Kapitel 6: Boolesche Schaltkreise

Boolesche Schaltkreise: Modell für
- einfache Computerchips
- paralleles Rechnen
- nicht-uniform: Für jede Eingabelänge n ein Algorithmus A_n (der ganz anders vorgehen kann als A_{n-1})

6.1 Boolesche Schaltkreise: Definition

Definition 6.1

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

(a) Ein Boolescher Schaltkreis (kurz: SK) mit n Eingabegattern und einem Ausgabegatter ist ein endlicher gerichteter azyklischer Graph, für den gilt:
- Es gibt n Knoten vom Ein-Grad 0, die mit den Symbolen $x_1, \dots, x_n$ markiert sind — die so genannten Eingabegatter.
- Es gibt einen Knoten vom Aus-Grad 0, das so genannte Ausgabegatter.
- Jeder Knoten hat Ein-Grad 0, 1 oder 2.
  - Jeder Knoten vom Ein-Grad 0, der kein Eingabegatter ist, ist mit einer der beiden Booleschen Konstanten **0** oder **1** markiert (und wird ⓪-Gatter bzw. ①-Gatter genannt).

- Jeder Knoten vom Ein-Grad 1 ist mit dem Symbol $\neg$ markiert (und wird $\neg$-Gatter genannt)
- Jeder Knoten vom Ein-Grad 2 ist mit einem der Symbole $\wedge$ bzw. $\vee$ markiert (und wird $\wedge$-Gatter bzw. $\vee$-Gatter genannt).

(b) Die Größe $|C|$ eines Sks C ist die Anzahl der Knoten von C.

(c) Ist C ein Sk mit n Eingabegattern und ist $x \in \{0,1\}^n$, so ist die Ausgabe $C(x) \in \{0,1\}$ wie folgt definiert:
$C(x) = val_{v_{out}}(x)$, für das Ausgabegatter v_{out}, wobei $val_v(x)$ für jeden Knoten v von C wie folgt definiert ist:

- Ist v ein mit x_i markiertes Eingabegatter, so ist $val_v(x) = x_i$
- Ist v ein 0-Gatter (bzw. 1-Gatter), so ist $val_v(x) = 0$ (bzw. 1)
- Ist v ein $\neg$-Gatter, so ist
$$val_v(x) = \begin{cases} 1, & \text{falls } val_u(x) = 0 \\ 0, & \text{falls } val_u(x) = 1, \end{cases}$$
wobei u der Knoten in C ist, von dem aus eine Kante nach v führt (u wird der Vorgängerknoten von v genannt).
- Ist v ein $\wedge$-Gatter, so ist

$$val_v(x) = \begin{cases} 1 & \text{falls } val_{u_1}(x) = 1 = val_{u_2}(x) \\ 0 & \text{sonst} \end{cases},$$

wobei u_1 und u_2 die beiden Vorgängerknoten von v in C sind.

- Ist v ein $\vee$-Gatter, so ist

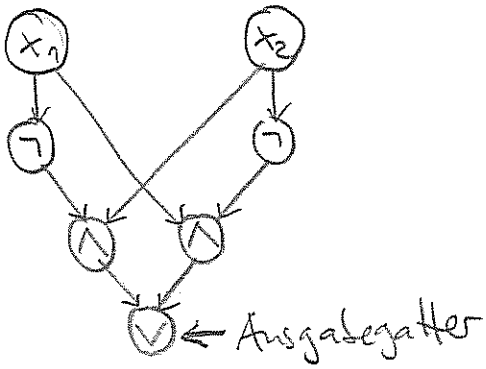
$$val_v(x) = \begin{cases} 0 & \text{falls } val_{u_1}(x) = 0 = val_{u_2}(x) \\ 1 & \text{sonst (d.h. } val_{u_1}(x) = 1 \text{ oder } val_{u_2}(x) = 1) \end{cases}$$

wobei u_1 und u_2 die beiden Vorgängerknoten von v in C sind.

(d) Ein SK C mit n Eingabegattern berechnet die Boolesche Funktion $f_C: \{0,1\}^n \to \{0,1\}$ mit $f_C(x) := C(x)$ f.a. $x \in \{0,1\}^n$

Beispiel 6.2

a) Der SK



berechnet die 2-stellige XOR-Funktion (mit $XOR(x_1,x_2) = 1 \iff x_1 \neq x_2$).

(b) Der Syntaxbaum einer aussagenlogischen Formel kann als SK aufgefasst werden, bei dem jeder Knoten, der kein Eingabegatter ist, den Aus-Grad ≤ 1 hat.

Notation 6.3

(a) In Zeichnungen von Skeln werden oft die Richtungen von Kanten oft weggelassen — Kanten verlaufen dann immer von oben nach unten

(b) Im Zusammenhang mit Skeln werden die Begriffe "Ein-Grad" und "Aus-Grad" auch oft "fan-in" bzw "fan-out" genannt.

Theorem 6.4 (Shannon, 1949)

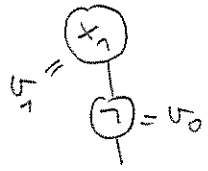
Für jedes $n \geq 1$ und jedes $f: \{0,1\}^n \to \{0,1\}$ gibt es einen SK C_f der Größe $\leq 2 \cdot \frac{2^n}{n}$, der f berechnet.

Beweis:

Behauptung 1: Für jedes $\ell \geq 1$ gibt es einen SK C_ℓ mit Eingabegatter $x_1, \dots, x_\ell$ und insgesamt $\leq 3 \cdot 2^\ell$ Gattern, in dem es für jedes $a \in \{0,1\}^\ell$ einen Knoten v_a gibt, der genau dann den Wert 1 hat, wenn an den Eingabegatter $x_1 \dots x_\ell$ die Werte $a_1 \dots a_\ell$ eingegeben werden

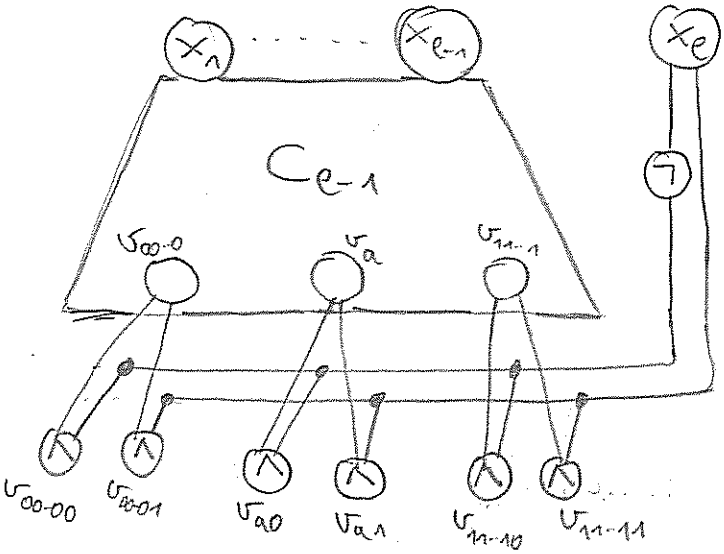
Beweis: Per Induktion nach ℓ

$\ell = 1$: C_1 sieht folgendermaßen aus:



klar: $|C_1| = 2 \leq 3 \cdot 2^1$

$\ell - 1 \to \ell$: C_ℓ sieht folgendermaßen aus:



Man sieht leicht, dass für jedes $a \in \{0,1\}^{\ell-1}$ die Knoten v_{a0} und v_{a1} das Gewünschte leisten.

Außerdem gilt:

$$|C_\ell| = |C_{\ell-1}| + 2^\ell + 2$$

Auflösen der Rekursionsgleichung liefert:

$$|C_\ell| = 2 + 2^\ell + |C_{\ell-1}| = 2 \cdot 2 + (2^\ell + 2^{\ell-1}) + |C_{\ell-2}| = 3 \cdot 2 + (2^\ell + 2^{\ell-1} + 2^{\ell-2}) + |C_{\ell-3}|$$
$$= \dots = i \cdot 2 + (2^\ell + 2^{\ell-1} + 2^{\ell-2} + \dots + 2^{\ell-(i-1)}) + |C_{\ell-i}| = \dots = (\ell-1) \cdot 2 + (2^\ell + 2^{\ell-1} + \dots + 2^2) + |C_1|$$
$$= 2\ell - 2 + 2^{\ell+1} - 1 - 2^0 - 2^1 + |C_1| = 2\ell + 2^{\ell+1} - 6 + 2 = 2 \cdot 2^\ell + \underbrace{2\ell - 4}_{> 2^\ell} < 3 \cdot 2^\ell$$

□ Beh. 1

Behauptung 2: Für jedes $k \geq 1$ gibt es einen SK D_k mit Eingabegattern $x_1, \dots, x_k$ und insgesamt $\leq 2 \cdot 2^k \cdot 2^{2^k}$ Gattern, in dem es für jede Funktion $g: \{0,1\}^k \to \{0,1\}$ einen Knoten v_g gibt, der die Funktion g berechnet (d.h. v_g hat genau dann den Wert 1, wenn an den Eingabegattern $x_1, \dots, x_k$ Werte $a_1 \dots a_k \in \{0,1\}^k$ eingegeben werden, für die gilt: $g(a_1 \dots a_k) = 1$)

Beweis:

Für jedes $g: \{0,1\}^k \to \{0,1\}$ gilt:

$$g(x_1, \dots, x_k) = \bigvee_{\substack{a \in \{0,1\}^k: \\ g(a)=1}} \text{"} x_1 \dots x_k = a \text{"}$$

Zum Bau des SKs D_k erweitern wir den SK C_k aus Behauptung 1 wie folgt:

Für jedes $g: \{0,1\}^k \to \{0,1\}$ fügen wir eine Kette von (höchstens $2^k - 1$) $\vee$-Gattern ein, durch die die Ver-ODER-ung sämtlicher Knoten v_a für $a \in \{0,1\}^k$ mit $g(a) = 1$ berechnet wird. Das letzte dieser $\vee$-Gatter ist der Knoten v_g.

Man sieht leicht, dass der so erhaltene SK D_k das Gewünschte leistet. Außerdem gilt:

$$|D_k| \leq |C_k| + \underbrace{2^{2^k}}_{\substack{\text{Anzahl der Fkten} \\ g: \{0,1\}^k \to \{0,1\}}} \cdot 2^k \underset{\text{Beh 1}}{\leq} 3 \cdot 2^k + 2^{2^k} \cdot 2^k \leq \underbrace{2 \cdot 2^k \cdot 2^{2^k}}_{\substack{3 \cdot 2^k < 2^{2^k} \cdot 2^k \\ \text{f.a. } k \geq 1}}$$

□ Beh 2

Abschluss des Beweises:

Sei $n \geq 1$ und sei $f: \{0,1\}^n \to \{0,1\}$.

Ziel: Konstruktion eines SK C_f der Größe $\leq 22 \cdot \frac{2^n}{n}$, der f berechnet.

Wir wählen $k := \log n - 2$ und $\ell := n - k$.

Sei D_k der SK aus Beh 2, und

sei C'_ℓ der SK aus Beh 1, der an Stelle von $x_1, \dots, x_\ell$ die Gatter $x_{k+1}, \dots, \underbrace{x_{k+\ell}}_{=n}$ als Eingabegatter hat.

Für jedes $a \in \{0,1\}^\ell$ sei die Funktion

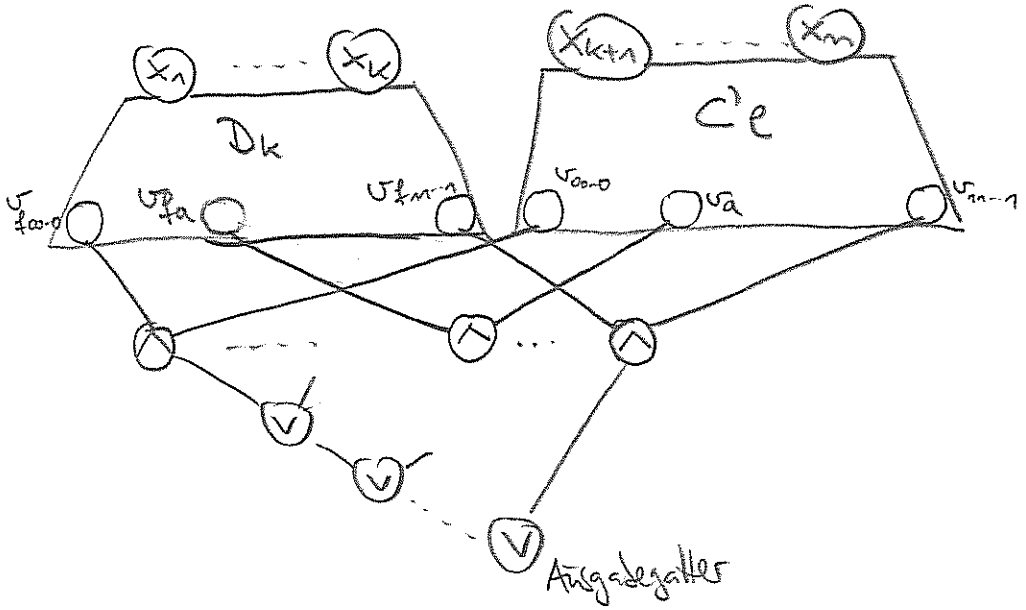
$f_a: \{0,1\}^k \to \{0,1\}$ definiert durch

$$f_a(x_1, \dots, x_k) := f(x_1, \dots, x_k, a_1, \dots, a_\ell)$$

Man sieht leicht, dass folgendes gilt:

$$\circledast \quad f(x_1, \dots, x_n) = \bigvee_{a \in \{0,1\}^\ell} \left(f_a(x_1, \dots, x_k) \wedge "x_{k+1} \dots x_n = a" \right)$$

Sei C_f der folgende SK, der $\circledast$ realisiert und daher f berechnet:



$\} 2^\ell$ $\wedge$-Gatter

$\} 2^\ell - 1$ $\vee$-Gatter

Es gilt:

$$|C_f| = |D_k| + |C'_\ell| + 2^\ell + 2^\ell - 1 \underset{\text{Beh 1+2}}{\leq} 2 \cdot 2^k \cdot 2^{2^k} + 3 \cdot 2^\ell + 2^\ell + 2^\ell - 1$$

Also: $|C_\ell| \leq 2 \cdot 2^k \cdot 2^{2^k} + 5 \cdot 2^\ell$ (*)

Wegen $k = \log n - 2$ und $\ell = n - k = n - \log n + 2$ gilt:

$$5 \cdot 2^\ell = 5 \cdot 2^{n - \log n + 2} = 5 \cdot 2^2 \cdot 2^{n - \log n} = 5 \cdot 4 \cdot \frac{2^n}{n} = 20 \cdot \frac{2^n}{n} \quad (**)$$

und

$$2 \cdot 2^k \cdot 2^{2^k} = 2 \cdot 2^{\log n - 2} \cdot 2^{2^{\log n - 2}} = 2^{\log n - 1} \cdot 2^{\frac{n}{4}} = 2^{\frac{n}{4} + \log n - 1} \quad (***)$$

[Behauptung 3: F.a. $n \geq 1$ gilt: $\frac{n}{4} + \log n - 1 \leq n - \log n + 1$

Beweis:

$$\frac{n}{4} + \log n - 1 \leq n - \log n + 1$$

$$\Leftrightarrow \quad 2 \log n \leq \frac{3}{4} n + 2$$

$$\Leftrightarrow \quad \frac{8}{3} \log n \leq n + \frac{8}{3}$$

$$\Leftrightarrow \quad 2^{\frac{8}{3} \log n} \leq 2^{n + \frac{8}{3}}$$

$$\Leftrightarrow \quad n^{8/3} \leq 2^{8/3} \cdot 2^n$$

$$\Leftarrow \quad n^3 \leq 2^2 \cdot 2^n = 4 \cdot 2^n$$

Somit sind wir fertig, sobald bewiesen ist, dass f.a. $n \geq 1$ gilt: $n^3 \leq 4 \cdot 2^n$. Rest: Übung!

□ Beh 3

Wegen Beh 3 und (***) gilt:

$$2 \cdot 2^k \cdot 2^{2^k} \leq 2^{n - \log n + 1} = 2 \cdot 2^{n - \log n} = 2 \cdot \frac{2^n}{n}.$$

Mit (*) und (**) folgt: $|C_\ell| \leq 2 \cdot \frac{2^n}{n} + 20 \frac{2^n}{n} = 22 \cdot \frac{2^n}{n}$. □

Das nächste Theorem zeigt, dass die Konstruktion des Schaltkreises C_ℓ im obigen Beweis fast optimal ist.

Theorem 6.5 (Shannon, 1949)

Für jedes $n \geq 1$ gibt es eine Funktion $f: \{0,1\}^n \to \{0,1\}$, die durch keinen SK der Größe $\leq \frac{1}{10} \cdot \frac{2^n}{n}$ berechnet wird.

Beweis: Wir nützen ein einfaches Zählargument.

Sei F_n die Anzahl der Funktionen $f: \{0,1\}^n \to \{0,1\}$, d.h. $F_n = 2^{2^n}$.

Für $s := \frac{1}{10} \cdot \frac{2^n}{n}$ sei K_s die Anzahl der SKe der Größe $\leq s$.

Wir zeigen im Folgenden, dass $K_s < 2^{2^n} = F_n$ ist.

Um K_s abzuschätzen, repräsentieren wir jeden SK C der Größe $\leq s$ wie folgt durch einen Bitstring $b(C)$:

- Nummeriere die Knoten von C mit den Zahlen $0, 1, \ldots, |C|-1 \leq s$. Der Ausgabeknoten bekommt die Nummer 0.
- Für jeden Knoten $i \in \{0, \ldots, |C|-1\}$ sei $b(i)$ der wie folgt definierte Bitstring:

  (1) Die ersten $\log(n+5)$ Bits geben an, von welcher Sorte aus $\{x_1, \ldots, x_n, \wedge, \vee, \neg, 0, 1\}$ der Knoten i ist

  (2) Die nächsten 0, $\log s$ oder $2\log s$ Bits geben die 0, 1 oder 2 Knoten an, von denen aus Kanten zum Knoten i hinführen

  Beachte: Für (1) und (2) werden $\leq \log(n+5) + 2\log s$ Bits benötigt. Das ist $\leq 9 \cdot \log s$, denn:

  $\log(n+5) \leq 7 \log s \iff n+5 \leq 2^7 \cdot s \iff n+5 \leq 128 \cdot s \iff$

  $n+5 \leq \frac{128}{10} \cdot \frac{2^n}{n} \impliedby n^2+5n \leq 10 \cdot 2^n$ — und dies gilt f.a. $n \geq 0$.

  (3) Wir füllen $b(i)$ mit Oen auf zu einem Bitstring der Länge $9 \log s$

- Wir konkatenieren die Bitstrings $b(i)$ für $i = 0, 1, \dots, |C|-1$ und füllen das Ganze mit Oen auf einem Bitstring $b(C)$ der Länge $s \cdot g\log s$

  Beachte: Aus $b(C)$ lässt sich eindeutig der SK C rekonstruieren.

Insbesondere gilt:

$K_s \overset{\text{Def}}{=}$ Anzahl der SKe der Größe $\leq s$ $\leq$ Anzahl der Bitstrings der Länge $g s \log s$ $\leq 2^{g s \log s}$.

Somit gilt:

$$K_s \leq 2^{g s \log s} \overset{s = \frac{1}{10}\frac{2^n}{n}}{=} 2^{\frac{g}{10}\cdot\frac{2^n}{n}\log\left(\frac{2^n}{10n}\right)} = 2^{2^n\left(\frac{g}{10n}\log\left(\frac{2^n}{10n}\right)\right)}$$

$$\leq 2^{2^n\left(\frac{g}{10n}\cdot\log(2^n)\right)}$$

$$= 2^{2^n\left(\frac{g}{10n}\cdot n\right)} = 2^{2^n\cdot\frac{g}{10}} < 2^{2^n} = \#_n$$

□

6.3 Die Klasse SIZE(S(n))

Definition 6.6 (Die Klasse SIZE(S(n)))

(a) Eine Schaltkreisfamilie (kurz: SK-Familie) ist eine Folge $(C_n)_{n\in\mathbb{N}}$, wobei für jedes $n\in\mathbb{N}$ C_n ein SK mit n Eingabegattern ist

(b) Die von einer SK-Familie $(C_n)_{n\in\mathbb{N}}$ berechnete (oder definierte) Sprache ist

$$L := \{ x\in\{0,1\}^* : n\in\mathbb{N},\ C_n(x) = 1 \}$$

(c) Sei $S: \mathbb{N}\to\mathbb{N}$.
Eine SK-Familie $(C_n)_{n\in\mathbb{N}}$ hat Größe $S(n)$, falls $\forall\, n\in\mathbb{N}$ gilt $|C_n| \le S(n)$.

(d) Sei $S:\mathbb{N}\to\mathbb{N}$.

$$\mathrm{SIZE}(S(n)) := \{ L\subseteq\{0,1\}^* : \text{Es gibt eine SK-Familie der Größe } S(n), \text{ die } L \text{ berechnet} \}$$

Bemerkung 6.7

Aus Theorem 6.4 folgt, dass für jede Funktion $S:\mathbb{N}\to\mathbb{N}$ mit $S(n) \ge 22\cdot\frac{2^n}{n}$ (für $n\ge 1$) gilt:

$$\mathrm{SIZE}(S(n)) = \{ L : L\subseteq\{0,1\}^* \}$$

(d.h. alle Sprachen liegen in $\mathrm{SIZE}(S(n))$).

Aus Shannons Theoremen 6.4 und 6.5 erhalten wir folgenden Hierarchiesatz:

Theorem 6.8 (Nichtuniformer Hierarchiesatz)

Seien $s: \mathbb{N} \to \mathbb{N}$ und $S: \mathbb{N} \to \mathbb{N}$ s.d. f.a. hinreichend großen $n \in \mathbb{N}$ gilt:

(1) $n \leq s(n) \leq \frac{1}{10} \cdot \frac{2^n}{n}$ und

(2) $440 \cdot s(n) \leq S(n)$.

Dann gilt:

$$SIZE(s(n)) \subsetneq SIZE(S(n))$$

Beweis: Sei n hinreichend groß, s.d. (1) und (2) gilt.

Fall 1: $S(n) \geq 22 \cdot \frac{2^n}{n}$

Wegen Theorem 6.4 kann jedes $f: \{0,1\}^n \to \{0,1\}$ durch einen SK der Größe $S(n)$ berechnet werden.
Wegen Theorem 6.5 gibt es ein $f: \{0,1\}^n \to \{0,1\}$, das durch keinen SK der Größe $s(n) \leq \frac{1}{10} \cdot \frac{2^n}{n}$ berechnet wird
Somit ist $SIZE(s(n)) \subsetneq SIZE(S(n))$.

Fall 2: $S(n) < 22 \cdot \frac{2^n}{n}$

Wähle $\ell \in \{1, \dots, n\}$ maximal, s.d. gilt: ⊛

$$S(n) \geq 22 \cdot \frac{2^\ell}{\ell} - (n-\ell)$$

Beachte: Es gilt $\ell \in \{1, \dots, n-1\}$, denn:

$22 \cdot \frac{2^1}{1} - (n-1) = 44 - n + 1 \leq 45 < 440 \cdot 1 \leq 440 \cdot s(n) \leq S(n)$ und

$22 \cdot \frac{2^n}{n} - (n-n) = 22 \cdot \frac{2^n}{n} > S(n)$, da wir in Fall 2 sind

Gemäß Theorem 6.5 und 6.4 gibt es ein $f: \{0,1\}^\ell \to \{0,1\}$, das durch keinen SK der Größe $\frac{1}{10}\frac{2^\ell}{\ell}$, aber durch einen SK der Größe $22 \cdot \frac{2^\ell}{\ell}$ berechnet werden kann.

Sei $g: \{0,1\}^n \to \{0,1\}$ mit $g(x_1 \dots x_n) := f(x_1, \dots, x_\ell)$.

Klar: g wird durch einen SK der Größe $\leq$

$$\underbrace{22 \cdot \frac{2^\ell}{\ell}}_{\text{SK für } f} + \underbrace{(n-\ell)}_{\substack{\text{zusätzliche} \\ \text{Eingabegatter} \\ \text{für } x_{\ell+1}, \dots, x_n}} \overset{\circledast}{\leq} S(n)$$

Behauptung: g wird durch keinen SK der Größe $s(n)$ berechnet.

Beweis: Angenommen doch, dann wird auch f durch diesen SK berechnet (indem die Eingabegatter $x_{\ell+1}, \dots, x_n$ durch ein mit 0 markiertes Gatter ersetzt werden).

Wir zeigen, dass $s(n) \leq \frac{1}{10} \cdot \frac{2^\ell}{\ell}$ ist — ein Widerspruch zur Wahl von f.

Wegen $\circledast$ (Maximalität von ℓ) gilt:

$$S'(n) < 22 \cdot \frac{2^{\ell+1}}{\ell+1} - \underbrace{(n-(\ell+1))}_{\geq 0, \text{ da } \ell+1 \leq n} \leq 22 \cdot \frac{2^{\ell+1}}{\ell} = 44 \cdot \frac{2^\ell}{\ell}$$

V1 (2)

$$440 \cdot s(n)$$

Somit ist $s(n) \leq \frac{44}{440} \cdot \frac{2^\ell}{\ell} = \frac{1}{10} \cdot \frac{2^\ell}{\ell}$. □ Behauptung.

Wir haben gezeigt, dass g durch einen SK der Größe $S(n)$, nicht aber durch einen SK der Größe $s(n)$ berechnet wird.

Somit ist $\text{SIZE}(s(n)) \subsetneq \text{SIZE}(S(n))$.

□

Folgerung 6.9

- $SIZE(n) \subsetneq SIZE(500n)$ und
- $SIZE(n) \subsetneq SIZE(n^2) \subsetneq SIZE(n^3) \subsetneq \dots$
- $\bigcup_{c \geq 1} SIZE(n^c) \subsetneq SIZE\left(\frac{2^n}{n}\right)$

6.4 Die Klasse P/poly

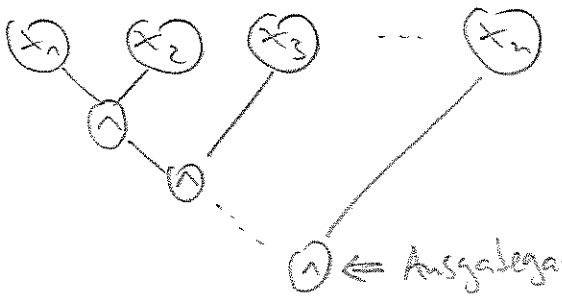
Definition 6.10 (P/poly)

$$P/poly := \bigcup_{c \geq 1} SIZE(n^c)$$

ist die Klasse aller Sprachen, die durch SK-Familien polynomieller Größe berechnet werden.

Beispiel 6.11

Die Sprache $\{1^n : n \in \mathbb{N}\}$ wird durch eine SK-Familie der Größe $2n-1$ berechnet und gehört daher zu P/poly.
Der SK C_n dieser SK-Familie ist einfach die Konjunktion aller n Eingabegatter:



Bemerkung 6.12

(a) Jede unäre Sprache L (d.h. $L \subseteq \{1\}^*$) gehört zu P/poly (und wird durch eine Sk-Familie der Größe $O(n)$ berechnet):
Die Sk-Familie $(C_n)_{n\in\mathbb{N}}$, die L berechnet, kann wie folgt gewählt werden:
F.a. $n\in\mathbb{N}$ gilt:

- Falls $1^n \in L$, so ist C_n der Sk aus Beispiel 6.11
- Falls $1^n \notin L$, so ist C_n ein Sk, der immer 0 ausgibt — etwa:

(b) P/poly enthält einige unentscheidbare Sprachen — z.B. die unäre Sprache

$$\text{UHALT} := \{1^n : \text{die Binärdarstellung von } n \text{ repräsentiert eine TM } M, \text{ die bei leerer Eingabe anhält}\}.$$

(c) Wegen Theorem 6.5 bzw. dem Hierarchiesatz Theorem 6.8 gibt es Sprachen, die nicht in P/poly liegen.

Theorem 6.13

$P \subseteq P/poly$

Beweisskizze:

Ähnlich wie der Beweis des Satzes von Cook und Levin (NP-Vollständigkeit von SAT):

Sei $L \in P$. Dann gibt es ein $c \geq 1$ und ein $T(n) := n^c$ - zeitbeschränkte stereotype (engl.: oblivious) det. 1-Band TM M, die L entscheidet (vgl. Bemerkung 1.10)

D.h.: Die Kopfposition im i-ten Berechnungsschritt hängt nur von der Länge der Eingabe ab, aber nicht von der konkreten Eingabe selbst.

Wir können außerdem o.B.d.A. annehmen, dass M in den ersten $n = |x|$ Berechnungsschritten im Zustand q_0 einmal von links nach rechts über das Band läuft, ohne dessen Beschriftung zu verändern. (*)

Zur Konstruktion des SKes C_n betrachten wir für jede Eingabe $x \in \{0,1\}^n$ die Folge

$$z_0, z_1, \dots, z_n, z_{n+1}, \dots, z_{T(n)}$$

von sog. Schnappschüssen von M. D.h.: z_i gibt den aktuellen Zustand sowie das an der aktuellen Kopfposition gelesene Symbol an. Insbes. gilt wegen (*):

$z_0 = (q_0, \triangleright)$ und $z_i = (q_0, x_i)$ f.a. $i \in \{1, \dots, n\}$.

Außerdem gilt f.a. $i \in \{n+1, \dots, T(n)\}$:

(**): Durch die Überführungsfunktion δ von M ist

z_i eindeutig festgelegt durch z_{i-1} und $z_{prev(i)}$, wobei prev(i) der letzte Zeitpunkt vor i ist, zu dem der Kopf auf derselben Bandposition wie zum Zeitpunkt i war (bzw $z_{prev(i)} := \perp$, falls i der erste Zeitpunkt ist, an dem diese Bandposition betreten wird)

Da M stereotyp ist, hängt prev(i) nicht von der Eingabe x ab, sondern nur von i und $n := |x|$. (***)

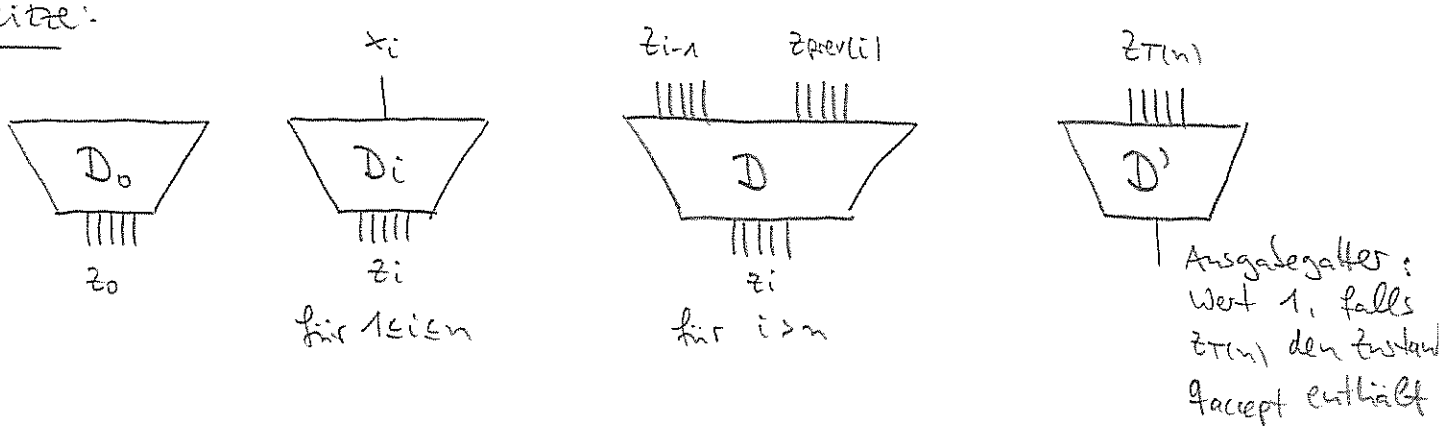
Jeden Schnappschuss können wir durch eine konstante Anzahl k von Bits repräsentieren (k hängt nur von der Zustandsmenge Q und dem Bandalphabet Γ von M ab, aber nicht von n oder x).

Unser SK C_n enthält für jeden der Schnappschüsse $z_0, z_1, \ldots, z_{T(n)}$ k Gatter, deren Wert bei Eingabe x den Schnappschuss repräsentieren.

- $z_0 = (q_0, \triangleright)$ kann durch einen SK D_0 berechnet werden, der aus k 0- bzw 1-Gattern besteht
- Für $i \in \{1, \ldots, n\}$ kann $z_i = (q_0, x_i)$ durch einen SK D_i berechnet werden, der aus dem Eingabegatter x_i sowie weiteren 0- bzw 1-Gattern besteht
- Wegen (**) gibt es einen SK D konstanter Größe, der für jedes $i > n$ bei Eingabe von z_{i-1} und $z_{prev(i)}$ den Schnappschuss z_i berechnet

- Außerdem gibt es einen SK D' konstanter Größe, der an seinem Ausgabegatter angibt, ob der Schnappschuss $z_{T(n)}$ den akzeptierenden Zustand q_{accept} enthält.

Skizze:



Der SK C_n wird aus den Sken $D_0, D_1, \dots, D_n, D'$ und $(T(n)-n)$ Sken der Form D zusammengesetzt. Wegen (***) hängt die "Verkabelung" von C_n nur von n ab (und nicht von der konkreten Eingabe x).

Insgesamt gilt:

- $|C_n| = O(T(n)) = O(n^c)$ und
- f.a. $x \in \{0,1\}^n$ ist

  $C_n(x) = 1 \iff M$ akzeptiert $x \iff x \in L$

Somit ist $(C_n)_{n \ge 0}$ eine SK-Familie polynomieller Größe, die L berechnet. Dh: $L \in P/poly$.

□

Bemerkung 6.14

Der im Beweis von Theorem 6.13 konstruierte SK C_n kann sogar von einer TM konstruiert werden — und zwar in Zeit poly(n) und auf Platz $O(\log n)$ (im Sinne von write-once logspace-Berechnungen).

Die stereotype TM M kann nämlich so gewählt werden, dass die Kopfposition zum Zeitpunkt i bei Eingabe eines Worts der Länge n auf Platz $O(\log n)$ berechnet werden kann.

Mit dieser Beweismethode lassen sich einige weitere Sätze zeigen — u.a. Folgendes:

Folgerung 6.15

(a) Das Problem

$$\text{CIRCUIT-EVAL} := \{ \langle C, x \rangle : C \text{ ist ein SK mit } n \text{ Eingabegattern}, n \in \mathbb{N}, x \in \{0,1\}^n, C(x) = 1 \}$$

ist P-vollständig im folgenden Sinn:

Ein Problem $L' \subseteq \{0,1\}^*$ ist P-vollständig, falls gilt:

- $L' \in P$ und
- L' ist P-hart, d.h. f.a. $L \in P$ gilt: $L \leq_e L'$

(Zur Erinnerung: $\leq_e$ bezeichnet logspace-Reduzierbarkeit)

(b) Das Problem

$$\text{CIRCUIT-SAT} := \{ C : C \text{ ist ein SK mit } n \text{ Eingabegattern } n \in \mathbb{N}, \text{ es gibt ein } x \in \{0,1\}^n \text{ s.d. } C(x) = 1 \}$$

ist NP-vollständig.

Beweis:
Man sieht leicht, dass CIRCUIT-EVAL in P und CIRCUIT-SAT in NP liegt.

(a) Die P-Härte von CIRCUIT-EVAL lässt sich aus dem Beweis von Theorem 6.13 und Bemerkung 6.14 folgern:

Für $L \in P$ erhalten wir $L \leq_e$ CIRCUIT-EVAL durch die Reduktion f, die ein $x \in \{0,1\}^*$ abbildet auf $f(x) := \langle C_{|x|}, x \rangle$, wobei $C_{|x|}$ der im Beweis von Theorem 6.13 konstruierte SK C_n für $n := |x|$ ist.
Es gilt: $x \in L \iff C_n(x) = 1 \iff \underbrace{\langle C_{|x|}, x \rangle}_{= f(x)} \in$ CIRCUIT-EVAL

Gemäß Bemerkung 6.14 ist f write-once logspace-berechenbar.

(b) Die NP-Härte von CIRCUIT-SAT erhalten wir wie folgt:
Sei $L \in NP$. Gemäß Definition 2.1 gibt es ein Polynom p und eine det. Polynomialzeit-TM M, s.d. f.a. $x \in \{0,1\}^*$ gilt: $x \in L \iff$ ex. $u \in \{0,1\}^{p(|x|)}$ s.d. $M(\langle x,u \rangle) = 1$.
Sei C_m der im Beweis von Theorem 6.13 für M konstruierte SK, mit $m = |\langle x,u \rangle|$, für den gilt:
$C_m(\langle x,u \rangle) = 1 \iff M(\langle x,u \rangle) = 1$.

Aus C_m und x lässt sich leicht ein SK C_x konstruieren, für den gilt:

$C_x \in$ CIRCUIT-SAT $\iff$ es $u \in \{0,1\}^{p(|x|)}$: $C_m(\langle x, u\rangle) = 1$ $\iff$ $x \in L$

(Details: Übung!)

Insgesamt liefert die Abbildung f mit $f(x) := C_x$ eine Polynomialzeit-Reduktion (sogar eine logspace-Reduktion) von L auf CIRCUIT-SAT. □

Theorem 6.13 besagt, dass $P \subseteq P/poly$ ist.

Frage: Gilt auch: $NP \subseteq P/poly$?

Antwort: Nein — es sei denn, die Polynomialzeit-Hierarchie kollabiert. Genauer:

Theorem 6.16 (Satz von Karp und Lipton, 1980)

Falls $NP \subseteq P/poly$, so ist $PH = \Sigma_2^P$

Beweis:

Wegen Satz 5.4 genügt es zu zeigen, dass $\Pi_2^P \subseteq \Sigma_2^P$.

Dazu genügt es, zu zeigen, dass das Π_2^P-vollständige Problem Π_2-SAT (vgl. Beobachtung 5.8) in Σ_2^P liegt.

Falls $NP \subseteq P/poly$, so gilt insbesondere: SAT $\in P/poly$.

D.h.: Es gibt eine SK-Familie $(C_n)_{n \in \mathbb{N}}$ polynomieller Größe, s.d. für C_n bei Eingabe einer aussagenlogischen Formel φ der Länge n gilt:

$C_n(\varphi) = 1$ $\iff$ φ ist erfüllbar.

Unter Verwendung des im Beweis von Satz 2.13 genutzten Algorithmus lässt sich aus $(C_n)_{n\in\mathbb{N}}$ eine SK-Familie $(C'_n)_{n\in\mathbb{N}}$ polynomieller Größe konstruieren, für die folgendes gilt:

⊛: $C'_n(\varphi)$ gibt an, ob φ erfüllbar ist. Und es gibt Gatter $z_1 \dots z_n$ in C'_n, s.d. gilt: Falls $C'_n(\varphi) = 1$, so gibt der Wert der Gatter $z_1 \dots z_n$ eine erfüllende Belegung von φ an.

Da $(C'_n)_{n\in\mathbb{N}}$ polynomielle Größe hat, gibt es ein Polynom $p: \mathbb{N} \to \mathbb{N}$ s.d. C'_n durch einen Bitstring der Länge $p(n)$ beschrieben werden kann.

Wir wollen nun ⊛ nutzen, um zu zeigen, dass $\Pi_2\text{-SAT} \in \Sigma_2^P$:

Zur Erinnerung:

$\Pi_2\text{-SAT} = \{ \Phi : \Phi$ ist eine wahre QBF der Form $\forall \bar{u}\, \exists \bar{v}\, \varphi$, wobei $\bar{u}$ und $\bar{v}$ Listen von Variablen sind und φ eine aussagenlogische Formel über den Variablen $\bar{u}, \bar{v}$ ist $\}$

Sei also $\Phi := \forall \bar{u}\, \exists \bar{v}\, \varphi$ eine Eingabe für $\Pi_2\text{-SAT}$.

Sei $\bar{u} = u_1 \dots u_\ell$, $\bar{v} = v_1 \dots v_m$.

Für jede Belegung $\bar{y} = y_1 \dots y_\ell \in \{0,1\}^\ell$ für $\bar{u}$ sei $\varphi_{\bar{y}}$ die aussagenlogische Formel, die aus φ entsteht, indem die Variablen $u_1 \dots u_\ell$ durch die Werte $y_1 \dots y_\ell$ ersetzt werden.

Sei $n := |\varphi_{\bar{y}}|$ die Länge der Formel $\varphi_{\bar{y}}$

(Beachte: n hängt nicht von der konkreten Wahl von $\bar{y}$ ab)

Für den SK C'_n aus (*) gilt dann:

(**) Φ ist wahr
$\Leftrightarrow \forall \bar{y} \in \{0,1\}^e$ gilt:
$C'_n(\varphi_{\bar{y}}) = 1$, und der Wert der Gatter $z_1, \dots, z_n$ gibt eine erfüllende Belegung für $\varphi_{\bar{y}}$ an.

Es folgt:

Φ ist wahr
$\Leftrightarrow \exists \gamma \in \{0,1\}^{p(n)} \quad \forall \bar{y} \in \{0,1\}^e$:

(***) γ beschreibt einen SK C' mit n Eingabegattern, einem Ausgabegatter und n speziellen Gattern $z_1, \dots, z_n$, s.d. gilt:
$C'(\varphi_{\bar{y}}) = 1$, und der Wert der Gatter $z_1, \dots, z_n$ gibt eine erfüllende Belegung für $\varphi_{\bar{y}}$ an.

("$\Rightarrow$" gilt wegen (**) und da C'_n durch einen Bitstring der Länge $p(n)$ repräsentiert werden kann;
"$\Leftarrow$" gilt offensichtlicherweise)

Man sieht leicht, dass (***) von einer det. Polyn.zeit-TM überprüft werden kann. Somit ist Π_2-SAT $\in \Sigma_2^P$.

□

Eine TM-basierte Charakterisierung der Klasse P/poly

Definition 6.17 (TM mit "Advice-Strings")

Sei $(\alpha_n)_{n\in\mathbb{N}}$ eine Folge von Bitstrings $\alpha_n \in \{0,1\}^*$.
Sei M eine det. TM und sei $L \subseteq \{0,1\}^*$
Wir sagen: M entscheidet L mit Advice $(\alpha_n)_{n\in\mathbb{N}}$,
falls f.a. $n\in\mathbb{N}$ und f.a. $x\in\{0,1\}^n$ gilt:

$$x \in L \iff M(\langle x, \alpha_n\rangle) = 1$$

(Dh: α_n kann als eine Liste von "Instruktionen" aufgefasst werden, die M bei Eingabe eines Worts x der Länge n befolgt).

Definition 6.18 (Die Klasse $DTIME(T(n))/a(n)$)

Seien $T:\mathbb{N}\to\mathbb{N}$ und $a:\mathbb{N}\to\mathbb{N}$. Die Klasse

$$DTIME(T(n))/a(n)$$

besteht aus allen Sprachen $L \subseteq \{0,1\}^*$, für die gilt:
Es gibt eine Folge $(\alpha_n)_{n\in\mathbb{N}}$ von Bitstrings $\alpha_n \in \{0,1\}^{a(n)}$ und eine det TM M, so dass M die Sprache L mit Advice $(\alpha_n)_{n\in\mathbb{N}}$ entscheidet, und bei Eingabe von $\langle x, \alpha_n\rangle$ nur $O(T(n))$ Schritte macht (für $n\in\mathbb{N}$, $x\in\{0,1\}^n$).

Beispiel 6.19

Jede unäre Sprache $L \subseteq \{1\}^*$ liegt in $DTIME(n)/1$, denn:

Wähle $d_n := \begin{cases} 1 & \text{falls } 1^n \in L \\ 0 & \text{falls } 1^n \notin L \end{cases}$

Sei M eine Linearzeit-TM, die bei Eingabe von $\langle x, d_n \rangle$ für $x \in \{0,1\}^n$ genau dann 1 ausgibt, wenn x nur aus 1en besteht und $d_n = 1$ ist.

Theorem 6.20 (TM-basierte Charakterisierung von P/poly)

$$P/poly = \bigcup_{\substack{c \geq 1, \\ d \geq 1}} DTIME(n^c)/n^d$$

D.h.: P/poly ist die Klasse aller Sprachen, die von det. Polynomialzeit-TMen mit polynomiell langen Advice-Strings entschieden werden.

Beweis:

"$\subseteq$": Sei $L \in P/poly$ und sei $(C_n)_{n \in \mathbb{N}}$ eine SK-Familie polynomieller Größe, die L berechnet. Für jedes $n \in \mathbb{N}$ sei d_n ein Bitstring, der den SK C_n repräsentiert. Sei M eine det. TM, die bei Eingabe von $\langle x, d_n \rangle$ (für $x \in \{0,1\}^n$, $n \in \mathbb{N}$) den SK C_n bei Eingabe x auswertet.

Klar: Das geht in polyn. Zeit. Somit ist $L \in \bigcup_{\substack{c,d \\ \geq 1}} DTIME(n^c)/n^d$

"$\supseteq$": Seien $c, d \geq 1$ und sei $L \in \mathrm{DTIME}(n^c)/n^1$. Sei $(a_n)_{n\in\mathbb{N}}$ mit $a_n \in \{0,1\}^{n^d}$ f.a. $n \in \mathbb{N}$, und sei M eine det. TM, die L mit Advice $(a_n)_{n\in\mathbb{N}}$ entscheidet und f.a. $n \in \mathbb{N}$, $x \in \{0,1\}^n$ bei Eingabe $\langle x, a_n\rangle$ nach $O(n^c)$ Schritten anhält.

Sei $(C_m)_{m\in\mathbb{N}}$ die im Beweis von Theorem 6.13 konstruierte SK-Familie, s.d. f.a. $n \in \mathbb{N}$, $x \in \{0,1\}^n$, $m := |\langle x, a_n\rangle|$ gilt:

$$C_m(\langle x, a_n\rangle) = 1 \iff M(\langle x, a_n\rangle) = 1.$$

C_m lässt sich unter Verwendung von a_n leicht zu einem SK C'_n umbauen, der bei Eingabe von x dasselbe tut wie C_m bei Eingabe $\langle x, a_n\rangle$ (Details: Übung!) Somit ist $(C'_n)_{n\in\mathbb{N}}$ eine SK-Familie der Größe $\mathrm{poly}(m) = \mathrm{poly}(n)$, die L berechnet. D.h.: $L \in P/\mathrm{poly}$. □

6.5 Uniforme Schaltkreis-Familien

Oft betrachtet man nur sog. uniforme SK-Familien $(C_n)_{n\in\mathbb{N}}$, für die es einen effizienten Algorithmus gibt, der bei Eingabe der Zahl n den SK C_n erzeugt. Insbesondere können uniforme SK-Familien daher nur entscheidbare Sprachen berechnen.

Definition 6.21

(a) Eine SK-Familie $(C_n)_{n\in\mathbb{N}}$ heißt P-uniform, falls es eine det. Polynomialzeit-TM gibt, die bei Eingabe eines Worts der Länge n (für $n\in\mathbb{N}$) den SK C_n erzeugt.

(b) Eine SK-Familie $(C_n)_{n\in\mathbb{N}}$ heißt Logspace-uniform, falls die Funktion $f: \{0,1\}^* \to \{0,1\}^*$ mit
$f(x) := C_{|x|}$ (f.a. $x\in\{0,1\}^*$)
write-once logspace-berechenbar ist.

Bemerkung 6.22

(a) Wegen $L \subseteq P$ ist jede logspace-uniforme SK-Familie auch P-uniform.

(b) Wegen Fakt 4.21 ist $(C_n)_{n\in\mathbb{N}}$ genau dann logspace-uniform, wenn jede der folgenden Funktionen auf Platz $O(\log n)$ berechnet werden kann:

- $\text{SIZE}(n) := |C_n|$
- $\text{TYPE}(n,i) :=$ die Markierung (aus $\{x_1,\neg x_1,\dots,x_n,\neg x_n,0,1,\wedge,\vee,?\}$) des i-ten Gatters von C_n
- $\text{EDGE}(n,i,j) :=$ 1 falls es in C_n eine Kante vom i-ten zum j-ten Gatter gibt

(Konvention: Das 0-te Gatter ist das Ausgabegatter)

(c) Für jede P-uniforme SK-Familie $(C_n)_{n\in\mathbb{N}}$ gibt es offensichtlicherweise ein $c\geq 1$ s.d. $|C_n| \leq n^c$ ist (f.a. $n\in\mathbb{N}$)

Definition 6.23

Sei $\mathcal{U} \in \{P, \text{logspace}\}$. Die Klasse

$\mathcal{U}$-uniformes P/poly

besteht aus allen Sprachen $L \subseteq \{0,1\}^*$, die von einer $\mathcal{U}$-uniformen SK-Familie polynomieller Größe berechnet werden

Satz 6.24

logspace-uniformes P/poly $\overset{①}{=}$ P-uniformes P/poly $\overset{②}{=}$ P.

Beweis:

"①, $\subseteq$": Folgt aus Bemerkung 6.22 (a).

"②, $\subseteq$": Sei $(C_n)_{n \in \mathbb{N}}$ eine P-uniforme SK-Familie, die eine Sprache L berechnet

Bei Eingabe von $x \in \{0,1\}^*$ kann eine det. Polynomialzeit-TM wie folgt vorgehen:

(1) Erzeuge den SK C_n für $n := |x|$

(2) Werte C_n mit Eingabe x aus.

Somit ist $L \in P$

"$P \subseteq$ logspace-uniformes P/poly":

Sei $L \in P$.

Aus dem Beweis von Theorem 6.13 erhalten wir für jedes $n \in \mathbb{N}$ einen SK C_n der Größe poly(n), sd. f.a. $x \in \{0,1\}^*$ gilt:

$$x \in L \iff C_n(x) = 1.$$

Gemäß Bemerkung 6.14 ist die SK-Familie $(C_n)_{n\in\mathbb{N}}$ logspace-uniform. □

Bemerkung 6.25 (Nachtrag zu Kapitel 5)

Auf ähnliche Art können wir Satz 5.13 (d): $\boxed{\mathrm{DTIME}(T(n)) \subseteq \mathrm{ASPACE}(\log T(n))}$ (für zeitkonstruierbares $T: \mathbb{N} \to \mathbb{N}$ mit $T(n) \geq n$) beweisen:

Sei $L \in \mathrm{DTIME}(T(n))$ und sei M eine stereotype TM, die L in Zeit $O(T(n)^2)$ entscheidet (vgl. Bemerkung 1.10).

Die Konstruktion aus dem Beweis von Theorem 6.13 liefert eine SK-Familie $(C_n)_{n\in\mathbb{N}}$ der Größe $\mathrm{poly}(T(n))$, s.d. f.a. $n \in \mathbb{N}$ und alle $x \in \{0,1\}^n$ gilt:

$$C_n(x) = 1 \iff M(x) = 1 \iff x \in L.$$

Und gemäß Bemerkung 6.14 kann C_n bei Eingabe von n auf Platz $O(\log T(n))$ berechnet werden (im Sinne von write-once logspace-Berechenbarkeit).

Man kann sich leicht davon überzeugen, dass C_n bei Eingabe von $x \in \{0,1\}^n$ von einer alternierenden TM ausgewertet werden kann, die Platz $O(\log T(n))$ benutzt: $\wedge$- bzw $\vee$-Gatter von C_n werden durch Zustände der TM ausgewertet, die mit "$\forall$" bzw mit "$\exists$" markiert sind. Details: Übung.

Insgesamt erhält man damit: $L \in \mathrm{ASPACE}(\log T(n))$.

□

6.6 Die Klassen NC^i, AC^i und NC als Modelle für effiziente parallele Berechnungen

Die Tiefe eines Schaltkreises C ist die Länge eines längsten gerichteten Weges in C.
Offensichtlicherweise kann für einen SK C der Größe s und Tiefe t der Wert $C(x)$ bei Eingabe eines Wortes $x \in \{0,1\}^n$ von s parallel arbeitenden Prozessoren in t Berechnungsschritten ermittelt werden.

Definition 6.26 (NC^i und NC)

(a) Sei $i \in \mathbb{N}$. Die Klasse

NC^i

besteht aus allen Sprachen $L \subseteq \{0,1\}^*$, die von einer SK-Familie $(C_n)_{n\in\mathbb{N}}$ der Größe $poly(n)$ und der Tiefe $O((\log n)^i)$ berechnet werden können.
(D.h.: $(C_n)_{n\in\mathbb{N}}$ berechnet L, und es gibt Konstanten $c,d \geq 1$ s.d. f.a. $n\in\mathbb{N}$ gilt: $|C_n| \leq n^c$ und die Tiefe von C_n ist $\leq d\cdot(\log n)^i$).

(b) $NC := \bigcup_{i\in\mathbb{N}} NC^i$ ist die Klasse aller Sprachen, die von einer SK-Familie polynomieller Größe und polylogarithmischer Tiefe berechnet werden können.

Die Bezeichnung "NC" steht für "Nick's class", benannt nach dem Komplexitätstheoretiker Nick Pippenger (diese Bezeichnung wurde von Stephen Cook geprägt). Klar: $NC \subseteq P/poly$.

Bemerkung 6.27

Für jedes sinnvolle Modell von Parallelrechnern lässt sich eine Variante der folgenden Aussage beweisen:

$L \in$ logspace-uniformes NC

$\Leftrightarrow$ L kann unter Verwendung von poly(n) vielen Prozessoren in Zeit polylog(n) entschieden werden.

Ähnlich wie wir die Klasse P als Modell für "effizient lösbare Probleme" sehen, betrachten wir die Klasse NC als Modell für "effizient parallelisierbare Probleme". Aus Satz 6.24 folgt: logspace-uniformes $NC \subseteq P$.

Beispiel 6.28

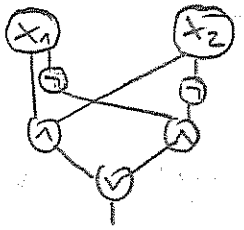
Für die Sprache

$PARITY := \{ x \in \{0,1\}^* : $ die Anzahl der 1en in x ist gerade $\}$

gilt: $PARITY \in NC^1$

Denn: Sei $(C_n)_{n \in \mathbb{N}}$ die SK-Familie mit

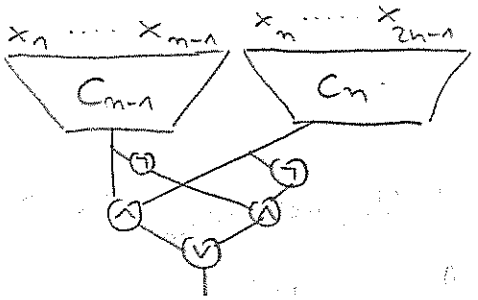
C_1: 

C_2: 

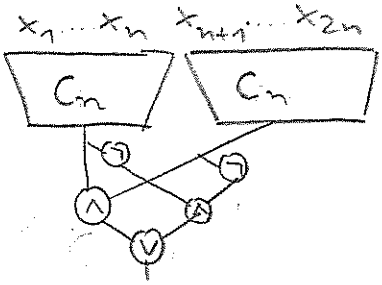
und

f.a. $n \geq 2$:

C_{2n-1}:

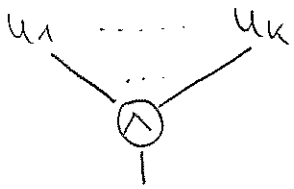


C_{2n}:



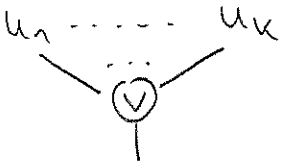
Dann gilt: $(C_n)_{n \in \mathbb{N}}$ ist eine SK-Familie der Größe $O(n)$ und Tiefe $O(\log n)$, die PARITY berechnet. □ Bsp 6.28

Manchmal betrachtet man auch Schaltkreise, bei denen ∧- und ∨-Gatter von unbeschränktem fan-in erlaubt sind. Ein ∧-Gatter der Form



erhält genau dann den Wert 1, wenn $u_1 = \cdots = u_k = 1$ ist.

Ein ∨-Gatter der Form



erhält genau dann den Wert 1, wenn mindestens eins der $u_1, \ldots, u_k$ den Wert 1 hat.

Wir sprechen im Folgenden von <u>uSKs</u>, um Schaltkreise zu bezeichnen, die zusätzlich zu den bisher betrachteten Gattern auch ∧-Gatter und ∨-Gatter von <u>unbeschränktem</u> fan-in benützen dürfen.

Definition 6.29 (AC^i)

(a) Sei $i \in \mathbb{N}$. Die Klasse

$$AC^i$$

besteht aus allen Sprachen $L \subseteq \{0,1\}^*$, die von einer uSK-Familie $(C_n)_{n\in\mathbb{N}}$ der Größe $poly(n)$ und der Tiefe $O((\log n)^i)$ berechnet werden können.

(b) $$AC := \bigcup_{i\in\mathbb{N}} AC^i$$

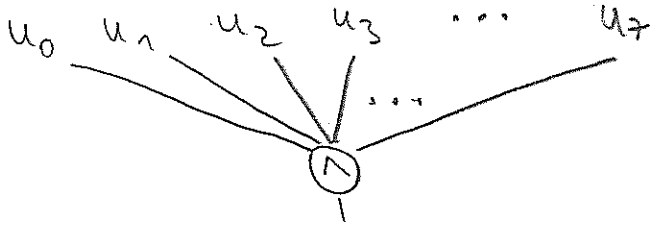
(Der Buchstabe 'A' bei "AC" steht für "alternierend".)

Fakt 6.30

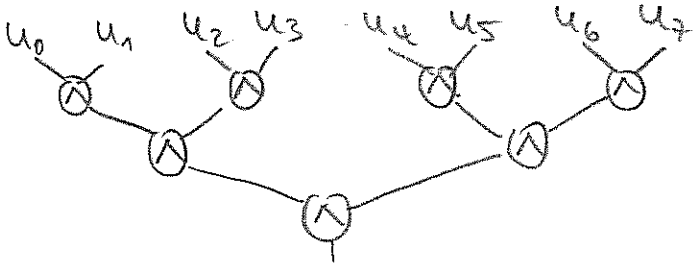
(a) F.a. $i \geq 0$ gilt: $NC^i \subseteq AC^i$ (klar per Definition)

(b) F.a. $i \geq 0$ gilt: $AC^i \subseteq NC^{i+1}$, denn:

Ein $\wedge$-Gatter vom fan-in 8



kann durch einen SK der Größe 7 und Tiefe 3 der folgenden Form ersetzt werden:



Allgemein kann ein ∧-Gatter (bzw. ein ∨-Gatter) vom fan-in $\leq s$ durch einen (herkömmlichen) SK der Größe $\leq s$ und Tiefe $\leq \lceil \log s \rceil$ ersetzt werden.

Daher kann ein uSK der Größe s und Tiefe t durch einen SK der Größe $\leq s \cdot \lceil \log s \rceil$ und Tiefe $\leq t \cdot \lceil \log s \rceil$ simuliert werden.

Daraus folgt insbes.: $AC^i \subseteq NC^{i+1}$.

(c) $AC = NC$ (folgt direkt aus (a) und (b)).

Satz 6.31

Für die Sprache PATH $:=$ $\{ \langle G, s, t \rangle :$ G ist ein endlicher gerichteter Graph, in dem es einen Weg von Knoten s zu Knoten t gibt

gilt: PATH $\in$ logspace-uniformes AC^1

Beweis:

Wir konstruieren eine uSK-Familie, die das iterierte Boolesche Produkt der Adjazenzmatrix von G berechnet.

Details: Übung!

□

Satz 6.32

NL $\subseteq$ logspace-uniformes AC^1

Beweis:

Sei $L \in NL$ und sei M eine $O(\log n)$ platzbeschränkte NDTM, die L entscheidet. F.a. $x \in \{0,1\}^*$ gilt:

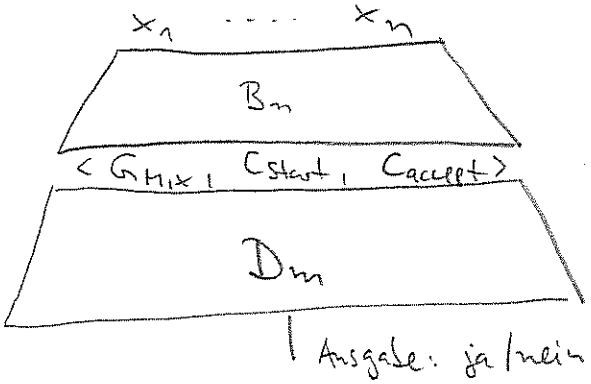
$x \in L \iff \langle G_{M,x}, C_{start}, C_{accept} \rangle \in PATH$,

wobei $G_{M,x}$ der Konfigurationsgraph von M bei Eingabe x ist.

Wir konstruieren eine uSK-Familie $(C_n)_{n \in \mathbb{N}}$, so dass C_n bei Eingabe von $x \in \{0,1\}^n$

- zunächst aus dem Bitstring x den Bitstring $\langle G_{M,x}, C_{start}, C_{accept} \rangle$ erzeugt (durch einen uSK B_n) und
- dann diesen mit dem uSK D_m aus Satz 6.31 verarbeitet, der testet, ob $\langle G_{M,x}, C_{start}, C_{accept} \rangle \in PATH$ ist.

Skizze:



Hier ist D_m der uSK der Größe poly(m) und Tiefe $O(\log m)$ aus Satz 6.31, der testet ob ein Bitstring $\langle G_{M,x}, C_{start}, C_{accept} \rangle$ der Länge m zur Sprache PATH gehört.

Beachte: Hier ist $m \leq n^c$, für eine Konstante c,
da $G_{M,x}$ höchstens $2^{O(\log n)}$ Knoten hat
(denn M ist $O(\log n)$ platzbeschränkt).
Somit hat D_m Größe poly(n) und Tiefe $O(\log n)$.

Daher sind wir fertig, sobald wir einen uSk B_n
der Größe poly(n) und Tiefe $O(\log n)$ konstruiert haben,
der bei Eingabe von $x \in \{0,1\}^n$ einen Bitstring der
Form $\langle G_{M,x}, C_{start}, C_{accept} \rangle$ erzeugt.

Wir repräsentieren jeden Knoten von $G_{M,x}$ durch einen
Bitstring der Länge $d \cdot \log n$, bei dem die ersten $\log n$ Bits
die aktuelle Kopfposition auf dem Eingabeband bezeichnet
und die restlichen Bits die Beschriftung und die
Kopfpositionen auf den Arbeitsbändern beschreiben.
Jeden Bitstring der Länge $d \cdot \log n$ identifizieren wir mit
einer Zahl aus $\{0, 1, \ldots, n^d - 1\}$. D.h.: Zahlen
$i \in \{0, 1, \ldots, n^d - 1\}$ repräsentieren Konfigurationen von M bei
Eingabe x.
Der Eintrag A_{ij} in Zeile i und Spalte j der
Adjazenzmatrix von $G_{M,x}$ hängt nur ab von
i, j, den Überführungsfunktionen δ_0 und δ_1 von M
und dem Eintrag von x an der Kopfposition $p(i)$
des Eingabebandes, die zur Konfiguration i gehört.

D.h: A_{ij} hängt nur von einem einzigen Bit der
Eingabe x ab, nämlich von $x_{p(i)}$.

Daher kann B_n sogar durch einen Sk der Größe

poly(n) und konstanter Tiefe erzeugt werden.

Durch Zusammensetzen von B_n und D_m erhalten wir einen uSK C_n der Größe poly(n) und Tiefe $O(\log n)$, der bei Eingabe von $x \in \{0,1\}^n$ testet, ob $x \in L$ ist.

Somit ist $NL \subseteq AC^1$.

Man kann sich leicht davon überzeugen, dass die konstruierte uSK-Familie $(C_n)_{n \in \mathbb{N}}$ logspace-uniform ist. □

Folgerung 6.33

(a) Für jedes $i \geq 1$ ist AC^i abgeschlossen unter logspace-Reduktionen
(d.h: Ist $A \leq_\ell B$ und $B \in AC^i$, so auch $A \in AC^i$)

(b) Für jedes $i \geq 2$ ist NC^i abgeschlossen unter logspace-Reduktionen

(c) Falls es ein P-vollständiges Problem L mit $L \in NC$ gibt, so ist $P \subseteq NC$.

Insbesondere gilt:
Falls CIRCUIT-EVAL $\in NC$, so $P \subseteq NC$.

Beweis:

(a) Nutze, dass gemäß Satz 6.32 gilt: $L \subseteq NL \subseteq AC^1$ und betrachte eine implizit logspace-berechenbare Funktion f, die A auf B reduziert.
Details: Übung!

(b) Analog zu (a)

(c) Folgt leicht aus (b), da $NC = \bigcup_{i \geq 0} NC^i$.
Zur Erinnerung: Von Folgerung 6.15 wissen wir, dass CIRCUIT-EVAL P-vollständig ist. □

Bemerkung 6.34

Aus Beispiel 6.28 wissen wir, dass PARITY $\in NC^1$ ist.

(Zur Erinnerung: $\text{PARITY} = \{x \in \{0,1\}^* : \text{die Anzahl der 1en in } x \text{ ist gerade}\}$.)

Man kann leicht zeigen, dass PARITY $\notin NC^0$ (Übung!).

Viel schwerer zu beweisen ist der folgende Satz:

Theorem 6.35 (Furst, Saxe, Sipser 1981; Ajtai 1983)

PARITY $\notin AC^0$

(Hier ohne Beweis)