Parallelität

Parallelisierbarkeit

- Welche Probleme in P sind parallelisierbar?

  Gibt es vom Standpunkt der Parallelisierbarkeit schwierigste Probleme in P *?*
  Gibt es einen entsprechenden Reduktionsbegriff?

- Was soll „parallelisierbar“ überhaupt bedeuten?

  Wenn es eine rasant schnelle(!?) Berechnung bei vernünftiger Größe(!?) gibt.

Zur Klärung der Begrifflichkeiten verwenden wir Schaltkreise.

Schaltkreise

Schaltkreise

Ein Schaltkreis S wird durch den Vektor

$$S = (G, Q, R, \text{gatter}, n, \text{eingabe})$$

beschrieben:

- $G = (V, E)$ ist ein gerichteter, azyklischer Graph.
- Q ist die Menge der Quellen (Knoten mit nur ausgehenden Kanten) und R die Menge der Senken (Knoten mit nur eingehenden Kanten) von G.
- Die Funktion eingabe: $Q \to \{1, \ldots, n\}$ weist jeder Quelle die Position des entsprechenden Eingabebits zu. Die Funktion gatter: $V \setminus Q \to \{\neg, \vee, \wedge\}$ weist jedem inneren Knoten von G eine Gatter-Funktion zu.

S berechnet die Funktion $f_S : \{0,1\}^n \to \{0,1\}^{|R|}$,
indem die n Eingabebits an die Quellen in G angelegt werden, und jeder Knoten das Ergebnis seiner Gatterfunktion weiterleitet.
Das Resultat wird an den Senken von G abgelesen.

Tiefe und Größe von Schaltkreisen

Sei $S = (G, Q, R, \text{gatter}, n, \text{eingabe})$ ein Schaltkreis.

1. Die **Tiefe** von S ist die Länge des längsten Weges in G.
2. Die **Größe** von S ist die Anzahl der Knoten von G, wobei Quellen nicht mitgezählt werden.
3. Der **Fanin** von S ist das Maximum, über alle Knoten v, der Anzahl eingehender Kanten von v.

Welche Größe und welche Tiefe ist für die Berechnung der XOR-Funktion

$$x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

asymptotisch hinreichend und notwendig?

Schaltkreis-Familien

(a) Eine Schaltkreisfamilie ist eine Folge

$$\mathcal{S} = (S_n)_{n \in \mathbb{N}}$$

von Schaltkreisen, so dass S_n eine Boolesche Funktion auf genau n Eingaben berechnet. $\mathcal{S}$ berechnet die Funktion

$$f : \{0,1\}^* \to \{0,1\},$$

wenn S_n die Funktion f eingeschränkt auf $\{0,1\}^n$ berechnet.

(b) Eine Schaltkreisfamilie $(S_n)_{n \in \mathbb{N}}$ ist **uniform**, wenn es eine $\log_2(\text{Größe}\,(S_n) + n)$-platzbeschränkte, deterministische Turingmaschine gibt, die für Eingabe 1^n

- alle Knoten von S_n aufzählt,
- jedem inneren Knoten eine Gatter-Funktion aus $\{\neg, \vee, \wedge\}$, zuweist,
- sämtliche Kanten von S_n aufzählt und
- jeder Quelle von S_n eine Bitposition zuweist.

Uniformität

(Nicht-uniforme) Schaltkreisfamilien

$$\mathcal{S} = (S_n \mid n \in \mathbb{N})$$

können „unanständig mächtig“ sein.

- Das spezielle Halteproblem

$$H_\varepsilon = \{1^{\langle M \rangle} \mid M \text{ hält auf dem leeren Wort } \}$$

  ist unentscheidbar.

- S_n akzeptiert $w \in \{0,1\}^n$ genau dann, wenn $w = 1^n$ und $n = \langle M \rangle$ für eine Turingmaschine M, die auf dem leeren Wort hält.

In einer uniformen Schaltkreisfamilie muss die **Konstruktion** der Schaltkreise **so einfach wie möglich sein!**

Die Tiefe und Größe uniformer Schaltkreise (1/2)

Für Funktionen $d, s : \mathbb{N} \to \mathbb{N}$ definiere

$$\mathsf{DEPTH}_{\text{uniform}}(d)$$

als die Klasse aller Sprachen L, die durch eine uniforme Schaltkreisfamilie in Tiefe $O(d(n))$ und mit Fanin zwei berechnet werden.

$$\mathsf{SIZE}_{\text{uniform}}(s)$$

ist die Klasse aller Sprachen L, die durch eine uniforme Schaltkreisfamilie in Größe $O(s(n))$ und mit Fanin zwei berechnet werden. Schließlich besteht

$$\mathsf{DEPTH{-}SIZE}_{\text{uniform}}(d, s) = \mathsf{DEPTH}_{\text{uniform}}(d) \cap \mathsf{SIZE}_{\text{uniform}}(s)$$

aus allen Sprachen, die sowohl in Tiefe $O(d)$ wie auch in Größe $O(s)$ durch eine uniforme Schaltkreisfamilie vom Fanin zwei berechnet werden.

Die Tiefe und Größe uniformer Schaltkreise (2/2)

Problem: Welche Tiefe und welche Größe sind asymptotisch notwendig und hinreichend für reguläre Sprachen?

NC,
Nick's class

Komplexitätsklassen (1/2)

Sei $k \in \mathbb{N}$.

1. $\mathrm{NC}^k := \bigcup_{l=0}^{\infty} \mathrm{DEPTH} - \mathrm{SIZE}_{\mathrm{uniform}}(\log_2^k n, n^l)$.
2. $\mathrm{NC} := \bigcup_{k \in \mathbb{N}} \mathrm{NC}^k$
3. AC^k ist wie NC^k definiert, allerdings beschränken wir den Fanin nicht.
4. $\mathrm{AC} := \bigcup_{k \in \mathbb{N}} \mathrm{AC}^k$

Rasant schnell = poly-logarithmische Zeit,
Vernünftige Größe = polynomielle Größe.

Komplexitätsklassen (2/2)

(a) $\mathrm{AC}^k \subseteq \mathrm{NC}^{k+1} \subseteq \mathrm{AC}^{k+1}$.

(b) $\mathrm{AC} = \mathrm{NC} \subseteq \mathrm{P}$.

(a) $\mathrm{NC}^{k+1} \subseteq \mathrm{AC}^{k+1}$ ist offensichtlich.
$\mathrm{AC}^k \subseteq \mathrm{NC}^{k+1}$: Die uniforme Schaltkreisfamilie $(S_n)_{n\in\mathbb{N}}$ sei gegeben.

- Ein **UND**- bzw. **UND**-Gatter mit p Eingängen kann durch einen binären Baum der Tiefe $\lceil \log_2 p \rceil$ und Größe höchstens $2p + 1$ simuliert werden.
- Der Fanin von S_n ist durch $n +$ Größe(S_n) beschränkt.
- Die Ersetzung der Knoten von S_n durch Binärbäume erhöht die Tiefe um höchstens den Faktor $O(\log_2 n)$, die Größe wird höchstens quadriert.

(b) $\mathrm{AC} = \mathrm{NC}$ folgt aus Teil (a).
$\mathrm{NC} \subseteq \mathrm{P}$: Uniforme Schaltkreise polynomieller Größe können in polynomieller Zeit ausgewerten werden.

Was geht wie schnell? (Ohne Beweis)

- In AC^0 liegen
  - die Addition von zwei n-Bit Zahlen,
  - die Multiplikation Boolescher Matrizen,
  - für jede Konstante $m \in \mathbb{N}$ und jedes $k \leqslant \log_2^m n$ die Frage, ob eine Eingabe höchstens, mindestens oder genau k Einsen besitzt.

- In NC^1 liegen
  - die Berechnung $x_1 \oplus x_2 \oplus \cdots \oplus x_n$ der XOR-Funktion,
  - die Multiplikation von zwei n-Bit Zahlen.

- In AC^1 liegen
  - die transitive Hülle von Graphen,
  - das Wortproblem für kontextfreie Sprachen.

- In NC liegt das Maximum Matching Problem.

- In P, aber wahrscheinlich nicht in NC, liegen
  - das Auswertungsproblem für einen Schaltkreis S:
    Bestimme die Ausgabe von S für Eingabe x.
  - das Problem der linearen Programmierung.

Parallele Rechenzeit und Speicherplatz

Parallele Rechenzeit und Speicherplatz (1/4)

Die uniforme Schaltkreisfamilie $\mathcal{S} = (S_n)_{n\in\mathbb{N}}$ habe die Tiefe $s(n) = \Omega(\log_2 n)$.

Eine platz-effiziente Simulation von S_n:

- Werte S_n durch eine in der Senke von S beginnende Tiefensuche aus.
- Anstatt S_n abzuspeichern, benutze die Uniformität von $\mathcal{S}$, um die jeweils benötigte Information über S_n neu zu berechnen.
  - Der Weg der Tiefensuche wird durch die Bitfolge $\vec{b}$ repräsentiert.
    - Falls $b_i = 1$ (bzw. $b_i = 0$), ist der $(i+1)$-te Knoten des Wegs der rechte (bzw. linke) Nachfolger des i-ten Knoten.
  - Die Länge von $\vec{b}$ ist proportional zur Tiefe $s(n)$.
  - Die Turingmaschine rechnet mit Speicherplatz $O(s(n))$.

Als Konsequenz:

$$\mathrm{DEPTH}_{\mathrm{uniform}}(s) \subseteq \mathrm{DSPACE}(s).$$

Parallele Rechenzeit und Speicherplatz (2/4)

Wir betrachten uniforme Schaltkreise mit unbeschränktem Fanin.

(a) Berechne das Boolesche Matrizenprodukt in Tiefe zwei mit $O(n^3)$ Gattern

$$(A \cdot B)[i,j] = \bigvee_{k=1}^{n} A[i,k] \wedge B[k,j]$$

(b) und die transitive Hülle eines gerichteten Graphen in Tiefe $O(\log_2 n)$ und Größe $O(n^3 \log_2 n)$.

- Das Matrizenprodukt: Ein Brute-Force Ansatz funktioniert.
- Die transitive Hülle: Für die Adjazenzmatrix A_G
  - zeige durch Induktion über d:

    Es gibt genau dann in G einen Weg der Länge d von i nach j , wenn $A_G^d[i,j] = 1$.
  - E_n sei die $n \times n$ Einheitsmatrix:

    Es gibt genau dann einen Weg von i nach j, wenn

    $$(A_G \vee E_n)^n[i,j] = 1.$$

Parallele Rechenzeit und Speicherplatz (3/4)

- Es gibt genau dann einen Weg von i nach j, wenn

$$(A_G \vee E_n)^n[i,j] = 1.$$

! Berechne $(A_G \vee E_n)^n$.

- Ein Boolesches Matrizenprodukt kann (mit unbeschränktem Fanin) in konstanter Tiefe und Größe $O(n^3)$ ausgeführt werden.
- Wenn n eine Zweierpotenz ist, dann benutze **iteriertes Quadrieren**

$$(A_G \vee E_n)^{2^{k+1}} = ((A_G \vee E_n)^{2^k})^2,$$

um $(A_G \vee E_n)^n$ in Tiefe $O(\log_2 n)$ und Größe $O(n^3 \log_2 n)$ zu berechnen.

Parallele Rechenzeit und Speicherplatz (4/4)

$s : \mathbb{N} \to \mathbb{N}$ mit $s(n) = \Omega(\log_2 n)$ sei platz-konstruierbar. Dann ist

$$\text{DSPACE}(s) \subseteq \text{NSPACE}(s) \subseteq \text{DEPTH} - \text{SIZE}_{\text{uniform}}(s^2, 2^{O(s)}) \subseteq \text{DSPACE}(s^2).$$

Zu zeigen $\text{NSPACE}(s) \subseteq \text{DEPTH} - \text{SIZE}_{\text{uniform}}(s^2, 2^{O(s)})$.

- Sei M eine nichtdeterministische TM mit $\text{NSPACE}_M(w) = O(s)$.
- Für Eingabe w ist zu entscheiden, ob es einen Weg vom Startknoten zu einem akzeptierenden Knoten im Berechnungsgraphen $G_M(w)$ gibt.
  - Man kann annehmen, dass es genau eine akzeptierende Konfiguration gibt.
  - Wir müssen D-REACHABILITY für $G_M(w)$ lösen. D-REACHABILITY ist aber ein Spezialfall der Berechnung der transitiven Hülle.
  - $G_M(w)$ hat $\leqslant N = 2^{O(s(n))}$ Knoten.

D-REACHABILITY ist in Tiefe $O(\log_2^2 N) = O(s^2)$ und Größe $N^3 \log N = 2^{O(s(n))}$ durch eine uniforme Schaltkreisfamilie vom Fanin zwei lösbar.

P-Vollständigkeit

Die LOGSPACE-Reduktion

Wir wählen die LOGSPACE-Reduzierbarkeit, um die Parallelisierbarkeit zweier Sprachen zu vergleichen. Warum?

(a) $\mathrm{DL} \subseteq \mathrm{NC}^2$: Schaltkreise können eine s-platzbeschränkte Turingmaschine in Tiefe s^2 **und** Größe $2^{O(s(n))}$ simulieren.

(b) Aus $L_1 \leqslant_{\mathrm{LOG}} L_2$ und $L_2 \in \mathrm{NC}$ folgt $L_1 \in \mathrm{NC}$.

- Es gibt eine TM mit $\mathrm{DSPACE}_M(w) = O(\log_2 |w|)$ und

$$w \in L_1 \Leftrightarrow M(w) \in L_2.$$

- Jedes Ausgabebit von M kann durch eine Schaltkreisfamilie $(T_n | n \in \mathbb{N})$ in polynomieller Größe und Tiefe $O(\log_2^2 n)$ berechnet werden.
- Wenn L_2 durch eine uniforme Schaltkreisfamilie (mit polynomieller Größe und polylogarithmischer Tiefe) erkannt wird, so ist auch L_1 in polynomieller Größe und polylogarithmischer Tiefe erkennbar.
- Daher folgt $L_2 \in \mathrm{NC} \Rightarrow L_1 \in \mathrm{NC}$.

P-Vollständigkeit

(a) Eine Sprache L heißt genau dann P-hart, wenn $K \leqslant_{\text{LOG}} L$ für alle Sprachen $K \in \text{P}$ gilt.

(b) Eine Sprache L heißt genau dann P-vollständig, wenn $L \in \text{P}$ und wenn L P-hart ist.

Wenn L eine P-vollständige Sprache ist, dann gilt

$$\text{P} = \text{NC} \Leftrightarrow \mathbf{L} \in \text{NC}.$$

(P-vollständige Sprachen sind vom Standpunkt der Parallelisierbarkeit die schwierigsten Probleme in P.)

- $\Rightarrow$: Da $L \in \text{P}$ und da nach Voraussetzung $\text{P} = \text{NC}$, ist $L \in \text{NC}$.
- $\Leftarrow$: Nach Voraussetzung ist $L \in \text{NC}$.
  - Aber L ist P-vollständig und deshalb ist $K \leqslant_{\text{LOG}} L$ für alle $K \in \text{P}$.
  - Also folgt $K \in \text{NC}$ für alle $K \in \text{P}$ und deshalb

$$\text{P} \subseteq \text{NC}.$$

  - Aber $\text{NC} \subseteq \text{P}$, und deshalb $\text{P} = \text{NC}$.

Circuit Value ist P-vollständig

Das Circuit Value Problem (CVP)

CVP spielt als generisches Problem für die P-Vollständigkeit dieselbe Rolle wie KNFSAT für die NP-Vollständigkeit.

Wir nehmen an, daß der Fanin für alle zu betrachtenden Schaltkreise höchstens zwei ist und betrachten die folgenden Auswertungsprobleme:

$$\begin{aligned}
\text{CVP} &= \{\langle S\rangle x \mid \text{der Schaltkreis } S \text{ akzeptiert Eingabe } x\}\\
\text{M-CVP} &= \{\langle S\rangle x \mid \text{der monotone Schaltkreis } S \text{ akzeptiert Eingabe } x\}
\end{aligned}$$

Ein Schaltkreis ohne Negationsgatter heißt monoton.

$$\text{NOR-CVP} = \{\langle S\rangle x \mid \text{der NOR-Schaltkreis } S \text{ akzeptiert Eingabe } x\}.$$

CVP, M-CVP und NOR-CVP liegen alle in P.

CVP ist P-vollständig (1/2)

- Zeige $L \leqslant_{\mathrm{LOG}} \mathrm{CVP}$ für eine beliebige Sprache $L \in \mathrm{P}$.
- $L = L(M)$ für eine deterministische Turingmaschine M, die L in höchstens $\mathrm{poly}(n)$ Schritten akzeptiert.
- Simuliere M für Eingaben der Länge n durch einen Schaltkreis S_n.
  - S_n hat die Grobstruktur einem zwei-dimensionalen Gitters.
  - Die i-te „Zeile“ des Gitters gibt Bandinhalt, Kopfposition und Zustand von M zum Zeitpunkt i wieder.
    - Die i-te Zeile von S ist aus kleinen Schaltkreisen $S_{i,j}$ konstanter Größe aufgebaut, wobei $S_{i,j}$ die Zelle j des Bandes zum Zeitpunkt i simuliert.
    - $S_{i,j}$ muß den von Schaltkreis $S_{i-1,j}$ berechneten Bandinhalt speichern können, *falls der Kopf von M zum Zeitpunkt i die Zelle j nicht besucht,*
    - Sonst muß $S_{i,j}$ den Bandinhalt verändern sowie den neuen Zustand und die neue Richtung festlegen, abhängig vom gegenwärtigen Zustand und Bandinhalt.
  - Das ist alles kein Problem, wenn wir die Ausgänge von $S_{i-1,j-1}$, $S_{i-1,j}$ und $S_{i-1,j+1}$ zu Eingängen von $S_{i,j}$ machen.

CVP ist P-vollständig (2/2)

- Sämtliche Schaltkreise $S_{i,j}$ können „baugleich“ gewählt werden, mit Ausnahme der Schaltkreise $S_{0,j}$,
  - die entweder zu setzen sind ($j \notin \{1, \ldots, n\}$)
  - oder an die Eingabe anzuschliessen sind ($j \in \{1, \ldots, n\}$).
- Wir müssen das Gitter noch „auswerten“, d.h. wir müssen feststellen, ob der letzte Zustand akzeptierend ist.
  - Dies gelingt, wenn wir einen binären Auswertungsbaum „auf“ das Gitter setzen.
- S_n kann durch eine logarithmisch-platzbeschränkte Turingmaschine konstruiert werden.

P, NP und kleine Schaltkreise

Unser P-Vollständigkeitsergebnis für CVP hat gezeigt:

- Wenn $L = L(M)$ für eine Turingmaschine M mit Laufzeit t, dann wird L von einer uniformen Schaltkreisfamilie der Größe $O(t^2)$ akzeptiert.
- Um P $\neq$ NP zu zeigen, genügt der Nachweis, dass KNF-SAT keine Schaltkreisfamilie polynomieller Größe besitzt.

M-CVP ist P-vollständig

- Wir zeigen CVP $\leqslant_{\text{LOG}}$ M-CVP.
  - Wir „schieben“ sämtliche Negationsgatter hinab zu den Quellen von S.
  - Die Booleschen Regeln implizieren, dass $\wedge$- und $\vee$-Gatter vertauscht werden müssen sowie neue Quellen für negierte Eingaben einzuführen sind.
  - Aber Achtung: Sowohl die Ausgabe eines Gatters wie auch die negierte Ausgabe können benötigt werden!
    - ⋆ Ersetze jedes Gatter v durch zwei Gatter $(v, 0)$ und $(v, 1)$ mit den Ausgaben v beziehungsweise $\neg v$.
    - ⋆ Durch eine Verdopplung der Größe können beide Gatterversionen simultan berechnet werden.
    - ⋆ Beachte, dass sich auch die Anzahl der Eingabebits verdoppelt: Statt $x_1, \ldots, x_n$ haben wir jetzt $x_1, \ldots, x_n$ **und** $y_1, \ldots, y_n$ (mit $x_i \oplus y_i = 1$).
- Der neue Schaltkreis und die neue Eingabe lässt sich durch eine logarithmisch-platzbeschränkte Turingmaschine beschreiben.

NOR-CVP ist P-vollständig

Es ist $\text{nor}(u, v) = \neg(u \vee v)$.

- Zeige: CVP $\leqslant_{\text{LOG}}$ NOR-CVP.
- Jede der Operationen $\wedge$, $\vee$ und $\neg$ kann man mit NOR-Gattern darstellen:
  - $\text{nor}(\neg u, \neg v) = u \wedge v$
  - $\text{nor}(\text{nor}(u, v), \text{nor}(u, v)) = u \vee v$.
  - $\text{nor}(u, u) = \neg u$
- Eine logarithmisch-platzbeschränkte Turingmaschine überführt einen $\{\wedge, \vee, \neg\}$-Schaltkreis S in einen äquivalenten NOR-Schaltkreis S^*.

Die Lineare Programmierung ist P-vollständig.

Die Lineare Programmierung

A ist eine $m \times n$ Matrix von ganzen Zahlen,
$b \in \mathbb{Z}^m$ und $c \in \mathbb{Z}^n$ sind Vektoren ganzer Zahlen.

(a) Im **Problem der linearen Ungleichungen** ist zu entscheiden, ob es einen Vektor $x \in \mathbb{Q}^n$ mit $Ax \leqslant b$ gibt.

(b) Im **Problem der linearen Programmierung** ist zu entscheiden, ob es einen Vektor $x \in \mathbb{Q}^n$ mit $Ax \leqslant b$ und $c \cdot x \geqslant t$ gibt.

Das Problem der linearen Programmierung wird konventionell als Optimierungsproblem formuliert:

Maximiere $c \cdot x$, so daß $Ax \leqslant b$ und $x \geqslant 0$ gilt.

(a) Das Problem der linearen Ungleichungen ist P-vollständig.

(b) Das Problem der linearen Programmierung ist P-vollständig.

Das Problem der linearen Ungleichungen ist P-vollständig

- Wir zeigen die Reduktion M-CVP$\leqslant_{\mathrm{LOG}}$ Lineare Ungleichungen.
- Sei (S, x) eine Eingabe für M-CVP.
  Weise jedem Gatter von S Ungleichungen zu.
  - Beschreibe die Quelle zu Eingabe x_i:
    - Falls $x_i = 0$, verwende die Ungleichungen $x_i \leqslant 0$ und $-x_i \leqslant 0$.
    - Falls $x_i = 1$, verwende die Ungleichungen $x_i \leqslant 1$ und $-x_i \leqslant -1$.
  - Für ein Gatter $v \equiv u \wedge w$ verwende die Ungleichungen:
    $v \leqslant u,\ v \leqslant w,\ u + w - 1 \leqslant v,\ 0 \leqslant v$.
  - Für ein Gatter $v \equiv u \vee w$ verwende die Ungleichungen:
    $u \leqslant v,\ w \leqslant v,\ v \leqslant u + w,\ v \leqslant 1$.
  - Füge die Ungleichung $-s \leqslant -1$ für die eindeutig bestimmte Senke s des Schaltkreises hinzu.
- Zeige durch Induktion über die topologische Ordnung der Gatter:

  das lineare Ungleichungssystem ist lösbar $\Leftrightarrow$ S akzeptiert x.

Das Problem der linearen Programmierung ist P-vollständig

Zeige die Reduktion

$$\text{Lineare Ungleichungen} \leqslant_{\text{LOG}} \text{Lineare Programmierung.}$$

Wenn (A, b) eine Eingabe für das Problem der linearen Programmierung ist,

- dann übernimm A, b und
- setze $c = \vec{0}$ und $t = 0$.

Die Parallelisierbarkeit von Greedy-Algorithmen

Eine Heuristik für das Independent-Set Problem

Das Independent Set Problem

Eingabe: $G = (V, E)$ ist ein ungerichteter Graph mit Knotenmenge $V = \{1, \ldots, n\}$.

Aufgabe: Bestimme eine unabhängige Knotenmenge größter Kardinalität. (Eine Knotenmenge ist unabhängig, wenn keine zwei Knoten der Menge durch eine Kante verbunden sind.)

Die Independent Set Heuristik

1. Setze $I(G) := \emptyset$.
2. FOR $v = 1$ TO n DO
   IF (v ist mit keinem Knoten in $I(G)$ verbunden) THEN
   $I(G) = I(G) \cup \{v\}$.
3. Die Menge $I(G)$ wird ausgegeben.

Kann für jeden Knoten v super-schnell festgestellt werden, ob $v \in I(G)$?

LFMIS

Lexicographically-First-Maximal-Independent-Set-Problem (LFMIS)

Eingabe: $G = (V, E)$ ist ein ungerichteter Graph mit $V = \{1, \ldots, n\}$ und $v \in V$ ist ein ausgezeichneter Knoten.

Aufgabe: Die Menge $I = I(G)$ werde von der Independent Set Heuristik berechnet. Entscheide, ob v in der Menge I vorkommt.

LFMIS ist P-vollständig.

LFMIS ist P-vollständig (1/2)

Zeige die Reduktion NOR-CVP $\leqslant_{\mathrm{LOG}}$ LFMIS.

- (S, x) –mit Graphstruktur $G = (V, E)$– sei Eingabe von NOR-CVP.
- Konstruiere einen ungerichteten Graphen $G^* = (V^*, E^*)$ für LFMIS:
  - Füge einen neuen Knoten v_0 hinzu und setze genau dann eine Kante $\{v_0, i\}$ zum Eingabeknoten i ein, wenn $x_i = 0$ ist.
  - Es ist also $V^* = V \cup \{v_0\}$ und $E^* = \{\{v_0, i\} \mid x_i = 0\} \cup E$.

- v_0 erhält die Nummer Eins. Die Nummer eines jeden anderen Knotens entspricht der topologischen Nummer wie vom Schaltkreis vorgegeben.
- Zeige durch **Induktion** über den Wert der zugewiesenen Nummer:

$$v \in I(G^*) \Leftrightarrow v = v_0 \text{ oder das Gatter } v \text{ hat den Wert 1.}$$

LFMIS ist P-vollständig (2/2)

Zeige $v \in I(G^*) \Leftrightarrow v = v_0$ oder das Gatter v hat den Wert 1
durch **Induktion** über den Wert der zugewiesenen Nummer.

Die Reduktion funktioniert.

- Die Basis der Induktion.
  - Die Heuristik wählt stets den Knoten 1: $v_0 \in I(G^*)$.
  - Erhält die Quelle i den Wert Null, ist $i \notin I(G^*)$
    Die Heuristik kann i wegen der Kante $\{v_0, i\}$ nicht wählen.
    Hat Quelle i den Wert Eins, ist $i \in I(G^*)$.
- Der Knoten w entspreche dem Gatter $\text{nor}(u, v)$.
  - Die Nummer der Knoten u und v ist kleiner als die Nummer von w.
  - Es ist $w = \neg(u \vee v) = 1 \Leftrightarrow (u = 0) \wedge (v = 0)$.
- Nach Induktionsannahme ist u (bzw. v) genau dann in $I(G^*)$, wenn $u = 1$ (bzw. $v = 1$) ist.
- Wegen der Kanten $\{u, w\}$ und $\{v, w\}$ nimmt die Heuristik w genau dann in die Menge $I(G^*)$ auf, wenn das Gatter w den Wert Eins hat.

Zusammenfassung

Zusammenfassung

- Wir haben Schaltkreise als paralleles Rechnermodell gewählt und die Klasse NC aller parallelisierbaren Sprachen eingeführt.
  - Es besteht ein enger Zusammenhang zwischen den Komplexitätsmaßen Tiefe und Speicherplatz.
  - Die **Parallel-Computation-Thesis** postuliert eine polynomielle Beziehung zwischen der Rechenzeit eines jeden vernünftigen parallelen Rechnermodells und der Speicherplatzkomplexität.
- Die P-vollständigen Sprachen sind die im Hinblick auf Parallelisierbarkeit schwierigsten Sprachen in P.
  - Das Circuit-Value Problem spielt als generisches Problem dieselbe Rolle für die P-Vollständigkeit wie KNFSAT für die NP-Vollständigkeit.
  - Die Lineare Programmierung wie auch die Bestimmung der lexikographisch ersten maximalen unabhängigen Menge sind weitere P-vollständige Probleme.