienten Algorithmen
- Eigener Anspruch: Möglichst $\mathcal{O}(\text{poly}(\log |V|))$ Operationen
- \Rightarrow Sequenzielles Vorgehen vermeiden

Operationsanzahl als grobes Laufzeitmaß

- Hauptaspekte impliziter Algorithmen: **Operationsanzahl** und **OBDD-Größen**
- Kleine OBDDs nur in Spezialfällen, Analyse schwierig
- Übliches Gütemaß: **Anzahl** der Operationen
 - Gentilini, Piazza, Policriti: Starker Zusammenhang (SODA'03)
 - Gentilini, Policriti: Zweizusammenhang (ISAAC'03)
 - ...
- Wenig aussagekräftige untere Schranke der Laufzeit
- Lediglich erster Schritt zu effizienten Algorithmen
- Eigener Anspruch: Möglichst $\mathcal{O}(\text{poly}(\log |V|))$ Operationen
- \Rightarrow Sequenzielles Vorgehen vermeiden

Operationsanzahl als grobes Laufzeitmaß

- Hauptaspekte impliziter Algorithmen: **Operationsanzahl** und **OBDD-Größen**
- Kleine OBDDs nur in Spezialfällen, Analyse schwierig
- Übliches Gütemaß: **Anzahl** der Operationen
 - Gentilini, Piazza, Policriti: Starker Zusammenhang (SODA'03)
 - Gentilini, Policriti: Zweizusammenhang (ISAAC'03)
 - ...
- Wenig aussagekräftige untere Schranke der Laufzeit
- Lediglich erster Schritt zu effizienten Algorithmen
- Eigener Anspruch: Möglichst $\mathcal{O}(\text{poly}(\log |V|))$ Operationen
- \Rightarrow Sequenzielles Vorgehen vermeiden

Operationsanzahl als grobes Laufzeitmaß

- Hauptaspekte impliziter Algorithmen: **Operationsanzahl** und **OBDD-Größen**
- Kleine OBDDs nur in Spezialfällen, Analyse schwierig
- Übliches Gütemaß: **Anzahl** der Operationen
 - Gentilini, Piazza, Policriti: Starker Zusammenhang (SODA'03)
 - Gentilini, Policriti: Zweizusammenhang (ISAAC'03)
 - ...
- Wenig aussagekräftige untere Schranke der Laufzeit
- Lediglich erster Schritt zu effizienten Algorithmen
- Eigener Anspruch: Möglichst $\mathcal{O}(\text{poly}(\log |V|))$ Operationen
- \Rightarrow Sequenzielles Vorgehen vermeiden

Operationsanzahl als grobes Laufzeitmaß

- Hauptaspekte impliziter Algorithmen: **Operationsanzahl** und **OBDD-Größen**
- Kleine OBDDs nur in Spezialfällen, Analyse schwierig
- Übliches Gütemaß: **Anzahl** der Operationen
 - Gentilini, Piazza, Policriti: Starker Zusammenhang (SODA'03)
 - Gentilini, Policriti: Zweizusammenhang (ISAAC'03)
 - ...
- Wenig aussagekräftige untere Schranke der Laufzeit
- Lediglich erster Schritt zu effizienten Algorithmen
- Eigener Anspruch: Möglichst $\mathcal{O}(\text{poly}(\log |V|))$ Operationen
- \Rightarrow Sequenzielles Vorgehen vermeiden

Operationsanzahl als grobes Laufzeitmaß

- Hauptaspekte impliziter Algorithmen: **Operationsanzahl** und **OBDD-Größen**
- Kleine OBDDs nur in Spezialfällen, Analyse schwierig
- Übliches Gütemaß: **Anzahl** der Operationen
 - Gentilini, Piazza, Policriti: Starker Zusammenhang (SODA'03)
 - Gentilini, Policriti: Zweizusammenhang (ISAAC'03)
 - ...
- Wenig aussagekräftige untere Schranke der Laufzeit
 - Lediglich erster Schritt zu effizienten Algorithmen
 - Eigener Anspruch: Möglichst $\mathcal{O}(\text{poly}(\log |V|))$ Operationen
 - \Rightarrow Sequenzielles Vorgehen vermeiden

Operationsanzahl als grobes Laufzeitmaß

- Hauptaspekte impliziter Algorithmen: **Operationsanzahl** und **OBDD-Größen**
- Kleine OBDDs nur in Spezialfällen, Analyse schwierig
- Übliches Gütemaß: **Anzahl** der Operationen
 - Gentilini, Piazza, Policriti: Starker Zusammenhang (SODA'03)
 - Gentilini, Policriti: Zweizusammenhang (ISAAC'03)
 - ...
- Wenig aussagekräftige untere Schranke der Laufzeit
- Lediglich erster Schritt zu effizienten Algorithmen
- Eigener Anspruch: Möglichst $\mathcal{O}(\text{poly}(\log |V|))$ Operationen
- \Rightarrow Sequenzielles Vorgehen vermeiden

Operationsanzahl als grobes Laufzeitmaß

- Hauptaspekte impliziter Algorithmen: **Operationsanzahl** und **OBDD-Größen**
- Kleine OBDDs nur in Spezialfällen, Analyse schwierig
- Übliches Gütemaß: **Anzahl** der Operationen
 - Gentilini, Piazza, Policriti: Starker Zusammenhang (SODA'03)
 - Gentilini, Policriti: Zweizusammenhang (ISAAC'03)
 - ...
- Wenig aussagekräftige untere Schranke der Laufzeit
- Lediglich erster Schritt zu effizienten Algorithmen
- Eigener Anspruch: Möglichst $\mathcal{O}(\text{poly}(\log |V|))$ Operationen
- \Rightarrow Sequenzielles Vorgehen vermeiden

Operationsanzahl als grobes Laufzeitmaß

- Hauptaspekte impliziter Algorithmen: **Operationsanzahl** und **OBDD-Größen**
- Kleine OBDDs nur in Spezialfällen, Analyse schwierig
- Übliches Gütemaß: **Anzahl** der Operationen
 - Gentilini, Piazza, Policriti: Starker Zusammenhang (SODA'03)
 - Gentilini, Policriti: Zweizusammenhang (ISAAC'03)
 - ...
- Wenig aussagekräftige untere Schranke der Laufzeit
- Lediglich erster Schritt zu effizienten Algorithmen
- Eigener Anspruch: Möglichst $\mathcal{O}(\text{poly}(\log |V|))$ Operationen
- \Rightarrow Sequenzielles Vorgehen vermeiden

Iteratives Quadrieren

- Einfaches Prinzip: Verdopple logarithmisch oft betrachtete Größen
- Beispiel: **Transitiver Abschluss** $C(x, y)$ von $\chi_G(x, y)$

$i := 0$

$C_0(x, y) := \chi_G(x, y)$

repeat

until $C_i = C_{i-1}$

- $\mathcal{O}(\log^2 |V|)$ OBDD-Operationen (wegen $(\exists z_{n-1}, \dots, z_0)$)

Iteratives Quadrieren

- Einfaches Prinzip: Verdopple logarithmisch oft betrachtete Größen
- Beispiel: **Transitiver Abschluss** $C(x, y)$ von $\chi_G(x, y)$

$i := 0$

$C_0(x, y) := \chi_G(x, y)$

repeat

$C_{i+1}(x, y) := (\exists z)[C_i(x, z) \wedge C_i(z, y)]$

$i := i + 1$

until $C_i = C_{i-1}$

- $\mathcal{O}(\log^2 |V|)$ OBDD-Operationen (wegen $(\exists z_{n-1}, \dots, z_0)$)

Iteratives Quadrieren

- Einfaches Prinzip: Verdopple logarithmisch oft betrachtete Größen
- Beispiel: **Transitiver Abschluss** $C(x, y)$ von $\chi_G(x, y)$

$i := 0$

$C_0(x, y) := \chi_G(x, y)$

repeat

$C_{i+1}(x, y) := (\exists z)[C_i(x, z) \wedge C_i(z, y)]$

$i := i + 1$

until $C_i = C_{i-1}$

- $\mathcal{O}(\log^2 |V|)$ OBDD-Operationen (wegen $(\exists z_{n-1}, \dots, z_0)$)

Iteratives Quadrieren

- Einfaches Prinzip: Verdopple logarithmisch oft betrachtete Größen
- Beispiel: **Transitiver Abschluss** $C(x, y)$ von $\chi_G(x, y)$

$i := 0$

$C_0(x, y) := \chi_G(x, y)$

repeat

$C_{i+1}(x, y) := (\exists z)[C_i(x, z) \wedge C_i(z, y)]$

$i := i + 1$

until $C_i = C_{i-1}$

- $\mathcal{O}(\log^2 |V|)$ OBDD-Operationen (wegen $(\exists z_{n-1}, \dots, z_0)$)

Iteratives Quadrieren

- Einfaches Prinzip: Verdopple logarithmisch oft betrachtete Größen
- Beispiel: **Transitiver Abschluss** $C(x, y)$ von $\chi_G(x, y)$

$i := 0$

$C_0(x, y) := \chi_G(x, y)$

repeat

$C_{i+1}(x, y) := (\exists z)[C_i(x, z) \wedge C_i(z, y)]$

$i := i + 1$

until $C_i = C_{i-1}$

- $\mathcal{O}(\log^2 |V|)$ OBDD-Operationen (wegen $(\exists z_{n-1}, \dots, z_0)$)

Iteratives Quadrieren

- Einfaches Prinzip: Verdopple logarithmisch oft betrachtete Größen
- Beispiel: **Transitiver Abschluss** $C(x, y)$ von $\chi_G(x, y)$

$i := 0$

$C_0(x, y) := \chi_G(x, y)$

repeat

$C_{i+1}(x, y) := (\exists z)[C_i(x, z) \wedge C_i(z, y)]$

$i := i + 1$

until $C_i = C_{i-1}$

- $\mathcal{O}(\log^2 |V|)$ OBDD-Operationen (wegen $(\exists z_{n-1}, \dots, z_0)$)

Iteratives Quadrieren

- Einfaches Prinzip: Verdopple logarithmisch oft betrachtete Größen
- Beispiel: **Transitiver Abschluss** $C(x, y)$ von $\chi_G(x, y)$

$i := 0$

$C_0(x, y) := \chi_G(x, y)$

repeat

$C_{i+1}(x, y) := (\exists z)[C_i(x, z) \wedge C_i(z, y)]$

$i := i + 1$

until $C_i = C_{i-1}$

- $\mathcal{O}(\log^2 |V|)$ OBDD-Operationen (wegen $(\exists z_{n-1}, \dots, z_0)$)

Iteratives Quadrieren

- Einfaches Prinzip: Verdopple logarithmisch oft betrachtete Größen
- Beispiel: **Transitiver Abschluss** $C(x, y)$ von $\chi_G(x, y)$

$i := 0$

$C_0(x, y) := \chi_G(x, y)$

repeat

$C_{i+1}(x, y) := (\exists z)[C_i(x, z) \wedge C_i(z, y)]$

$i := i + 1$

until $C_i = C_{i-1}$

- $\mathcal{O}(\log^2 |V|)$ OBDD-Operationen (wegen $(\exists z_{n-1}, \dots, z_0)$)

Iteratives Quadrieren

- Einfaches Prinzip: Verdopple logarithmisch oft betrachtete Größen
- Beispiel: **Transitiver Abschluss** $C(x, y)$ von $\chi_G(x, y)$

$i := 0$

$C_0(x, y) := \chi_G(x, y)$

repeat

$C_{i+1}(x, y) := (\exists z)[C_i(x, z) \wedge C_i(z, y)]$

$i := i + 1$

until $C_i = C_{i-1}$

- $\mathcal{O}(\log^2 |V|)$ OBDD-Operationen (wegen $(\exists z_{n-1}, \dots, z_0)$)

Algorithmen mit $\mathcal{O}(n^k)$ Operationen

- Topologisches Sortieren (Wölfel, MFCS'03)
- All-Pairs Shortest-Paths (S., WG'04)
- Unveröffentlicht:
 - s - t -Zusammenhang
 - Bipartitheit
 - Kreisfreiheit
 - Eulerkreis
 - Minimale Spannbäume
 - ...

Algorithmen mit $\mathcal{O}(n^k)$ Operationen

- Topologisches Sortieren (Wölfel, MFCS'03)
- All-Pairs Shortest-Paths (S., WG'04)
- Unveröffentlicht:
 - s - t -Zusammenhang
 - Bipartitheit
 - Kreisfreiheit
 - Eulerkreis
 - Minimale Spannbäume
 - ...

Algorithmen mit $\mathcal{O}(n^k)$ Operationen

- Topologisches Sortieren (Wölfel, MFCS'03)
- All-Pairs Shortest-Paths (S., WG'04)
- Unveröffentlicht:
 - s - t -Zusammenhang
 - Bipartitheit
 - Kreisfreiheit
 - Eulerkreis
 - Minimale Spannbäume
 - ...

Algorithmen mit $\mathcal{O}(n^k)$ Operationen

- Topologisches Sortieren (Wölfel, MFCS'03)
- All-Pairs Shortest-Paths (S., WG'04)
- Unveröffentlicht:
 - s - t -Zusammenhang
 - Bipartitheit
 - Kreisfreiheit
 - Eulerkreis
 - Minimale Spannbäume
 - ...

Algorithmen mit $\mathcal{O}(n^k)$ Operationen

- Topologisches Sortieren (Wölfel, MFCS'03)
- All-Pairs Shortest-Paths (S., WG'04)
- Unveröffentlicht:
 - s - t -Zusammenhang
 - Bipartitheit
 - Kreisfreiheit
 - Eulerkreis
 - Minimale Spannbäume
 - ...

Algorithmen mit $\mathcal{O}(n^k)$ Operationen

- Topologisches Sortieren (Wölfel, MFCS'03)
- All-Pairs Shortest-Paths (S., WG'04)
- Unveröffentlicht:
 - s - t -Zusammenhang
 - Bipartitheit
 - Kreisfreiheit
 - Eulerkreis
 - Minimale Spannbäume
 - ...

Algorithmen mit $\mathcal{O}(n^k)$ Operationen

- Topologisches Sortieren (Wölfel, MFCS'03)
- All-Pairs Shortest-Paths (S., WG'04)
- Unveröffentlicht:
 - s - t -Zusammenhang
 - Bipartitheit
 - Kreisfreiheit
 - Eulerkreis
 - Minimale Spannbäume
 - ...

Algorithmen mit $\mathcal{O}(n^k)$ Operationen

- Topologisches Sortieren (Wölfel, MFCS'03)
- All-Pairs Shortest-Paths (S., WG'04)
- Unveröffentlicht:
 - s - t -Zusammenhang
 - Bipartitheit
 - Kreisfreiheit
 - Eulerkreis
 - Minimale Spannbäume
 - ...

Algorithmen mit $\mathcal{O}(n^k)$ Operationen

- Topologisches Sortieren (Wölfel, MFCS'03)
- All-Pairs Shortest-Paths (S., WG'04)
- Unveröffentlicht:
 - s - t -Zusammenhang
 - Bipartitheit
 - Kreisfreiheit
 - Eulerkreis
 - Minimale Spannbäume
 - ...

Implizite vs. parallele Algorithmen

- Beobachtung: Implizite Algorithmen ähneln parallelen Algorithmen.
- Bsp.: $\chi_G(x, y) \wedge P(x)$ behandelt alle $x \in \{0, 1\}^n$ „parallel“.
- Beziehung zwischen beiden Konzepten?
Formalisiere **Symbolische RAM**:
- Arbeitsregister R_0, R_1, \dots und Funktionsregister S_0, S_1, \dots
- Eingabe $\chi_I(x) = I_{|x|}$ und Ausgabe $\chi_O(x) = O_{|x|}$ in S_0 ,
 $N := |I| = 2^n$, $\chi_I, \chi_O: \{0, 1\}^n \rightarrow \{0, 1\}$

Satz

SRAM \mathcal{M} mit Zeit $t_{\mathcal{M}}(N)$ kann von PRAM \mathcal{M}' mit N^k Prozessoren in Zeit $\mathcal{O}((t_{\mathcal{M}}(N))^2 \cdot \log^2 N)$ simuliert werden.

Implizite vs. parallele Algorithmen

- Beobachtung: Implizite Algorithmen ähneln parallelen Algorithmen.
- Bsp.: $\chi_G(x, y) \wedge P(x)$ behandelt alle $x \in \{0, 1\}^n$ „parallel“.
- Beziehung zwischen beiden Konzepten?
Formalisiere **Symbolische RAM**:
- Arbeitsregister R_0, R_1, \dots und Funktionsregister S_0, S_1, \dots
- Eingabe $\chi_I(x) = I_{|x|}$ und Ausgabe $\chi_O(x) = O_{|x|}$ in S_0 ,
 $N := |I| = 2^n$, $\chi_I, \chi_O: \{0, 1\}^n \rightarrow \{0, 1\}$

Satz

SRAM \mathcal{M} mit Zeit $t_{\mathcal{M}}(N)$ kann von PRAM \mathcal{M}' mit N^k Prozessoren in Zeit $\mathcal{O}((t_{\mathcal{M}}(N))^2 \cdot \log^2 N)$ simuliert werden.

Implizite vs. parallele Algorithmen

- Beobachtung: Implizite Algorithmen ähneln parallelen Algorithmen.
- Bsp.: $\chi_G(x, y) \wedge P(x)$ behandelt alle $x \in \{0, 1\}^n$ „parallel“.
- Beziehung zwischen beiden Konzepten?
Formalisiere **Symbolische RAM**:
 - Arbeitsregister R_0, R_1, \dots und Funktionsregister S_0, S_1, \dots
 - Eingabe $\chi_I(x) = I_{|x|}$ und Ausgabe $\chi_O(x) = O_{|x|}$ in S_0 ,
 $N := |I| = 2^n$, $\chi_I, \chi_O: \{0, 1\}^n \rightarrow \{0, 1\}$

Satz

SRAM \mathcal{M} mit Zeit $t_{\mathcal{M}}(N)$ kann von PRAM \mathcal{M}' mit N^k Prozessoren in Zeit $\mathcal{O}((t_{\mathcal{M}}(N))^2 \cdot \log^2 N)$ simuliert werden.

Implizite vs. parallele Algorithmen

- Beobachtung: Implizite Algorithmen ähneln parallelen Algorithmen.
- Bsp.: $\chi_G(x, y) \wedge P(x)$ behandelt alle $x \in \{0, 1\}^n$ „parallel“.
- Beziehung zwischen beiden Konzepten?
Formalisiere **Symbolische RAM**:
- Arbeitsregister R_0, R_1, \dots und Funktionsregister S_0, S_1, \dots
- Eingabe $\chi_I(x) = I_{|x|}$ und Ausgabe $\chi_O(x) = O_{|x|}$ in S_0 ,
 $N := |I| = 2^n$, $\chi_I, \chi_O: \{0, 1\}^n \rightarrow \{0, 1\}$

Satz

SRAM \mathcal{M} mit Zeit $t_{\mathcal{M}}(N)$ kann von PRAM \mathcal{M}' mit N^k Prozessoren in Zeit $\mathcal{O}((t_{\mathcal{M}}(N))^2 \cdot \log^2 N)$ simuliert werden.

Implizite vs. parallele Algorithmen

- Beobachtung: Implizite Algorithmen ähneln parallelen Algorithmen.
- Bsp.: $\chi_G(x, y) \wedge P(x)$ behandelt alle $x \in \{0, 1\}^n$ „parallel“.
- Beziehung zwischen beiden Konzepten?
Formalisiere **Symbolische RAM**:
- Arbeitsregister R_0, R_1, \dots und Funktionsregister S_0, S_1, \dots
- Eingabe $\chi_I(x) = I_{|x|}$ und Ausgabe $\chi_O(x) = O_{|x|}$ in S_0 ,
 $N := |I| = 2^n$, $\chi_I, \chi_O: \{0, 1\}^n \rightarrow \{0, 1\}$

Satz

SRAM \mathcal{M} mit Zeit $t_{\mathcal{M}}(N)$ kann von PRAM \mathcal{M}' mit N^k Prozessoren in Zeit $\mathcal{O}((t_{\mathcal{M}}(N))^2 \cdot \log^2 N)$ simuliert werden.

Implizite vs. parallele Algorithmen

- Beobachtung: Implizite Algorithmen ähneln parallelen Algorithmen.
- Bsp.: $\chi_G(x, y) \wedge P(x)$ behandelt alle $x \in \{0, 1\}^n$ „parallel“.
- Beziehung zwischen beiden Konzepten?
Formalisiere **Symbolische RAM**:
- Arbeitsregister R_0, R_1, \dots und Funktionsregister S_0, S_1, \dots
- Eingabe $\chi_I(x) = I_{|x|}$ und Ausgabe $\chi_O(x) = O_{|x|}$ in S_0 ,
 $N := |I| = 2^n$, $\chi_I, \chi_O: \{0, 1\}^n \rightarrow \{0, 1\}$

Satz

SRAM \mathcal{M} mit Zeit $t_{\mathcal{M}}(N)$ kann von PRAM \mathcal{M}' mit N^k Prozessoren in Zeit $\mathcal{O}((t_{\mathcal{M}}(N))^2 \cdot \log^2 N)$ simuliert werden.

Beweisidee

Satz

SRAM \mathcal{M} mit Zeit $t_{\mathcal{M}}(N)$ kann von PRAM \mathcal{M}' mit N^k Prozessoren in Zeit $\mathcal{O}((t_{\mathcal{M}}(N))^2 \cdot \log^2 N)$ simuliert werden.

- Prozessor P_a verwaltet $S_0(a), \dots, S_{\leq t_{\mathcal{M}}(N)}(a)$ für Belegung $a \in \{0, 1\}^n$.
- Simulation jeder symb. Operation in Zeit $\mathcal{O}(t_{\mathcal{M}}(N) \cdot \log^2 N)$.
- Bsp.: $S_i \wedge S_j$ entspricht $S_i(a) \wedge S_j(a)$ für alle P_a .

Beweisidee

Satz

SRAM \mathcal{M} mit Zeit $t_{\mathcal{M}}(N)$ kann von PRAM \mathcal{M}' mit N^k Prozessoren in Zeit $\mathcal{O}((t_{\mathcal{M}}(N))^2 \cdot \log^2 N)$ simuliert werden.

- Prozessor P_a verwaltet $S_0(a), \dots, S_{\leq t_{\mathcal{M}}(N)}(a)$ für Belegung $a \in \{0, 1\}^n$.
- Simulation jeder symb. Operation in Zeit $\mathcal{O}(t_{\mathcal{M}}(N) \cdot \log^2 N)$.
- Bsp.: $S_i \wedge S_j$ entspricht $S_i(a) \wedge S_j(a)$ für alle P_a .

Beweisidee

Satz

SRAM \mathcal{M} mit Zeit $t_{\mathcal{M}}(N)$ kann von PRAM \mathcal{M}' mit N^k Prozessoren in Zeit $\mathcal{O}((t_{\mathcal{M}}(N))^2 \cdot \log^2 N)$ simuliert werden.

- Prozessor P_a verwaltet $S_0(a), \dots, S_{\leq t_{\mathcal{M}}(N)}(a)$ für Belegung $a \in \{0, 1\}^n$.
- Simulation jeder symb. Operation in Zeit $\mathcal{O}(t_{\mathcal{M}}(N) \cdot \log^2 N)$.
- Bsp.: $S_i \wedge S_j$ entspricht $S_i(a) \wedge S_j(a)$ für alle P_a .

Ergebnis für P-vollständige Probleme

Falls $P \neq NC$:

Satz

P-vollständige Probleme haben keine parallelen Algo. mit Zeit $\mathcal{O}(\log^k N)$ auf $\mathcal{O}(N^k)$ Prozessoren.

Folgerung

P-vollständige Probleme haben keine symbolischen Algorithmen mit $\mathcal{O}(\log^k N)$ logischen Operationen.

- Resultat ist unabhängig von der Funktionsdatenstruktur (OBDDs, etc.).
- Simulation paralleler Algorithmen erscheint schwierig.
- Offen: 0-1 Flussmaximierung $\in NC$?

Ergebnis für P-vollständige Probleme

Falls $P \neq NC$:

Satz

P-vollständige Probleme haben keine parallelen Algo. mit Zeit $\mathcal{O}(\log^k N)$ auf $\mathcal{O}(N^k)$ Prozessoren.

Folgerung

P-vollständige Probleme haben keine symbolischen Algorithmen mit $\mathcal{O}(\log^k N)$ logischen Operationen.

- Resultat ist unabhängig von der Funktionsdatenstruktur (OBDDs, etc.).
- Simulation paralleler Algorithmen erscheint schwierig.
- Offen: 0-1 Flussmaximierung $\in NC$?

Ergebnis für P-vollständige Probleme

Falls $P \neq NC$:

Satz

P-vollständige Probleme haben keine parallelen Algo. mit Zeit $\mathcal{O}(\log^k N)$ auf $\mathcal{O}(N^k)$ Prozessoren.

Folgerung

P-vollständige Probleme haben keine symbolischen Algorithmen mit $\mathcal{O}(\log^k N)$ logischen Operationen.

- Resultat ist unabhängig von der Funktionsdatenstruktur (OBDDs, etc.).
- Simulation paralleler Algorithmen erscheint schwierig.
- Offen: 0-1 Flussmaximierung $\in NC$?

Ergebnis für P-vollständige Probleme

Falls $P \neq NC$:

Satz

P-vollständige Probleme haben keine parallelen Algo. mit Zeit $\mathcal{O}(\log^k N)$ auf $\mathcal{O}(N^k)$ Prozessoren.

Folgerung

P-vollständige Probleme haben keine symbolischen Algorithmen mit $\mathcal{O}(\log^k N)$ logischen Operationen.

- Resultat ist unabhängig von der Funktionsdatenstruktur (OBDDs, etc.).
- Simulation paralleler Algorithmen erscheint schwierig.
- Offen: 0-1 Flussmaximierung $\in NC$?

Ergebnis für P-vollständige Probleme

Falls $P \neq NC$:

Satz

P-vollständige Probleme haben keine parallelen Algo. mit Zeit $\mathcal{O}(\log^k N)$ auf $\mathcal{O}(N^k)$ Prozessoren.

Folgerung

P-vollständige Probleme haben keine symbolischen Algorithmen mit $\mathcal{O}(\log^k N)$ logischen Operationen.

- Resultat ist unabhängig von der Funktionsdatenstruktur (OBDDs, etc.).
- Simulation paralleler Algorithmen erscheint schwierig.
- Offen: 0-1 Flussmaximierung $\in NC$?

Inhalt

- 1 OBDDs und Motivation
- 2 Anzahl der OBDD-Operationen
- 3 Nichtexistenz von Festparameteralgorithmen**
- 4 Konkrete exponentielle untere Schranken
- 5 Fazit und Ausblick

Masterproblem s - t -Zusammenhang

- Eingabe: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hart!
 - Technik: Erzeuge kleines OBDD für Konfigurationsgraph einer pol. platzbeschr. TM.
 - Für $\Pi \in PSPACE$, TM M_Π und Eingabe $I \in \{0, 1\}^m$: Erzeuge OBDD $\chi_{\Pi, I}$ der Größe $\mathcal{O}(\text{poly}(m))$.
 - Startkonf. und akzeptierende Konf. verbunden?
- Bzgl. Graphgröße: Kein $\mathcal{O}(\log^k |V|)$ -Algorithmus.
- Frage: Welche Einschränkungen ermöglichen pol. Algorithmen?

Masterproblem s - t -Zusammenhang

- Eingabe: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hart!
 - Technik: Erzeuge kleines OBDD für Konfigurationsgraph einer pol. platzbeschr. TM.
 - Für $\Pi \in PSPACE$, TM M_Π und Eingabe $I \in \{0, 1\}^m$: Erzeuge OBDD $\chi_{\Pi, I}$ der Größe $\mathcal{O}(\text{poly}(m))$.
 - Startkonf. und akzeptierende Konf. verbunden?
- Bzgl. Graphgröße: Kein $\mathcal{O}(\log^k |V|)$ -Algorithmus.
- Frage: Welche Einschränkungen ermöglichen pol. Algorithmen?

Masterproblem s - t -Zusammenhang

- Eingabe: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hart!
 - Technik: Erzeuge kleines OBDD für Konfigurationsgraph einer pol. platzbeschr. TM.
 - Für $\Pi \in PSPACE$, TM M_Π und Eingabe $I \in \{0, 1\}^m$: Erzeuge OBDD $\chi_{\Pi, I}$ der Größe $\mathcal{O}(\text{poly}(m))$.
 - Startkonf. und akzeptierende Konf. verbunden?
- Bzgl. Graphgröße: Kein $\mathcal{O}(\log^k |V|)$ -Algorithmus.
- Frage: Welche Einschränkungen ermöglichen pol. Algorithmen?

Masterproblem s - t -Zusammenhang

- Eingabe: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hart!
 - Technik: Erzeuge kleines OBDD für Konfigurationsgraph einer pol. platzbeschr. TM.
 - Für $\Pi \in PSPACE$, TM M_Π und Eingabe $I \in \{0, 1\}^m$: Erzeuge OBDD $\chi_{\Pi, I}$ der Größe $\mathcal{O}(\text{poly}(m))$.
 - Startkonf. und akzeptierende Konf. verbunden?
- Bzgl. Graphgröße: Kein $\mathcal{O}(\log^k |V|)$ -Algorithmus.
- Frage: Welche Einschränkungen ermöglichen pol. Algorithmen?

Masterproblem s - t -Zusammenhang

- Eingabe: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hart!
 - Technik: Erzeuge kleines OBDD für Konfigurationsgraph einer pol. platzbeschr. TM.
 - Für $\Pi \in PSPACE$, TM M_Π und Eingabe $I \in \{0, 1\}^m$: Erzeuge OBDD $\chi_{\Pi, I}$ der Größe $\mathcal{O}(\text{poly}(m))$.
 - Startkonf. und akzeptierende Konf. verbunden?
- Bzgl. Graphgröße: Kein $\mathcal{O}(\log^k |V|)$ -Algorithmus.
- Frage: Welche Einschränkungen ermöglichen pol. Algorithmen?

Masterproblem s - t -Zusammenhang

- Eingabe: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hart!
 - Technik: Erzeuge kleines OBDD für Konfigurationsgraph einer pol. platzbeschr. TM.
 - Für $\Pi \in PSPACE$, TM M_Π und Eingabe $I \in \{0, 1\}^m$: Erzeuge OBDD $\chi_{\Pi, I}$ der Größe $\mathcal{O}(\text{poly}(m))$.
 - Startkonf. und akzeptierende Konf. verbunden?
- Bzgl. Graphgröße: Kein $\mathcal{O}(\log^k |V|)$ -Algorithmus.
- Frage: Welche Einschränkungen ermöglichen pol. Algorithmen?

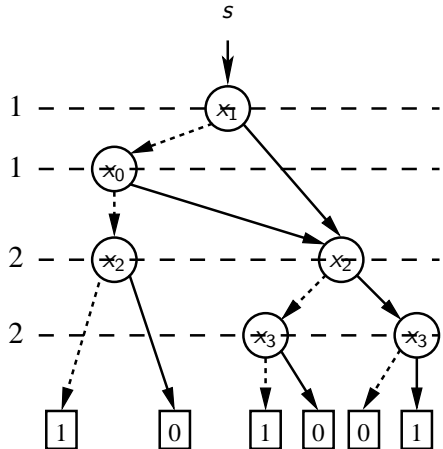
Masterproblem s - t -Zusammenhang

- Eingabe: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hart!
 - Technik: Erzeuge kleines OBDD für Konfigurationsgraph einer pol. platzbeschr. TM.
 - Für $\Pi \in PSPACE$, TM M_Π und Eingabe $I \in \{0, 1\}^m$: Erzeuge OBDD $\chi_{\Pi, I}$ der Größe $\mathcal{O}(\text{poly}(m))$.
 - Startkonf. und akzeptierende Konf. verbunden?
- Bzgl. Graphgröße: Kein $\mathcal{O}(\log^k |V|)$ -Algorithmus.
- Frage: Welche Einschränkungen ermöglichen pol. Algorithmen?

Vollst. OBDD-Breite

Definition

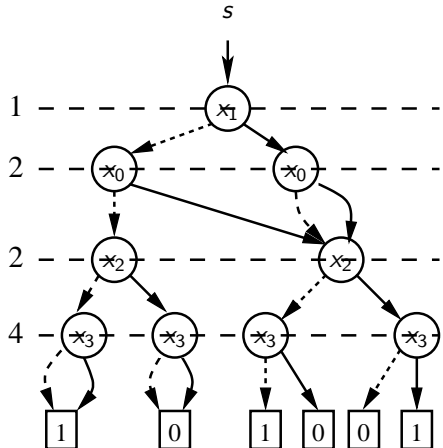
Die **vollst. Breite** eines OBDDs ist die maximale Anzahl Knoten mit gleicher Var.-Markierung im vollst. OBDD.



Vollst. OBDD-Breite

Definition

Die **vollst. Breite** eines OBDDs ist die maximale Anzahl Knoten mit gleicher Var.-Markierung im vollst. OBDD.



Graphfolgen und vollst. OBDD-Breite

- Betrachte **Graphfolge** $(G_n)_n$ und OBDD-Folge $(\chi_{G_n})_n$.
- vollst. Breite $B = \mathcal{O}(1) \Rightarrow$ **Festparameteralgorithmen**

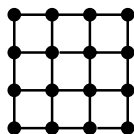
G_1



G_2

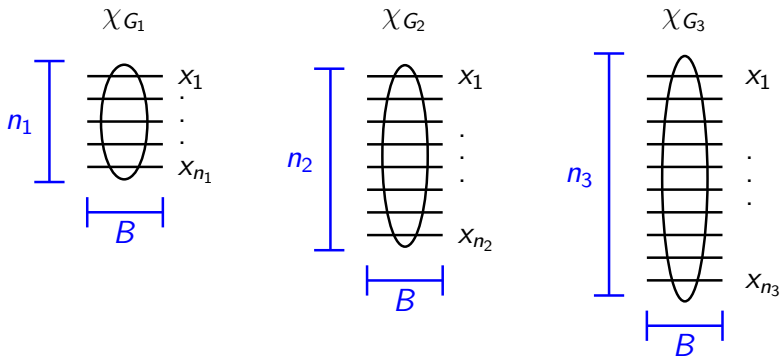


G_3



Graphfolgen und vollst. OBDD-Breite

- Betrachte **Graphfolge** $(G_n)_n$ und OBDD-Folge $(\chi_{G_n})_n$.
- vollst. Breite $B = \mathcal{O}(1) \Rightarrow$ **Festparameteralgorithmen**



Vollst. OBDD-Breite als Festparameter

- Gibt es effiziente Algorithmen für „schmale“ OBDDs?
- Habe χ_G vollst. Breite B , sei α bel. Funktion:
- Parametrisierte Laufzeit $\mathcal{O}(\text{poly}(nB) \cdot \alpha(B))$ möglich?
- Feigenbaum-Beweis: $B = \mathcal{O}(1) \Rightarrow$ Kein FPT-algo. for s - t -Zusammenhang
- Neu: **Fixed-Parameter Intractability** für weitere Probleme.

Satz

$P \neq \text{PSPACE} \Rightarrow \text{MaxFlow, APSP, SSSP, Erreichbarkeit, Trans. Abschluss, MST, Kreisfreiheit, Bipartitheit, Planarität, } \dots \notin \text{FPT}$

Vollst. OBDD-Breite als Festparameter

- Gibt es effiziente Algorithmen für „schmale“ OBDDs?
- Habe χ_G vollst. Breite B , sei α bel. Funktion:
 - Parametrisierte Laufzeit $\mathcal{O}(\text{poly}(nB) \cdot \alpha(B))$ möglich?
 - Feigenbaum-Beweis: $B = \mathcal{O}(1) \Rightarrow$ Kein FPT-algo. for s - t -Zusammenhang
 - Neu: **Fixed-Parameter Intractability** für weitere Probleme.

Satz

$P \neq \text{PSPACE} \Rightarrow \text{MaxFlow, APSP, SSSP, Erreichbarkeit, Trans. Abschluss, MST, Kreisfreiheit, Bipartitheit, Planarität, } \dots \notin \text{FPT}$

Vollst. OBDD-Breite als Festparameter

- Gibt es effiziente Algorithmen für „schmale“ OBDDs?
- Habe χ_G vollst. Breite B , sei α bel. Funktion:
- Parametrisierte Laufzeit $\mathcal{O}(\text{poly}(nB) \cdot \alpha(B))$ möglich?
- Feigenbaum-Beweis: $B = \mathcal{O}(1) \Rightarrow$ Kein FPT-algo. for s - t -Zusammenhang
- Neu: **Fixed-Parameter Intractability** für weitere Probleme.

Satz

$P \neq \text{PSPACE} \Rightarrow \text{MaxFlow, APSP, SSSP, Erreichbarkeit, Trans. Abschluss, MST, Kreisfreiheit, Bipartitheit, Planarität, } \dots \notin \text{FPT}$

Vollst. OBDD-Breite als Festparameter

- Gibt es effiziente Algorithmen für „schmale“ OBDDs?
- Habe χ_G vollst. Breite B , sei α bel. Funktion:
- Parametrisierte Laufzeit $\mathcal{O}(\text{poly}(nB) \cdot \alpha(B))$ möglich?
- Feigenbaum-Beweis: $B = \mathcal{O}(1) \Rightarrow$ Kein FPT-algo. for s - t -Zusammenhang
- Neu: **Fixed-Parameter Intractability** für weitere Probleme.

Satz

$P \neq \text{PSPACE} \Rightarrow \text{MaxFlow, APSP, SSSP, Erreichbarkeit, Trans. Abschluss, MST, Kreisfreiheit, Bipartitheit, Planarität, } \dots \notin \text{FPT}$

Vollst. OBDD-Breite als Festparameter

- Gibt es effiziente Algorithmen für „schmale“ OBDDs?
- Habe χ_G vollst. Breite B , sei α bel. Funktion:
- Parametrisierte Laufzeit $\mathcal{O}(\text{poly}(nB) \cdot \alpha(B))$ möglich?
- Feigenbaum-Beweis: $B = \mathcal{O}(1) \Rightarrow$ Kein FPT-algo. for s - t -Zusammenhang
- Neu: **Fixed-Parameter Intractability** für weitere Probleme.

Satz

$P \neq \text{PSPACE} \Rightarrow \text{MaxFlow, APSP, SSSP, Erreichbarkeit, Trans. Abschluss, MST, Kreisfreiheit, Bipartitheit, Planarität, } \dots \notin \text{FPT}$

Vollst. OBDD-Breite als Festparameter

- Gibt es effiziente Algorithmen für „schmale“ OBDDs?
- Habe χ_G vollst. Breite B , sei α bel. Funktion:
- Parametrisierte Laufzeit $\mathcal{O}(\text{poly}(nB) \cdot \alpha(B))$ möglich?
- Feigenbaum-Beweis: $B = \mathcal{O}(1) \Rightarrow$ Kein FPT-algo. for s - t -Zusammenhang
- Neu: **Fixed-Parameter Intractability** für weitere Probleme.

Satz

$P \neq \text{PSPACE} \Rightarrow \text{MaxFlow, APSP, SSSP, Erreichbarkeit, Trans. Abschluss, MST, Kreisfreiheit, Bipartitheit, Planarität, } \dots \notin \text{FPT}$

Breitenerhaltende Reduktionen von Π auf Π'

Abbildung ϕ ist **breitenerhaltende Reduktion** von Π auf Π' wenn

- sie χ_G auf $\chi_{G'}$ abbildet, so dass

$$G \in \Pi \Leftrightarrow G' \in \Pi',$$

- vollst. Breite B' von $\chi_{G'}$ nur von B abhängt (und nicht von $|V|$).

Satz

Eine Folge von $\mathcal{O}(1)$ OBDD-Operationen und Quantorenfolgen ist breitenerhaltend.

Breitenerhaltende Reduktionen von Π auf Π'

Abbildung ϕ ist **breitenerhaltende Reduktion** von Π auf Π' wenn

- sie χ_G auf $\chi_{G'}$ abbildet, so dass

$$G \in \Pi \Leftrightarrow G' \in \Pi',$$

- vollst. Breite B' von $\chi_{G'}$ nur von B abhängt (und nicht von $|V|$).

Satz

Eine Folge von $\mathcal{O}(1)$ OBDD-Operationen und Quantorenfolgen ist breitenerhaltend.

Breitenerhaltende Reduktionen von Π auf Π'

Abbildung ϕ ist **breitenerhaltende Reduktion** von Π auf Π' wenn

- sie χ_G auf $\chi_{G'}$ abbildet, so dass

$$G \in \Pi \Leftrightarrow G' \in \Pi',$$

- vollst. Breite B' von $\chi_{G'}$ nur von B abhängt (und nicht von $|V|$).

Satz

Eine Folge von $\mathcal{O}(1)$ OBDD-Operationen und Quantorenfolgen ist breitenerhaltend.

Breitenerhaltende Reduktionen von Π auf Π'

Abbildung ϕ ist **breitenerhaltende Reduktion** von Π auf Π' wenn

- sie χ_G auf $\chi_{G'}$ abbildet, so dass

$$G \in \Pi \Leftrightarrow G' \in \Pi',$$

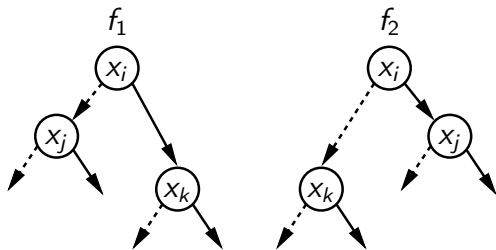
- vollst. Breite B' von $\chi_{G'}$ nur von B abhängt (und nicht von $|V|$).

Satz

Eine Folge von $\mathcal{O}(1)$ OBDD-Operationen und Quantorenfolgen ist breitenerhaltend.

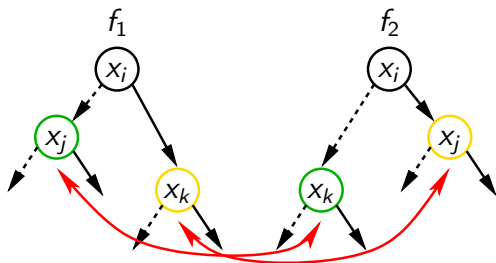
Vollst. OBDD-Breite und Binäre Synthese

- Allgemein: $|f_1 \otimes f_2| = \Omega(|f_1| \cdot |f_2|)$
- Vollst. Breite B : $|f_1 \otimes f_2| = \mathcal{O}(n \cdot B^2)$.



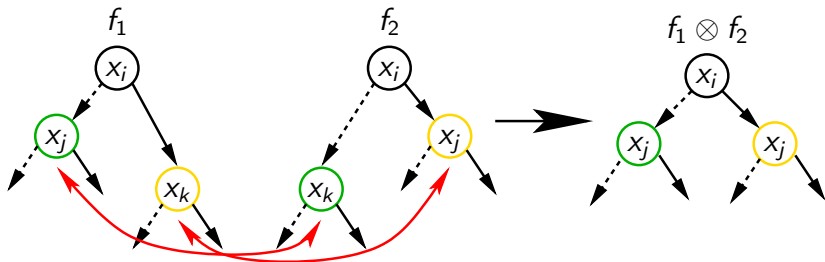
Vollst. OBDD-Breite und Binäre Synthese

- Allgemein: $|f_1 \otimes f_2| = \Omega(|f_1| \cdot |f_2|)$
- Vollst. Breite B .: $|f_1 \otimes f_2| = \mathcal{O}(n \cdot B^2)$.



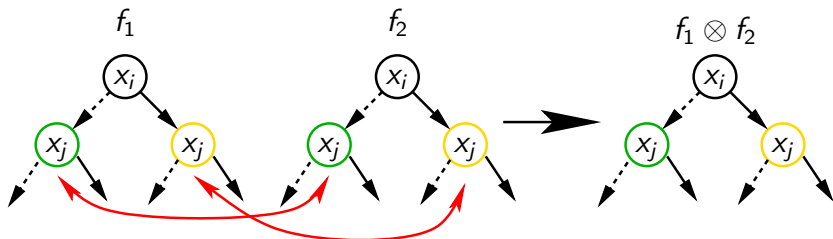
Vollst. OBDD-Breite und Binäre Synthese

- Allgemein: $|f_1 \otimes f_2| = \Omega(|f_1| \cdot |f_2|)$
- Vollst. Breite B .: $|f_1 \otimes f_2| = \mathcal{O}(n \cdot B^2)$.



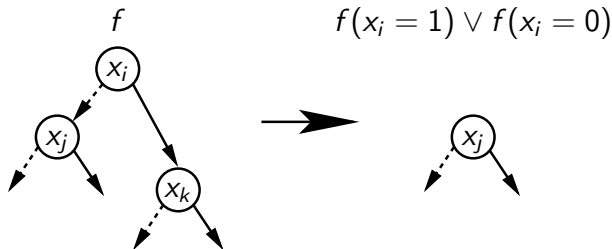
Vollst. OBDD-Breite und Binäre Synthese

- Allgemein: $|f_1 \otimes f_2| = \Omega(|f_1| \cdot |f_2|)$
- Vollst. Breite B .: $|f_1 \otimes f_2| = \mathcal{O}(n \cdot B^2)$.



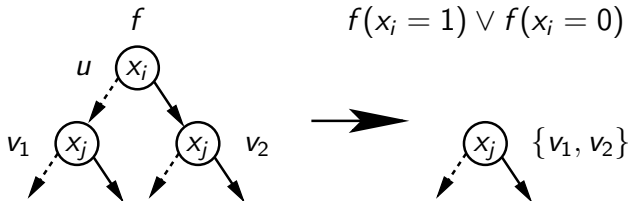
Vollst. OBDD-Breite und Quantifizierung

- Allgemeiner Fall: $|(\exists x_{i_1}, \dots, x_{i_r}) f| = \Omega(|f|^{2^r})$
- Vollst. Breite B : $|(\exists x_{i_1}, \dots, x_{i_r}) f| = \mathcal{O}(n \cdot 2^B)$



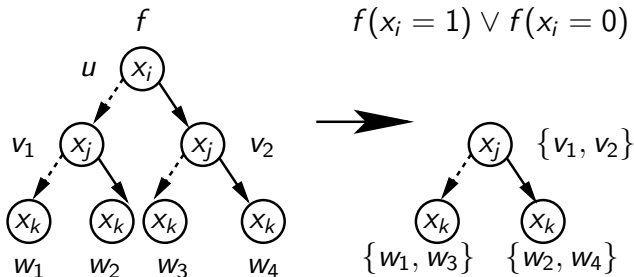
Vollst. OBDD-Breite und Quantifizierung

- Allgemeiner Fall: $|(\exists x_{i_1}, \dots, x_{i_r}) f| = \Omega(|f|^{2^r})$
- Vollst. Breite B : $|(\exists x_{i_1}, \dots, x_{i_r}) f| = \mathcal{O}(n \cdot 2^B)$



Vollst. OBDD-Breite und Quantifizierung

- Allgemeiner Fall: $|(\exists x_{i_1}, \dots, x_{i_r}) f| = \Omega(|f|^{2^r})$
- Vollst. Breite B : $|(\exists x_{i_1}, \dots, x_{i_r}) f| = \mathcal{O}(n \cdot 2^B)$



Threshold-Funktionen

Weiterer Baustein: **Threshold-Funktionen**

$$f(x^{(1)}, \dots, x^{(k)}) = \left(\sum_{i=1}^k \delta_i \cdot |x^{(i)}| \geq T \right)$$

Für $\Delta := \max\{\delta_i\}$:

Satz

Threshold-Funktionen haben vollständige OBDD-Breite $\mathcal{O}(k^2 \Delta)$.

- Realisieren $=, \neq, <, \leq, >$

Threshold-Funktionen

Weiterer Baustein: **Threshold-Funktionen**

$$f(x^{(1)}, \dots, x^{(k)}) = \left(\sum_{i=1}^k \delta_i \cdot |x^{(i)}| \geq T \right)$$

Für $\Delta := \max\{\delta_i\}$:

Satz

Threshold-Funktionen haben vollständige OBDD-Breite $\mathcal{O}(k^2 \Delta)$.

- Realisieren $=, \neq, <, \leq, >$

Threshold-Funktionen

Weiterer Baustein: **Threshold-Funktionen**

$$f(x^{(1)}, \dots, x^{(k)}) = \left(\sum_{i=1}^k \delta_i \cdot |x^{(i)}| \geq T \right)$$

Für $\Delta := \max\{\delta_i\}$:

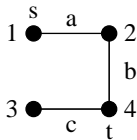
Satz

Threshold-Funktionen haben vollständige OBDD-Breite $\mathcal{O}(k^2\Delta)$.

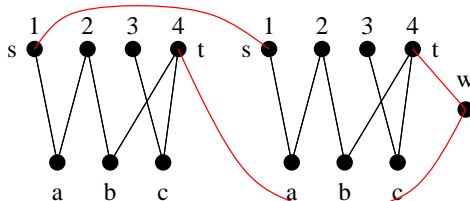
- Realisieren $=, \neq, <, \leq, >$

Beispiel 1: Bipartitheit

Reduktion von **ungericht. s - t -Zusammenhang** auf **Bipartitheit**:



G



Kopie 1

G'

Kopie 2

- Ausdruck konstanter Länge \Rightarrow breiterehaltend
- Konstante Breite von $\chi_G \Rightarrow$ konstante Breite von $\chi_{G'}$.
- \exists FPT-Algo. für **Bipartitheit** $\Rightarrow \exists$ FPT-Algo für s - t -Zusammenhang ζ

Beispiel 1: Bipartitheit

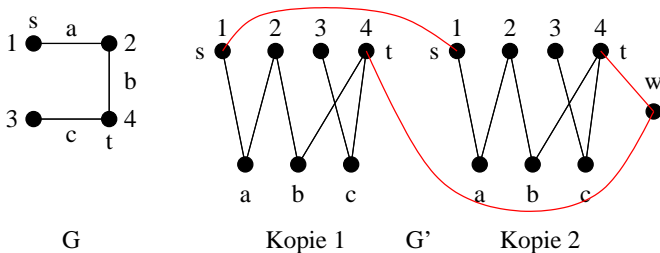
Reduktion von **ungericht. s - t -Zusammenhang** auf **Bipartitheit**:

$$\begin{aligned} \chi_{G'}(x, y) := & [(T(x) = v) \wedge (T(y) = e) \wedge (i(x) = i(y)) \wedge (c(x) = c(y)) \wedge \chi_G(i(y), j(y))] \\ & \vee [(T(x) = e) \wedge (T(y) = v) \wedge (j(x) = j(y)) \wedge (c(x) = c(y)) \wedge \chi_G(i(x), j(x))] \\ & \vee [(T(x) = T(y) = v) \wedge (v_{|i(x)|} = v_{|i(y)|} = s) \wedge (c(x) \neq c(y))] \\ & \vee [(T(x) = v) \wedge (T(y) = w) \wedge (v_{|i(x)|} = t)] , \end{aligned}$$

- Ausdruck konstanter Länge \Rightarrow breitenerhaltend
- Konstante Breite von $\chi_G \Rightarrow$ konstante Breite von $\chi_{G'}$.
- \exists FPT-Algo. für **Bipartitheit** $\Rightarrow \exists$ FPT-Algo für s - t -Zusammenhang \Downarrow

Beispiel 1: Bipartitheit

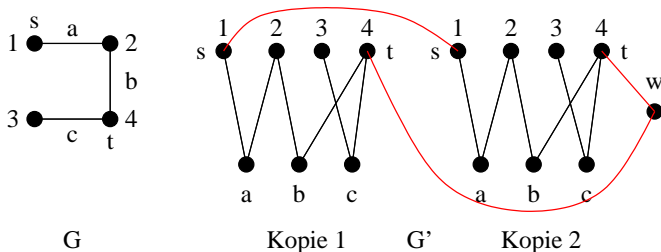
Reduktion von **ungericht. s - t -Zusammenhang** auf **Bipartitheit**:



- Ausdruck konstanter Länge \Rightarrow breitenerhaltend
- Konstante Breite von $\chi_G \Rightarrow$ konstante Breite von $\chi_{G'}$.
- \exists FPT-Algo. für **Bipartitheit** $\Rightarrow \exists$ FPT-Algo für s - t -Zusammenhang \nleftrightarrow

Beispiel 1: Bipartitheit

Reduktion von **ungericht. s - t -Zusammenhang** auf **Bipartitheit**:



- Ausdruck konstanter Länge \Rightarrow breitenerhaltend
- Konstante Breite von $\chi_G \Rightarrow$ konstante Breite von $\chi_{G'}$.
- \exists FPT-Algo. für **Bipartitheit** $\Rightarrow \exists$ FPT-Algo für s - t -Zusammenhang ζ

Beispiel 2: Minimale Spannbäume

Technik: Modifiziere Reduktion

$$L \leq_p s\text{-}t\text{-Zusammenhang auf OBDD } \chi_G$$

für $L \in PSPACE$ (Feigenbaum et al., STACS'98).

- Zustandsgraph $G_{\mathcal{M}}$ einer TM \mathcal{M} hat konstante OBDD-Breite
- \mathcal{M} pol. platzbeschränkt \Rightarrow polynomielle Konf.-kodierung
- **MST**: Eingabe $\chi_G(x, y, a)$, Ausgabe $MST(x, y, a)$

Beispiel 2: Minimale Spannbäume

Technik: Modifiziere Reduktion

$$L \leq_p s\text{-}t\text{-Zusammenhang auf OBDD } \chi_G$$

für $L \in PSPACE$ (Feigenbaum et al., STACS'98).

- Zustandsgraph $G_{\mathcal{M}}$ einer TM \mathcal{M} hat konstante OBDD-Breite
- \mathcal{M} pol. platzbeschränkt \Rightarrow polynomielle Konf.-kodierung
- **MST**: Eingabe $\chi_G(x, y, a)$, Ausgabe $MST(x, y, a)$

Beispiel 2: Minimale Spannbäume

Technik: Modifiziere Reduktion

$$L \leq_p s\text{-}t\text{-Zusammenhang auf OBDD } \chi_G$$

für $L \in PSPACE$ (Feigenbaum et al., STACS'98).

- Zustandsgraph $G_{\mathcal{M}}$ einer TM \mathcal{M} hat konstante OBDD-Breite
- \mathcal{M} pol. platzbeschränkt \Rightarrow polynomielle Konf.-kodierung
- **MST**: Eingabe $\chi_G(x, y, a)$, Ausgabe $MST(x, y, a)$

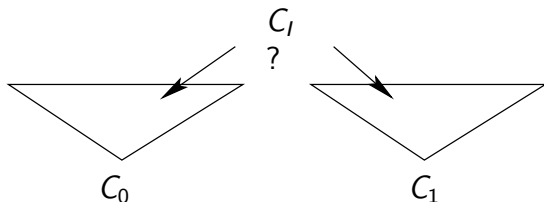
Beispiel 2: Minimale Spannbäume

Technik: Modifiziere Reduktion

$L \leq_p$ s - t -Zusammenhang auf OBDD χ_G

für $L \in PSPACE$ (Feigenbaum et al., STACS'98).

- Zustandsgraph $G_{\mathcal{M}}$ einer TM \mathcal{M} hat konstante OBDD-Breite
- \mathcal{M} pol. platzbeschränkt \Rightarrow polynomielle Konf.-kodierung
- **MST**: Eingabe $\chi_G(x, y, a)$, Ausgabe $MST(x, y, a)$



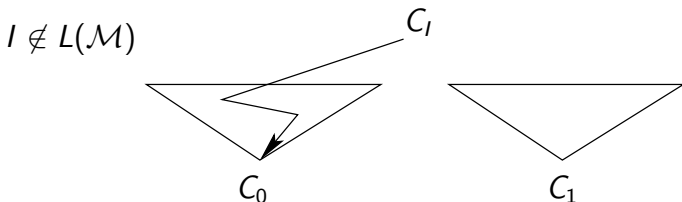
Beispiel 2: Minimale Spannbäume

Technik: Modifiziere Reduktion

$L \leq_p$ s - t -Zusammenhang auf OBDD χ_G

für $L \in PSPACE$ (Feigenbaum et al., STACS'98).

- Zustandsgraph $G_{\mathcal{M}}$ einer TM \mathcal{M} hat konstante OBDD-Breite
- \mathcal{M} pol. platzbeschränkt \Rightarrow polynomielle Konf.-kodierung
- **MST**: Eingabe $\chi_G(x, y, a)$, Ausgabe $MST(x, y, a)$



Beispiel 2: Minimale Spannbäume

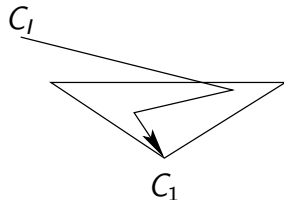
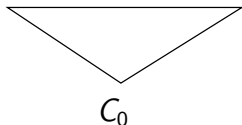
Technik: Modifiziere Reduktion

$$L \leq_p s-t\text{-Zusammenhang auf OBDD } \chi_G$$

für $L \in PSPACE$ (Feigenbaum et al., STACS'98).

- Zustandsgraph $G_{\mathcal{M}}$ einer TM \mathcal{M} hat konstante OBDD-Breite
- \mathcal{M} pol. platzbeschränkt \Rightarrow polynomielle Konf.-kodierung
- **MST**: Eingabe $\chi_G(x, y, a)$, Ausgabe $\text{MST}(x, y, a)$

$$I \in L(\mathcal{M})$$



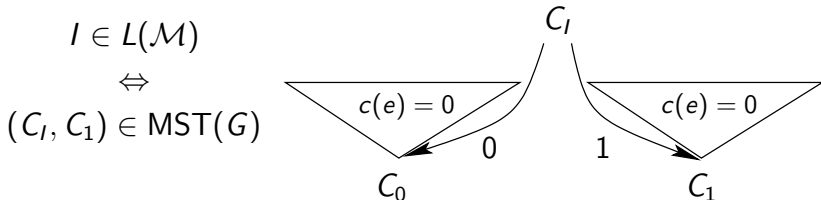
Beispiel 2: Minimale Spannbäume

Technik: Modifiziere Reduktion

$L \leq_p s-t$ -Zusammenhang auf OBDD χ_G

für $L \in PSPACE$ (Feigenbaum et al., STACS'98).

- Zustandsgraph $G_{\mathcal{M}}$ einer TM \mathcal{M} hat konstante OBDD-Breite
- \mathcal{M} pol. platzbeschränkt \Rightarrow polynomielle Konf.-kodierung
- **MST**: Eingabe $\chi_G(x, y, a)$, Ausgabe $\text{MST}(x, y, a)$



Inhalt

- 1 OBDDs und Motivation
- 2 Anzahl der OBDD-Operationen
- 3 Nichtexistenz von Festparameteralgorithmen
- 4 Konkrete exponentielle untere Schranken**
- 5 Fazit und Ausblick

Problem und Ansatz

- Bisher: Probleme auf OBDDs PSPACE-hart
- Gewünscht: Unbedingte exp. untere Schranken für Ergebnis-OBDDs (also Zeit und Platz)
- 1) Eingabe-OBDD konstant breit
 - Mit Bausteinfunkt. und FPT-Operationen
- 2) Ausgabe-OBDDs exp. groß
 - ? – bisher: Multiplikation, ISA, HWB
- Betrachte Funktionen $F_n = (f_{n,1}, \dots, f_{n,m})$
- Idee: Übertrage bekannte Schranken für $f_{n,i}$ auf F_n -GRAPH mit

$$F_n\text{-GRAPH}(x, y) = \bigwedge_i f_i(x) = y_i.$$

Problem und Ansatz

- Bisher: Probleme auf OBDDs PSPACE-hart
- Gewünscht: Unbedingte exp. untere Schranken für Ergebnis-OBDDs (also Zeit und Platz)
- 1) Eingabe-OBDD konstant breit
 - Mit Bausteinfunkt. und FPT-Operationen
- 2) Ausgabe-OBDDs exp. groß
 - ? – bisher: Multiplikation, ISA, HWB
- Betrachte Funktionen $F_n = (f_{n,1}, \dots, f_{n,m})$
- Idee: Übertrage bekannte Schranken für $f_{n,i}$ auf F_n -GRAPH mit

$$F_n\text{-GRAPH}(x, y) = \bigwedge_i f_i(x) = y_i.$$

Problem und Ansatz

- Bisher: Probleme auf OBDDs PSPACE-hart
- Gewünscht: Unbedingte exp. untere Schranken für Ergebnis-OBDDs (also Zeit und Platz)
- 1) Eingabe-OBDD konstant breit
 - Mit Bausteinfunkt. und FPT-Operationen
- 2) Ausgabe-OBDDs exp. groß
 - ? – bisher: Multiplikation, ISA, HWB
- Betrachte Funktionen $F_n = (f_{n,1}, \dots, f_{n,m})$
- Idee: Übertrage bekannte Schranken für $f_{n,i}$ auf F_n -GRAPH mit

$$F_n\text{-GRAPH}(x, y) = \bigwedge_i f_i(x) = y_i.$$

Problem und Ansatz

- Bisher: Probleme auf OBDDs PSPACE-hart
- Gewünscht: Unbedingte exp. untere Schranken für Ergebnis-OBDDs (also Zeit und Platz)
- 1) Eingabe-OBDD konstant breit
 - Mit Bausteinfunkt. und FPT-Operationen
- 2) Ausgabe-OBDDs exp. groß
 - ? – bisher: Multiplikation, ISA, HWB
- Betrachte Funktionen $F_n = (f_{n,1}, \dots, f_{n,m})$
- Idee: Übertrage bekannte Schranken für $f_{n,i}$ auf F_n -GRAPH mit

$$F_n\text{-GRAPH}(x, y) = \bigwedge_i f_i(x) = y_i.$$

Problem und Ansatz

- Bisher: Probleme auf OBDDs PSPACE-hart
- Gewünscht: Unbedingte exp. untere Schranken für Ergebnis-OBDDs (also Zeit und Platz)
- 1) Eingabe-OBDD konstant breit
 - Mit Bausteinfunkt. und FPT-Operationen
- 2) Ausgabe-OBDDs exp. groß
 - ? – bisher: Multiplikation, ISA, HWB
- Betrachte Funktionen $F_n = (f_{n,1}, \dots, f_{n,m})$
- Idee: Übertrage bekannte Schranken für $f_{n,i}$ auf F_n -GRAPH mit

$$F_n\text{-GRAPH}(x, y) = \bigwedge_i f_i(x) = y_i.$$

Problem und Ansatz

- Bisher: Probleme auf OBDDs PSPACE-hart
- Gewünscht: Unbedingte exp. untere Schranken für Ergebnis-OBDDs (also Zeit und Platz)
- 1) Eingabe-OBDD konstant breit
 - Mit Bausteinfunkt. und FPT-Operationen
- 2) Ausgabe-OBDDs exp. groß
 - ? – bisher: Multiplikation, ISA, HWB
- Betrachte Funktionen $F_n = (f_{n,1}, \dots, f_{n,m})$
- Idee: Übertrage bekannte Schranken für $f_{n,i}$ auf F_n -GRAPH mit

$$F_n\text{-GRAPH}(x, y) = \bigwedge_i f_i(x) = y_i.$$

Problem und Ansatz

- Bisher: Probleme auf OBDDs PSPACE-hart
- Gewünscht: Unbedingte exp. untere Schranken für Ergebnis-OBDDs (also Zeit und Platz)
- 1) Eingabe-OBDD konstant breit
 - Mit Bausteinfunkt. und FPT-Operationen
- 2) Ausgabe-OBDDs exp. groß
 - ? – bisher: Multiplikation, ISA, HWB
- Betrachte Funktionen $F_n = (f_{n,1}, \dots, f_{n,m})$
- Idee: Übertrage bekannte Schranken für $f_{n,i}$ auf F_n -GRAPH mit

$$F_n\text{-GRAPH}(x, y) = \bigwedge_i f_i(x) = y_i.$$

Problem und Ansatz

- Bisher: Probleme auf OBDDs PSPACE-hart
- Gewünscht: Unbedingte exp. untere Schranken für Ergebnis-OBDDs (also Zeit und Platz)
- 1) Eingabe-OBDD konstant breit
 - Mit Bausteinfunkt. und FPT-Operationen
- 2) Ausgabe-OBDDs exp. groß
 - ? – bisher: Multiplikation, ISA, HWB
- Betrachte Funktionen $F_n = (f_{n,1}, \dots, f_{n,m})$
- Idee: Übertrage bekannte Schranken für $f_{n,i}$ auf F_n -GRAPH mit

$$F_n\text{-GRAPH}(x, y) = \bigwedge_i f_i(x) = y_i.$$

Resultate

- Untere Schranken für $f_{n,i}$ gelten nicht unbedingt für F_n -GRAPH.
- Übertragung gelingt für Multiplikation, ISA und HWB (S., SOFSEM'05).
- Konkret: $MULT_n\text{-GRAPH}(x, y, z) = \bigwedge_i [(|x| \cdot |y|)_i = z_i]$

Satz

MaxFlow, SSSP, Erreichbarkeit haben exp. Platzkomplexität auf OBDD-Eingaben konstanter Breite.

Resultate

- Untere Schranken für $f_{n,i}$ gelten nicht unbedingt für F_n -GRAPH.
- Übertragung gelingt für Multiplikation, ISA und HWB (S., SOFSEM'05).
- Konkret: $MULT_n\text{-GRAPH}(x, y, z) = \bigwedge_i [(|x| \cdot |y|)_i = z_i]$

Satz

MaxFlow, SSSP, Erreichbarkeit haben exp. Platzkomplexität auf OBDD-Eingaben konstanter Breite.

Resultate

- Untere Schranken für $f_{n,i}$ gelten nicht unbedingt für F_n -GRAPH.
- Übertragung gelingt für Multiplikation, ISA und HWB (S., SOFSEM'05).
- Konkret: $MULT_n\text{-GRAPH}(x, y, z) = \bigwedge_i [(|x| \cdot |y|)_i = z_i]$

Satz

MaxFlow, SSSP, Erreichbarkeit haben exp. Platzkomplexität auf OBDD-Eingaben konstanter Breite.

Resultate

- Untere Schranken für $f_{n,i}$ gelten nicht unbedingt für F_n -GRAPH.
- Übertragung gelingt für Multiplikation, ISA und HWB (S., SOFSEM'05).
- Konkret: $MULT_n\text{-GRAPH}(x, y, z) = \bigwedge_i [(|x| \cdot |y|)_i = z_i]$

Satz

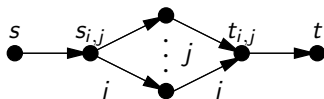
MaxFlow, SSSP, Erreichbarkeit haben exp. Platzkomplexität auf OBDD-Eingaben konstanter Breite.

Resultate

- Untere Schranken für $f_{n,i}$ gelten nicht unbedingt für F_n -GRAPH.
- Übertragung gelingt für Multiplikation, ISA und HWB (S., SOFSEM'05).
- Konkret: $MULT_n\text{-GRAPH}(x, y, z) = \bigwedge_i [(|x| \cdot |y|)_i = z_i]$

Satz

MaxFlow, SSSP, Erreichbarkeit haben exp. Platzkomplexität auf OBDD-Eingaben konstanter Breite.



Inhalt

- 1 OBDDs und Motivation
- 2 Anzahl der OBDD-Operationen
- 3 Nichtexistenz von Festparameteralgorithmen
- 4 Konkrete exponentielle untere Schranken
- 5 Fazit und Ausblick**

Ergebnisübersicht

	#Ops.	I-FPT-T	IO-FPT-T	I-FPT-S	IO-FPT-S
MaxFlow					
0-1 MaxFlow					
APSP					
APSP > 0					
SSSP					
SSSP > 0					
Reach					
TransClos					
MST					
<i>s-t</i> -Conn.					
Bip.					
Acycl.					
Conn.					
SConn.					
BiConn.					
Euler C.					
Plan.					

Ergebnisübersicht

	#Ops.	I-FPT-T	IO-FPT-T	I-FPT-S	IO-FPT-S
MaxFlow	×				
0-1 MaxFlow	?				
APSP	✓				
APSP > 0	✓				
SSSP	✓				
SSSP > 0	✓				
Reach	✓				
TransClos	✓				
MST	✓				
<i>s-t</i> -Conn.	✓				
Bip.	✓				
Acycl.	✓				
Conn.	✓				
SConn.	✓				
BiConn.	✓				
Euler C.	✓				
Plan.	?				

Ergebnisübersicht

	#Ops.	I-FPT-T	IO-FPT-T	I-FPT-S	IO-FPT-S
MaxFlow	×	×			
0-1 MaxFlow	?	×			
APSP	✓	×			
APSP > 0	✓	×			
SSSP	✓	×			
SSSP > 0	✓	×			
Reach	✓	×			
TransClos	✓	×			
MST	✓	×			
<i>s-t</i> -Conn.	✓	×			
Bip.	✓	×			
Acycl.	✓	×			
Conn.	✓	×			
SConn.	✓	×			
BiConn.	✓	×			
Euler C.	✓	×			
Plan.	?	×			

Ergebnisübersicht

	#Ops.	I-FPT-T	IO-FPT-T	I-FPT-S	IO-FPT-S
MaxFlow	×	×	×		
0-1 MaxFlow	?	×	×		
APSP	✓	×	×		
APSP > 0	✓	×	✓		
SSSP	✓	×	×		
SSSP > 0	✓	×	?		
Reach	✓	×	×		
TransClos	✓	×	×		
MST	✓	×	×		
<i>s-t</i> -Conn.	✓	×	-		
Bip.	✓	×	-		
Acycl.	✓	×	-		
Conn.	✓	×	-		
SConn.	✓	×	-		
BiConn.	✓	×	-		
Euler C.	✓	×	-		
Plan.	?	×	-		

Ergebnisübersicht

	#Ops.	I-FPT-T	IO-FPT-T	I-FPT-S	IO-FPT-S
MaxFlow	×	×	×	×	
0-1 MaxFlow	?	×	×	?	
APSP	✓	×	×	×	
APSP > 0	✓	×	✓	×	
SSSP	✓	×	×	×	
SSSP > 0	✓	×	?	×	
Reach	✓	×	×	×	
TransClos	✓	×	×	×	
MST	✓	×	×	?	
<i>s-t</i> -Conn.	✓	×	-	✓	
Bip.	✓	×	-	✓	
Acycl.	✓	×	-	✓	
Conn.	✓	×	-	✓	
SConn.	✓	×	-	✓	
BiConn.	✓	×	-	✓	
Euler C.	✓	×	-	✓	
Plan.	?	×	-	?	

Ergebnisübersicht

	#Ops.	I-FPT-T	IO-FPT-T	I-FPT-S	IO-FPT-S
MaxFlow	×	×	×	×	✓
0-1 MaxFlow	?	×	×	?	✓
APSP	✓	×	×	×	✓
APSP > 0	✓	×	✓	×	✓
SSSP	✓	×	×	×	✓
SSSP > 0	✓	×	?	×	✓
Reach	✓	×	×	×	✓
TransClos	✓	×	×	×	✓
MST	✓	×	×	?	✓
<i>s-t</i> -Conn.	✓	×	-	✓	-
Bip.	✓	×	-	✓	-
Acycl.	✓	×	-	✓	-
Conn.	✓	×	-	✓	-
SConn.	✓	×	-	✓	-
BiConn.	✓	×	-	✓	-
Euler C.	✓	×	-	✓	-
Plan.	?	×	-	?	-

Fazit

- **Stark veränderte algorithmische Situation**
- Beziehungen zu parallelen und FPT Algorithmen
- Viele Probleme sind (trotz Breitereinschränkung) schwierig
- \Rightarrow Aspekt “OBDD-Breite” als erster Schritt
- Praktischer Einfluss der Breite/sonstiger Eingabeeigenschaften?

Fazit

- Stark veränderte algorithmische Situation
- Beziehungen zu parallelen und FPT Algorithmen
- Viele Probleme sind (trotz Breitereinschränkung) schwierig
- \Rightarrow Aspekt “OBDD-Breite” als erster Schritt
- Praktischer Einfluss der Breite/sonstiger Eingabeeigenschaften?

Fazit

- Stark veränderte algorithmische Situation
- Beziehungen zu parallelen und FPT Algorithmen
- Viele Probleme sind (trotz Breitereinschränkung) schwierig
- ⇒ Aspekt “OBDD-Breite” als erster Schritt
- Praktischer Einfluss der Breite/sonstiger Eingabeeigenschaften?

Fazit

- Stark veränderte algorithmische Situation
- Beziehungen zu parallelen und FPT Algorithmen
- Viele Probleme sind (trotz Breiteneinschränkung) schwierig
- \Rightarrow Aspekt “OBDD-Breite” als erster Schritt
- Praktischer Einfluss der Breite/sonstiger Eingabeeigenschaften?

Fazit

- Stark veränderte algorithmische Situation
- Beziehungen zu parallelen und FPT Algorithmen
- Viele Probleme sind (trotz Breitereinschränkung) schwierig
- \Rightarrow Aspekt “OBDD-Breite” als erster Schritt
- Praktischer Einfluss der Breite/sonstiger Eingabeeigenschaften?

“That’s all Folks!”

Literatur: <http://ls2-www.cs.uni-dortmund.de/spp1126/>