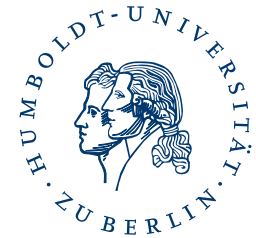
Einführung in die Theoretische Informatik

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2017/18

Zeitkomplexität von Turingmaschinen

Die Laufzeit einer NTM M bei Eingabe x ist die maximale Anzahl an Rechenschritten, die $M(x)$ ausführt.

Definition

- Die Laufzeit einer NTM M bei Eingabe x ist definiert als

$$time_M(x) = \sup\{t \geq 0 \mid \exists K : K_x \vdash^t K\},$$

  wobei $\sup \mathbb{N} = \infty$ ist.
- Sei $t : \mathbb{N} \to \mathbb{N}$ eine monoton wachsende Funktion.
- Dann ist M $t(n)$-zeitbeschränkt, falls für alle Eingaben x gilt:

$$time_M(x) \leq t(|x|).$$

Die Zeitschranke $t(n)$ beschränkt also die Laufzeit bei allen Eingaben der Länge n (worst-case Komplexität).

Zeitkomplexitätsklassen

Wir fassen alle Sprachen und Funktionen, die in einer vorgegebenen Zeitschranke $t(n)$ entscheidbar bzw. berechenbar sind, in folgenden Komplexitätsklassen zusammen.

Definition

- Die in deterministischer Zeit $t(n)$ entscheidbaren Sprachen bilden die Sprachklasse

$$\mathrm{DTIME}(t(n)) = \{L(M) \mid M \text{ ist eine } t(n)\text{-zeitbeschränkte DTM}\}.$$

- Die in nichtdeterministischer Zeit $t(n)$ entscheidbaren Sprachen bilden die Sprachklasse

$$\mathrm{NTIME}(t(n)) = \{L(M) \mid M \text{ ist eine } t(n)\text{-zeitbeschränkte NTM}\}.$$

- Die in deterministischer Zeit $t(n)$ berechenbaren Funktionen bilden die Funktionenklasse

$$\mathrm{FTIME}(t(n)) = \left\{ f \;\middle|\; \begin{array}{l} \text{es gibt eine } t(n)\text{-zeitbeschränkte} \\ \text{DTM } M, \text{ die } f \text{ berechnet} \end{array} \right\}.$$

Die wichtigsten Zeitkomplexitätsklassen

- Die wichtigsten deterministischen Zeitkomplexitätsklassen sind

$$\mathrm{LINTIME} = \bigcup_{c\geq 1} \mathrm{DTIME}(cn+c) \qquad \text{„Linearzeit“}$$

$$\mathrm{P} = \bigcup_{c\geq 1} \mathrm{DTIME}(n^c+c) \qquad \text{„Polynomialzeit“}$$

$$\mathrm{E} = \bigcup_{c\geq 1} \mathrm{DTIME}(2^{cn+c}) \qquad \text{„Lineare Exponentialzeit“}$$

$$\mathrm{EXP} = \bigcup_{c\geq 1} \mathrm{DTIME}(2^{n^c+c}) \qquad \text{„Exponentialzeit“}$$

- Die nichtdeterministischen Klassen NLINTIME, NP, NE, NEXP und die Funktionenklassen FLINTIME, FP, FE, FEXP sind analog definiert.
- Für eine Klasse $\mathcal{F}$ von Funktionen sei $\mathrm{DTIME}(\mathcal{F}) = \bigcup_{t\in\mathcal{F}} \mathrm{DTIME}(t(n))$ (die Klassen $\mathrm{NTIME}(\mathcal{F})$ und $\mathrm{FTIME}(\mathcal{F})$ sind analog definiert).

Asymptotische Laufzeit und Landau-Notation

Definition

Seien f und g Funktionen von $\mathbb{N}$ nach $\mathbb{R}^+ \cup \{0\} = [0, \infty)$.

- Wir schreiben $f(n) = \mathcal{O}(g(n))$, falls es Zahlen n_0 und c gibt mit

  $$\forall n \geq n_0 : f(n) \leq c \cdot g(n).$$

  Bedeutung: „f wächst nicht wesentlich schneller als g."

- Formal bezeichnet der Term $\mathcal{O}(g(n))$ die Klasse aller Funktionen f, die obige Bedingung erfüllen, d.h.

  $$\mathcal{O}(g(n)) = \{f : \mathbb{N} \to [0, \infty) \mid \exists n_0, c \in \mathbb{N}\, \forall n \geq n_0 : f(n) \leq c \cdot g(n)\}.$$

- Die Gleichung $f(n) = \mathcal{O}(g(n))$ drückt also in Wahrheit eine Element-Beziehung $f \in \mathcal{O}(g(n))$ aus.
- O-Terme können auch auf der linken Seite vorkommen. In diesem Fall wird eine Inklusionsbeziehung ausgedrückt.
- So steht $n^2 + \mathcal{O}(n) = \mathcal{O}(n^2)$ für die Aussage

  $$\{n^2 + f \mid f \in \mathcal{O}(n)\} \subseteq \mathcal{O}(n^2).$$

Asymptotische Laufzeit und Landau-Notation

Beispiel

- $7\log(n) + n^3 = \mathcal{O}(n^3)$ ist richtig.
- $7\log(n)n^3 = \mathcal{O}(n^3)$ ist falsch.
- $2^{n+\mathcal{O}(1)} = \mathcal{O}(2^n)$ ist richtig.
- $2^{\mathcal{O}(n)} = \mathcal{O}(2^n)$ ist falsch (siehe Übungen).

Mit der O-Notation lassen sich die wichtigsten deterministischen Zeitkomplexitätsklassen wie folgt charakterisieren:

$$\begin{aligned} \text{LINTIME} &= \text{DTIME}(\mathcal{O}(n)) && \text{„Linearzeit“} \\ \text{P} &= \text{DTIME}(n^{\mathcal{O}(1)}) && \text{„Polynomialzeit“} \\ \text{E} &= \text{DTIME}(2^{\mathcal{O}(n)}) && \text{„Lineare Exponentialzeit“} \\ \text{EXP} &= \text{DTIME}(2^{n^{\mathcal{O}(1)}}) && \text{„Exponentialzeit“} \end{aligned}$$

Das P-NP-Problem

- Wie wir gesehen haben, sind NTMs nicht mächtiger als DTMs, d.h. jede NTM kann von einer DTM simuliert werden.
- Die Frage, wieviel Zeit eine DTM zur Simulation einer NTM benötigt, ist eines der wichtigsten offenen Probleme der Informatik.
- Wegen $\mathrm{NTIME}(t) \subseteq \mathrm{DTIME}(2^{\mathcal{O}(t)})$ erhöht sich die Laufzeit im schlimmsten Fall exponentiell.
- Insbesondere die Klasse NP enthält viele für die Praxis überaus wichtige Probleme, für die kein Polynomialzeitalgorithmus bekannt ist.
- Für viele dieser Probleme A konnte folgende Implikation gezeigt werden:

  $$A \in \mathrm{P} \Rightarrow \mathrm{P} = \mathrm{NP}$$

- Da jedoch nur Probleme in P als effizient lösbar angesehen werden, hat das P-NP-Problem, also die Frage, ob $\mathrm{NP} = \mathrm{P}$ ist, eine immense praktische Bedeutung.

Die Polynomialzeitreduktion

Definition

- Eine Sprache $A \subseteq \Sigma^*$ ist auf $B \subseteq \Gamma^*$ in Polynomialzeit reduzierbar ($A \leq^p B$), falls eine Funktion $f : \Sigma^* \to \Gamma^*$ in FP existiert mit

$$\forall x \in \Sigma^* : x \in A \Leftrightarrow f(x) \in B.$$

- Eine Sprache A heißt $\leq^p$-hart für eine Sprachklasse $\mathcal{C}$ (kurz: $\mathcal{C}$-hart oder $\mathcal{C}$-schwer), falls gilt:

$$\forall L \in \mathcal{C} : L \leq^p A.$$

- Eine $\mathcal{C}$-harte Sprache A, die zu $\mathcal{C}$ gehört, heißt $\mathcal{C}$-vollständig (bzgl. $\leq^p$).
- NPC bezeichnet die Klasse aller NP-vollständigen Sprachen.

Lemma

- Aus $A \leq^p B$ folgt $A \leq B$.
- Die Reduktionsrelation $\leq^p$ ist reflexiv und transitiv (s. Übungen).

Die Polynomialzeitreduktion

Satz

Die Klassen P und NP sind unter $\leq^p$ abgeschlossen.

Beweis

- Sei $B \in \mathrm{P}$ und gelte $A \leq^p B$ mittels einer Funktion $f \in \mathrm{FP}$.
- Seien M und T DTMs mit $L(M) = B$ und $T(x) = f(x)$.
- Weiter seien p und q polynomielle Zeitschranken für M und T.
- Betrachte die DTM M', die bei Eingabe x zuerst T simuliert, um $f(x)$ zu berechnen, und danach M bei Eingabe $f(x)$ simuliert. Dann gilt

$$x \in A \Leftrightarrow f(x) \in B \Leftrightarrow f(x) \in L(M) \Leftrightarrow x \in L(M').$$

- Also ist $L(M') = A$ und wegen

$$time_{M'}(x) \leq time_T(x) + time_M(f(x)) \leq q(|x|) + p(q(|x|))$$

ist M' polynomiell zeitbeschränkt und somit A in P.

- Der Abschluss von NP unter $\leq^p$ folgt analog. □

NP-Vollständigkeit

Satz

1. $A \leq^p B$ und A ist NP-hart $\Rightarrow$ B ist NP-hart.
2. Falls ein NP-vollständiges Problem A in P enthalten ist, folgt P = NP.

Beweis

1. Da A NP-hart ist, ist jede NP-Sprache L auf A reduzierbar.
   Da zudem $A \leq^p B$ gilt und $\leq^p$ transitiv ist, folgt $L \leq^p B$.
2. Sei A eine NP-vollständige Sprache in P. Dann ist jede NP-Sprache L auf A reduzierbar und da P unter $\leq^p$ abgeschlossen ist, folgt $L \in$ P. □

Platzkomplexität von Turingmaschinen

- Als nächstes definieren wir den Platzverbrauch von NTMs.
- Intuitiv ist dies die Anzahl aller besuchten Bandfelder.
- Wollen wir auch sublinearen Platz sinnvoll definieren, so dürfen wir hierbei das erste Band offensichtlich nicht berücksichtigen.
- Um sicherzustellen, dass eine NTM M das erste Band nur zum Lesen der Eingabe und nicht auch zum Speichern von weiteren Informationen benutzt, verlangen wir, dass M
  - die Felder auf dem Eingabeband nicht verändert und
  - sich höchstens ein Feld von der Eingabe entfernt.

Definition

Eine NTM M heißt offline-NTM (oder NTM mit Eingabeband), falls für jede von M bei Eingabe x erreichbare Konfiguration

$$K = (q, u_1, a_1, v_1, \ldots, u_k, a_k, v_k)$$

gilt, dass $u_1 a_1 v_1$ ein Teilwort von $\sqcup x \sqcup$ ist.

Platzkomplexität von Turingmaschinen

Definition

- Der Platzverbrauch einer offline-NTM M bei Eingabe x ist definiert als

$$space_M(x) = \sup \left\{ s \geq 1 \;\middle|\; \begin{array}{l} \exists K = (q, u_1, a_1, v_1, \ldots, u_k, a_k, v_k) \\ \text{mit } K_x \vdash^* K \text{ und } s = \sum_{i=2}^{k} |u_i a_i v_i| \end{array} \right\}.$$

- Sei $s : \mathbb{N} \to \mathbb{N}$ eine monoton wachsende Funktion.
- M heißt $s(n)$-platzbeschränkt, falls für alle Eingaben x gilt:

$$space_M(x) \leq s(|x|) \text{ und } time_M(x) < \infty.$$

Platzkomplexitätsklassen

Wir fassen alle Sprachen, die in einer vorgegebenen Platzschranke $s(n)$ entscheidbar sind, in folgenden Platzkomplexitätsklassen zusammen.

Definition

- Die auf deterministischem Platz $s(n)$ entscheidbaren Sprachen bilden die Klasse

$$\mathrm{DSPACE}(s(n)) = \{L(M) \mid M \text{ ist eine } s(n)\text{-platzb. offline-DTM}\}.$$

- Die auf nichtdeterministischem Platz $s(n)$ entscheidbaren Sprachen bilden die Klasse

$$\mathrm{NSPACE}(s(n)) = \{L(M) \mid M \text{ ist eine } s(n)\text{-platzb. offline-NTM}\}.$$

Die wichtigsten Platzkomplexitätsklassen

- Die wichtigsten deterministischen Platzkomplexitätsklassen sind

$$\begin{aligned} \mathsf{L} &= \mathrm{DSPACE}(\mathcal{O}(\log n)) && \text{„Logarithmischer Platz“} \\ \mathsf{LINSPACE} &= \mathrm{DSPACE}(\mathcal{O}(n)) && \text{„Linearer Platz“} \\ \mathsf{PSPACE} &= \mathrm{DSPACE}(n^{\mathcal{O}(1)}) && \text{„Polynomieller Platz“} \end{aligned}$$

- Die nichtdeterministischen Klassen NL, NLINSPACE und NPSPACE sind analog definiert.

Elementare Beziehungen zwischen Komplexitätsklassen

Frage

Welche elementaren Beziehungen gelten zwischen den verschiedenen Zeit- und Platzklassen?

Satz

- Für jede Funktion $t(n) \geq n+2$ gilt

  $$\mathrm{DTIME}(t) \subseteq \mathrm{NTIME}(t) \subseteq \mathrm{DSPACE}(\mathcal{O}(t)).$$

- Für jede Funktion $s(n) \geq \log n$ gilt

  $$\mathrm{DSPACE}(s) \subseteq \mathrm{NSPACE}(s) \subseteq \mathrm{DTIME}(2^{\mathcal{O}(s)}) \text{ und}$$

  $$\mathrm{NSPACE}(s) \subseteq \mathrm{DSPACE}(s^2).$$ (Satz von Savitch)

Korollar

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} = \mathrm{NPSPACE} \subseteq \mathrm{EXP} \subseteq \mathrm{NEXP} \subseteq \mathrm{EXPSPACE}.$$

Komplexität der Stufen der Chomsky-Hierarchie

$$\mathrm{REG} = \mathrm{DSPACE}(\mathcal{O}(1)) = \mathrm{NSPACE}(\mathcal{O}(1)) \subsetneq \mathrm{L},$$

$$\mathrm{DCFL} \subsetneq \mathrm{LINTIME},$$

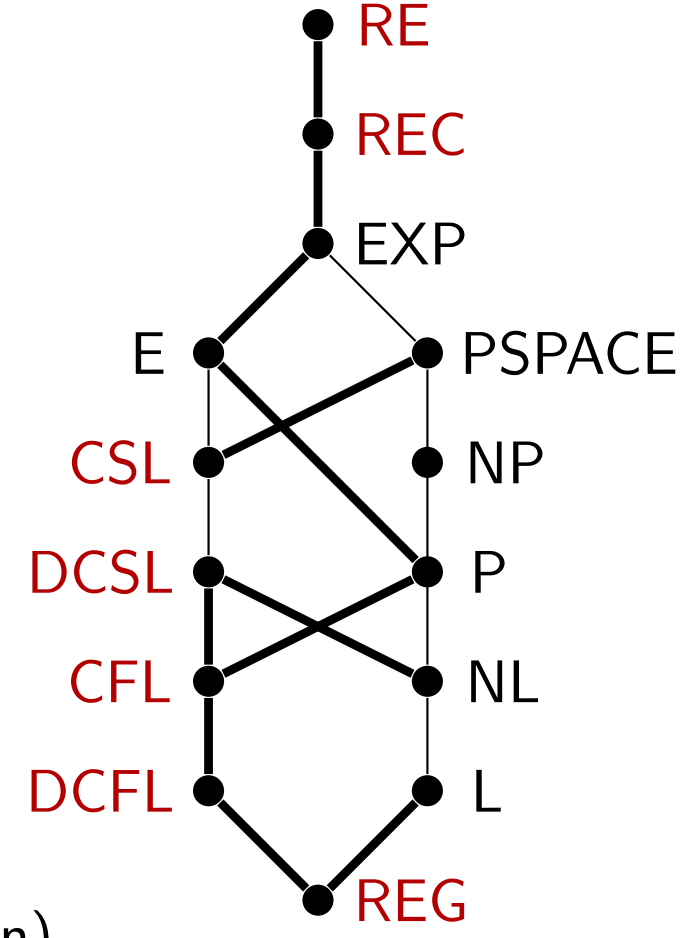
$$\mathrm{CFL} \subsetneq \mathrm{NLINTIME} \cap \mathrm{DTIME}(\mathcal{O}(n^3)) \subsetneq \mathrm{P},$$

$$\mathrm{DCSL} = \mathrm{LINSPACE} \subseteq \mathrm{CSL},$$

$$\mathrm{CSL} = \mathrm{NLINSPACE} \subseteq \mathrm{PSPACE} \cap \mathrm{E},$$

$$\begin{aligned}\mathrm{REC} &= \bigcup_f \mathrm{DSPACE}(f(n)) \\ &= \bigcup_f \mathrm{NSPACE}(f(n)) \\ &= \bigcup_f \mathrm{DTIME}(f(n)) \\ &= \bigcup_f \mathrm{NTIME}(f(n)),\end{aligned}$$

wobei f alle (oder äquivalent: alle berechenbaren) Funktionen $f : \mathbb{N} \to \mathbb{N}$ durchläuft.



Aussagenlogische Formeln

- Die Menge der booleschen (oder aussagenlogischen) Formeln über den Variablen $x_1, \ldots, x_n$, $n \geq 0$, ist induktiv wie folgt definiert:
  - Die Konstanten 0 und 1 sind boolesche Formeln.
  - Jede Variable x_i ist eine boolesche Formel.
  - Mit G und H sind auch die Konjunktion $(G \wedge H)$ und die Disjunktion $(G \vee H)$ von G und H sowie die Negation $\neg G$ von G Formeln.
- Eine Belegung von $x_1, \ldots, x_n$ ist ein Wort $a = a_1 \ldots a_n \in \{0,1\}^n$.
- Der Wert $F(a)$ von F unter a ist induktiv wie folgt definiert:

F	0	1	x_i	$\neg G$	$(G \wedge H)$	$(G \vee H)$
$F(a)$	0	1	a_i	$1 - G(a)$	$G(a)H(a)$	$G(a) + H(a) - G(a)H(a)$

- Durch die Formel F wird also eine n-stellige boolesche Funktion $F : \{0,1\}^n \to \{0,1\}$ definiert, die wir ebenfalls mit F bezeichnen.

Notation

Wir benutzen die Implikation $G \to H$ als Abkürzung für die Formel $\neg G \vee H$ und die Äquivalenz $G \leftrightarrow H$ als Abkürzung für $(G \to H) \wedge (H \to G)$.

Beispiel (Wahrheitswerttabelle)

Die Formel $F = (G \to H)$ mit den Teilformeln

- $G = (x_2 \wedge x_3)$ und
- $H = (\neg x_1 \vee \neg x_2)$

berechnet nebenstehende boolesche Funktion $F : \{0,1\}^3 \to \{0,1\}$.

a	$G(a)$	$H(a)$	$F(a)$
000	0	1	1
001	0	1	1
010	0	1	1
011	1	1	1
100	0	1	1
101	0	1	1
110	0	0	1
111	1	0	0

◁

Definition

- Zwei Formeln F und G heißen (logisch) äquivalent (kurz $F \equiv G$), wenn sie dieselbe boolesche Funktion berechnen.
- Eine Formel F heißt erfüllbar, falls es eine Belegung a mit $F(a) = 1$ gibt.
- Gilt sogar für alle Belegungen a, dass $F(a) = 1$ ist, so heißt F Tautologie.

Beispiel

- Die Formel $F = (G \to H)$ mit $G = (x_2 \wedge x_3)$ und $H = (\neg x_1 \vee \neg x_2)$ ist erfüllbar, da $F(000) = 1$ ist.
- F ist aber keine Tautologie, da $F(111) = 0$ ist. ◁

Präzedenzregeln zur Klammerersparnis

- Der Junktor $\wedge$ bindet stärker als der Junktor $\vee$ und dieser wiederum stärker als die Junktoren $\rightarrow$ und $\leftrightarrow$.
- Formeln der Form $(F_1 \circ (F_2 \circ (F_3 \circ \cdots \circ F_n)\cdots)))$, $\circ \in \{\wedge, \vee\}$, kürzen wir durch $(F_1 \circ \cdots \circ F_n)$ ab und schreiben dafür auch $\bigwedge\limits_{1\leq i\leq n} F_i$ bzw. $\bigvee\limits_{1\leq i\leq n} F_i$.

Beispiel (Formel für die mehrstellige Entweder-Oder Funktion)

- Folgende Formel nimmt unter einer Belegung $a = a_1 \ldots a_n$ genau dann den Wert 1 an, wenn $\sum_{i=1}^{n} a_i = 1$ ist:

$$G(x_1, \ldots, x_n) = (x_1 \vee \cdots \vee x_n) \wedge \bigwedge_{1\leq i<j\leq n} \neg(x_i \wedge x_j)$$

- D.h. es gilt genau dann $G(a) = 1$, wenn genau eine Variable x_i mit dem Wert $a_i = 1$ belegt ist.
- Diese Formel wird im Beweis des nächsten Satzes benötigt. $\triangleleft$

Das aussagenlogische Erfüllbarkeitsproblem

Erfüllbarkeitsproblem für boolesche Formeln (*satisfiability*, SAT):

Gegeben: Eine boolesche Formel F.
Gefragt: Ist F erfüllbar?

Dabei kodieren wir boolesche Formeln F durch Binärstrings w_F und ordnen umgekehrt jedem Binärstring w eine Formel F_w zu.

Um die Notation zu vereinfachen, werden wir zukünftig jedoch F anstelle von w_F schreiben.

Satz (Cook, Karp, Levin)

SAT ist NP-vollständig.

SAT ist NP-vollständig

SAT ∈ NP

Eine NTM kann bei Eingabe einer booleschen Formel F zunächst eine Belegung a nichtdeterministisch raten und dann in Polynomialzeit testen, ob $F(a) = 1$ ist (*guess and verify* Strategie).

SAT ist NP-hart

- Sei L eine beliebige NP-Sprache und sei $M = (Z, \Sigma, \Gamma, \delta, q_0)$ eine durch ein Polynom p zeitbeschränkte k-NTM mit $L(M) = L$.
- Da sich jede $t(n)$-zeitbeschränkte k-NTM in Zeit $O(t^2(n))$ durch eine 1-NTM simulieren lässt, können wir $k = 1$ annehmen.
- Zudem können wir $Z = \{q_0, \ldots, q_m\}$, $E = \{q_m\}$ und $\Gamma = \{a_1, \ldots, a_l\}$ sowie $\delta(q_m, a) = \{(q_m, a, N)\}$ für alle $a \in \Gamma$ annehmen.
- Unsere Aufgabe besteht nun darin, zu jedem Wort $w = w_1 \ldots w_n \in \Sigma^*$ eine Formel F_w mit folgenden Eigenschaften zu konstruieren:
  - $w \in L(M) \iff F_w \in \mathrm{SAT}$,
  - die Reduktionsfunktion $w \mapsto F_w$ ist in FP berechenbar.

SAT ist NP-hart

Idee:

Konstruiere F_w so, dass F_w unter einer Belegung a genau dann wahr wird, wenn a eine akzeptierende Rechnung von $M(w)$ beschreibt.

- Wir bilden F_w über den Variablen

$$\begin{aligned} x_{t,i}, &\quad \text{für } 0 \leq t \leq p(n), 0 \leq i \leq m \\ y_{t,j}, &\quad \text{für } 0 \leq t \leq p(n), -p(n) \leq j \leq p(n) \\ z_{t,j,a}, &\quad \text{für } 0 \leq t \leq p(n), -p(n) \leq j \leq p(n), a \in \Gamma \end{aligned}$$

- Diese Variablen stehen für folgende Aussagen:

$x_{t,i}$: zum Zeitpunkt t befindet sich M im Zustand q_i

$y_{t,j}$: zur Zeit t besucht M das Feld mit der Nummer j

$z_{t,j,a}$: zur Zeit t steht das Zeichen a auf dem Feld mit der Nr. j

- Konkret sei $F_w = R \wedge S_w \wedge \ddot{U}_1 \wedge \ddot{U}_2 \wedge E$.

SAT ist NP-hart

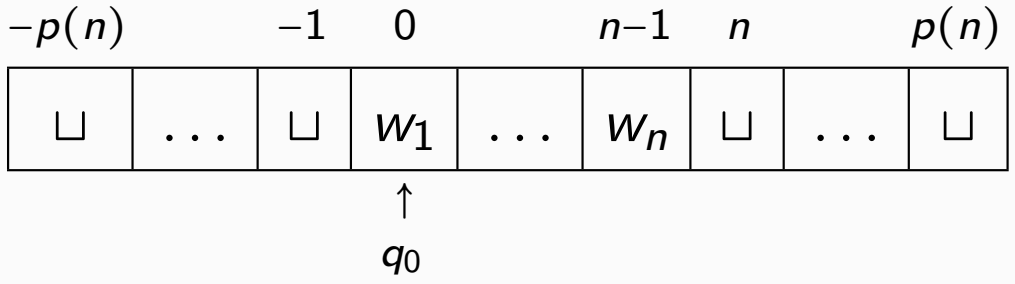
- Konkret sei $F_w = R \wedge S_w \wedge \ddot{U}_1 \wedge \ddot{U}_2 \wedge E$.
- Dabei stellt die Formel $R = \bigwedge_{t=0}^{p(n)} R_t$ (Randbedingungen) sicher, dass wir jeder erfüllenden Belegung von F_w eindeutig eine Folge von Konfigurationen $K_0, \ldots, K_{p(n)}$ zuordnen können:

$$R_t = \; G(x_{t,0}, \ldots, x_{t,m}) \wedge G(y_{t,-p(n)}, \ldots, y_{t,p(n)}) \wedge \bigwedge_{j=-p(n)}^{p(n)} G(z_{t,j,a_1}, \ldots, z_{t,j,a_l}).$$

- Die Teilformel R_t sorgt also dafür, dass zum Zeitpunkt t
  - genau ein Zustand $q_i \in Z$ eingenommen wird,
  - genau ein Bandfeld $j \in \{-p(n), \ldots, p(n)\}$ besucht wird und
  - auf jedem Feld j genau ein Zeichen $a_k \in \Gamma = \{a_1, \ldots, a_l\}$ steht.

SAT ist NP-hart

- Die Formel S_w (wie Startbedingung) stellt sicher, dass zum Zeitpunkt 0 tatsächlich die Startkonfiguration vorliegt:



$$S_w = x_{0,0} \wedge y_{0,0} \wedge \bigwedge_{j=-p(n)}^{-1} z_{0,j,\sqcup} \wedge \bigwedge_{j=0}^{n-1} z_{0,j,w_{j+1}} \wedge \bigwedge_{j=n}^{p(n)} z_{0,j,\sqcup}$$

- Die Formel $\ddot{U}_1$ sorgt dafür, dass der Inhalt von nicht besuchten Feldern beim Übergang von K_t zu K_{t+1} unverändert bleibt:

$$\ddot{U}_1 = \bigwedge_{t=0}^{p(n)-1} \bigwedge_{j=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} \left(\neg y_{t,j} \wedge z_{t,j,a} \rightarrow z_{t+1,j,a} \right)$$

SAT ist NP-hart

- $\ddot{U}_2$ achtet darauf, dass sich bei jedem Rechenschritt der Zustand, die Kopfposition und das gerade gelesene Zeichen gemäß einer Anweisung in δ verändern:

$$\ddot{U}_2 = \bigwedge_{t=0}^{p(n)-1} \bigwedge_{j=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} \bigwedge_{i=0}^{m} \Big(x_{t,i} \wedge y_{t,j} \wedge z_{t,j,a} \rightarrow \bigvee_{(q_{i'},a',D) \in \delta(q_i,a)} x_{t+1,i'} \wedge y_{t+1,j+D} \wedge z_{t+1,j,a'} \Big),$$

wobei

$$j + D = \begin{cases} j-1, & D = L \\ j, & D = N \\ j+1, & D = R \end{cases}$$

- Schließlich überprüft E, ob $M(w)$ nach (spätestens) $p(n)$ Schritten den Endzustand q_m erreicht hat:

$$E = x_{p(n),m}$$

SAT ist NP-hart

- Da der Aufbau der Formel $f(w) = F_w$ einem einfachen Bildungsgesetz folgt und ihre Länge polynomiell in n ist, folgt $f \in \mathrm{FP}$.
- Es ist klar, dass F_w im Fall $w \in L(M)$ erfüllbar ist, indem wir die Variablen von F_w gemäß einer akz. Rechnung von $M(w)$ belegen.
- Umgekehrt führt eine Belegung a mit $F_w(a) = 1$ wegen $R(a) = 1$ eindeutig auf eine Konfigurationenfolge $K_0, \ldots, K_{p(n)}$.
- Für diese gilt:
  - K_0 ist Startkonfiguration von $M(w)$ (wegen $S_w(a) = 1$)
  - $K_i \vdash K_{i+1}$ für $i = 0, \ldots, p(n) - 1$ (wegen $\ddot{U}_1(a) = \ddot{U}_2(a) = 1$)
  - der Zustand von $K_{p(n)}$ ist m (wegen $E(a) = 1$)
- Also gilt für alle $w \in \Sigma^*$ die Äquivalenz

  $$w \in L(M) \Leftrightarrow F_w \in \mathrm{SAT},$$

  d.h. die FP-Funktion $f : w \mapsto F_w$ reduziert $L(M)$ auf SAT. □

Boolesche Schaltkreise

Als nächstes betrachten wir das Erfüllbarkeitsproblem für Schaltkreise.

Definition

- Ein boolescher Schaltkreis über den Variablen $x_1, \ldots, x_n$ ist eine Folge $S = (g_1, \ldots, g_m)$ von Gattern

  $$g_l \in \{0, 1, x_1, \ldots, x_n, (\neg, j), (\wedge, j, k), (\vee, j, k)\} \text{ mit } 1 \leq j, k < l.$$

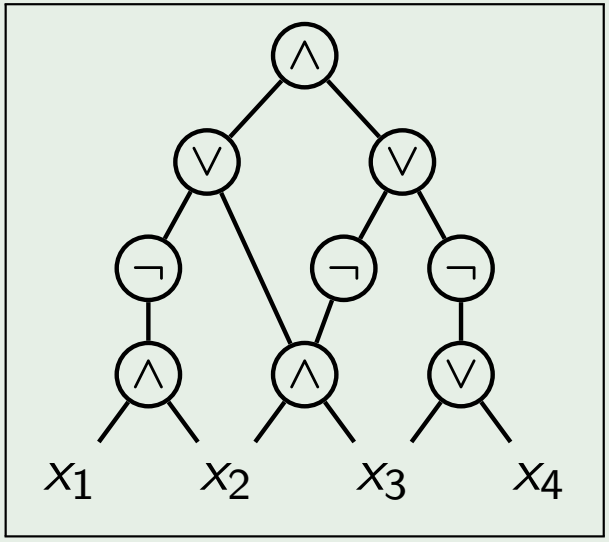
- Jedes Gatter g_l berechnet eine n-stellige boolesche Funktion $g_l : \{0,1\}^n \to \{0,1\}$.
- Für $a = a_1 \ldots a_n \in \{0,1\}^n$ ist $g_l(a)$ induktiv wie folgt definiert:

g_l	0	1	x_i	$(\neg, j)$	$(\wedge, j, k)$	$(\vee, j, k)$
$g_l(a)$	0	1	a_i	$1 - g_j(a)$	$g_j(a)g_k(a)$	$g_j(a) + g_k(a) - g_j(a)g_k(a)$

- S berechnet die boolesche Funktion $S(a) = g_m(a)$.
- S heißt erfüllbar, wenn eine Eingabe $a \in \{0,1\}^n$ ex. mit $S(a) = 1$.

Boolesche Schaltkreise

Beispiel



Graphische Darstellung des Schaltkreises

$S = (x_1, x_2, x_3, x_4, (\wedge, 1, 2), (\wedge, 2, 3), (\vee, 3, 4), (\neg, 5), (\neg, 6), (\neg, 7), (\vee, 6, 8), (\vee, 9, 10), (\wedge, 11, 12))$.

Bemerkung

- Die Anzahl der Eingänge eines Gatters g wird als Fanin von g bezeichnet,
- die Anzahl der Ausgänge von g (also die Anzahl der Gatter, die g als Eingabe benutzen) als Fanout.
- Boolesche Formeln entsprechen also genau den booleschen Schaltkreisen $S = (g_1, \ldots, g_m)$, bei denen jedes Gatter g_i, $1 \leq i \leq m-1$, Fanout 1 hat.
- Eine boolesche Formel F kann somit leicht in einen äquivalenten Schaltkreis S mit $S(a) = F(a)$ für alle Belegungen a transformiert werden.

Das Erfüllbarkeitsproblem für boolesche Schaltkreise

Erfüllbarkeitsproblem für boolesche Schaltkreise (CirSat):

Gegeben: Ein boolescher Schaltkreis S.
Gefragt: Ist S erfüllbar?

Satz

CirSat ist NP-vollständig.

Beweis

Klar, da Sat $\leq^p$ CirSat und CirSat $\in$ NP gilt. □

Bemerkung

- Da Sat NP-vollständig ist, ist auch CirSat auf Sat reduzierbar.
- Dies bedeutet, dass sich jeder Schaltkreis S in Polynomialzeit in eine erfüllbarkeitsäquivalente Formel F_S überführen lässt.
  F_S und S müssen aber nicht logisch äquivalent sein.
- CirSat ist sogar auf eine spezielle Sat-Variante reduzierbar.

Formeln in konjunktiver Normalform (KNF)

- Ein Literal ist eine Variable x_i oder eine negierte Variable $\neg x_i$, die wir auch kurz mit $\bar{x}_i$ bezeichnen.
- Eine Klausel ist eine Disjunktion $C = \bigvee_{j=1}^{k} l_j$ von Literalen $l_1, \ldots, l_k$.
- Hierbei ist auch $k = 0$ zulässig, d.h. die leere Klausel repräsentiert die Konstante 0 und wird üblicherweise mit $\square$ bezeichnet.
- Eine boolesche Formel F ist in konjunktiver Normalform (kurz KNF), falls $F = \bigwedge_{j=1}^{m} C_j$ eine Konjunktion von Klauseln $C_1, \ldots, C_m$ ist.
- Auch hier ist $m = 0$ zulässig, wobei die leere Konjunktion die Konstante 1 repräsentiert.
- Enthält jede Klausel höchstens k Literale, so heißt F in k-KNF.
- Klauseln werden oft als Menge $C = \{l_1, \ldots, l_k\}$ ihrer Literale und KNF-Formeln als Menge $F = \{C_1, \ldots, C_m\}$ ihrer Klauseln dargestellt.
- Enthält F die leere Klausel, so ist F unerfüllbar.
- Dagegen ist die leere Formel eine Tautologie.

Erfüllbarkeitsprobleme für KNF-Formeln

Erfüllbarkeitsproblem für k-KNF Formeln (k-SAT):

Gegeben: Eine boolesche Formel F in k-KNF.

Gefragt: Ist F erfüllbar?

Beispiel

- Der 3-KNF Formel $F = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4)$ entspricht die Klauselmenge $F = \{\{x_1, \bar{x}_2\}, \{\bar{x}_1, x_3\}, \{x_2, \bar{x}_3, x_4\}\}$.
- Offenbar ist $F(0000) = 1$, d.h. $F \in$ 3-SAT.

Satz

k-SAT ist für $k \leq 2$ in P entscheidbar und für $k \geq 3$ NP-vollständig.