Komplexitätstheorie II

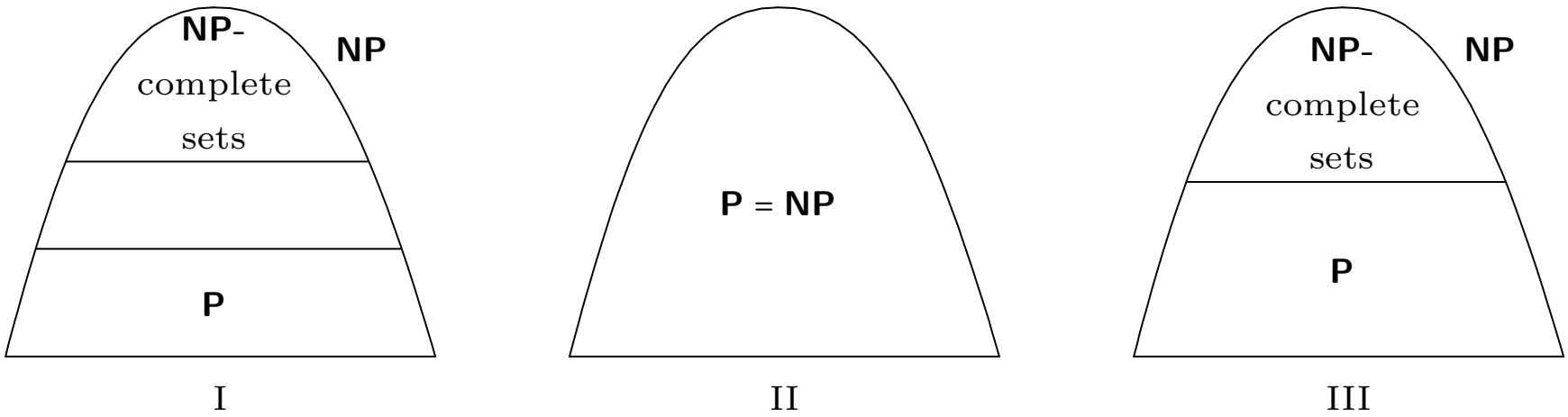
Markus Lohrey

Universität Siegen

Sommersemester 2023

Nicht **NP**-vollständige Mengen innerhalb von **NP** ∖ **P**

Nach unserem bisherigen Wissen wären folgende 3 Situationen möglich:



Im folgenden werden wir zeigen, dass die 3. Möglichkeit prinzipiell nicht möglich ist.

Satz 1 (Ladner)

Ist **P** ≠ **NP**, dann existiert (effektiv) eine Sprache $L \in \mathbf{NP} \setminus \mathbf{P}$, die nicht **NP**-vollständig (unter Polynomialzeitreduktionen) ist.

Nicht **NP**-vollständige Mengen innerhalb von **NP** $\setminus$ **P**

Beweis: Für eine schwach monoton wachsende Funktion $f : \mathbb{N} \to \mathbb{N}$ $(x_2 > x_1 \Rightarrow f(x_2) \geq f(x_1))$ sei

$$L_f = \{x \in \Sigma^* \mid x \in \mathrm{SAT} \wedge f(|x|) \text{ ist gerade}\}.$$

Im folgenden geben wir ein det. Programm an, welches eine Funktion $f : \mathbb{N} \to \mathbb{N}$ rekursiv in Zeit $\mathcal{O}(n)$ berechnet. $\rightsquigarrow$ $L_f \in$ **NP**.

Sei $M_1, M_2, \ldots$ eine Aufzählung aller deterministischen Turingmaschinen mit einer zusätzlichen polynomiellen Zeitschranke.

Eigentlich zählen wir alle Paare (M_i, p_j) von deterministischen Turingmaschinen und Polynomen auf.

Wir erhalten daraus obige Aufzählung, indem M_i mit einem zusätzlichen Schrittzähler, welcher die polynomiale Zeitschranke p_j sicherstellt, versehen wird.

Beachte: Jedes Polynom ist zeitkonstruierbar.

Nicht **NP**-vollständige Mengen innerhalb von **NP** $\setminus$ **P**

Sei analog $R_1, R_2, \ldots$ eine Aufzählung aller Polynomialzeitreduktionen.

Wir verwenden ferner im folgenden einen beliebigen (exponentiellen) deterministischen Algorithmus zur Erkennung von SAT.

Für Mengen L und K ist

$$L \Delta K = L \setminus K \cup K \setminus L$$

die symmetrische Differez.

Beachte: Ist $L \in \mathcal{C}$ für eine der in der Vorlesung betrachteten Komplexitätsklassen $\mathcal{C}$ und ist $|L \Delta K| < \infty$, so gilt auch $K \in \mathcal{C}$ (bis auf endliche viele Ausnahmen kann eine Maschine für L auch für K verwendet werden).

Nicht **NP**-vollständige Mengen innerhalb von **NP** $\setminus$ **P**

FUNCTION $f(n)$
if $n = 0$ **then return** 0
else **(*** $n > 0$ ***)**
  $i := 0$; $k := 0$
  loop für genau n Befehlsschritte
    $k := f(i)$; $i := i + 1$
  endloop
  (* Beachte: alle rekursiven Aufrufe $f(i)$ geschehen nur mit $i < n$. ***)**
  (* Der Wert von k ist nach Durchlaufen dieser Schleife sehr klein. ***)**
  if $k = 2j$ **(*** k gerade ***) then**
    suche für max. n Schritte in längenlexikograph. Reihenfolge ein $x \in L(M_j) \Delta L_f$
  endif
  if $k = 2j + 1$ **(*** k ungerade ***) then**
    suche für max. n Schritte in längenlexikograph. Reihenfolge ein $x \in \Sigma^*$ mit
    $(x \in \mathrm{SAT} \wedge R_j(x) \notin L_f) \vee (x \notin \mathrm{SAT} \wedge R_j(x) \in L_f)$
  endif
  if ein solcher Zeuge $x \in \Sigma^*$ wurde gefunden **then return** $k + 1$
  else return k
  endif
endif

Nicht **NP**-vollständige Mengen innerhalb von **NP** $\setminus$ **P**

Behauptung 1: $f(n)$ ist wohldefiniert und wird in $\mathcal{O}(n)$ Schritten berechnet. Es gilt weiter $f(n) \leq f(n+1) \leq f(n)+1$ für alle $n \in \mathbb{N}$.

Wir zeigen nur $f(n) \leq f(n+1) \leq f(n)+1$ für alle $n \in \mathbb{N}$, der Rest ist klar.

Zunächst zeigen wir mit Induktion über n, dass $f(n) \leq f(n+1)$ gilt.

Induktionsanfang: Da $f(0) = 0$ gilt, folgt $f(0) \leq f(1)$.

Induktionsschritt: Sei k_0 (bzw. k_1) der im Algorithmus bei Eingabe n (bzw. $n+1$) in der **loop**-Schleife berechnete k-Wert.

Also gibt es $i_0 < n$, $i_1 < n+1$ mit $i_0 \leq i_1$, $f(i_0) = k_0$ und $f(i_1) = k_1$.

Aus der Induktionshypothese folgt $k_0 \leq k_1$.

Falls $k_0 = k_1$, so folgt $f(n) \leq f(n+1)$ direkt aus dem Algorithmus für f.

Falls $k_0 + 1 \leq k_1$, so gilt $f(n) \leq k_0 + 1 \leq k_1 \leq f(n+1)$.

Da $f(n+1) \leq f(i)+1$ für ein $i \leq n$ gilt, erhalten wir somit auch $f(n+1) \leq f(i)+1 \leq f(n)+1$.

Nicht **NP**-vollständige Mengen innerhalb von **NP** $\setminus$ **P**

Wir zeigen jetzt mit Widerspruch: wenn $\mathbf{P} \neq \mathbf{NP}$ gilt, so liegt L_f weder in $\mathbf{P}$ noch ist L_f $\mathbf{NP}$-vollständig (beachte: $L_f \in \mathbf{NP}$).

Gelte also $\mathbf{P} \neq \mathbf{NP}$ und ($L_f \in \mathbf{P}$ oder L_f ist $\mathbf{NP}$-vollständig).

Behauptung 2: f ist beschränkt, d.h. $\exists n_0, m \; \forall n \geq n_0 : f(n) = m$.

1. Fall: $L_f \in \mathbf{P}$

Dann existiert ein j mit $L_f = L(M_j)$.

Angenommen es existiert ein $n \in \mathbb{N}$ mit $f(n) > 2j$.

Wegen Behauptung 1 existiert ein $n \in \mathbb{N}$ mit $f(n) = 2j + 1$.

Sei n minimal, so dass $f(n) = 2j + 1$.

Falls bei der Berechnung von $f(n)$ kein Zeuge x gefunden wird, gibt es ein $i < n$ mit $f(i) = f(n) = 2j + 1$.

Widerspruch zur Minimalität von n

Nicht **NP**-vollständige Mengen innerhalb von **NP** $\setminus$ **P**

Falls bei der Berechnung von $f(n)$ ein Zeuge x gefunden wird, muss $k = 2j$ bei der Berechnung von $f(n)$ gelten.

Also gehört der gefundene Zeuge x zu $L(M_j) \Delta L_f$.

Es gilt aber $L(M_j) \Delta L_f = \varnothing$, Widerspruch!

Also gilt $f(n) \leq 2j$ für alle n.

2. Fall: L_f ist **NP**-vollständig, d.h. $\mathrm{SAT} \leq_m^p L_f$.

Also gibt es ein $j \geq 0$ mit $\forall x \in \Sigma^* : x \in \mathrm{SAT} \Leftrightarrow R_j(x) \in L_f$.

$\rightsquigarrow \neg\exists x \in \Sigma^* : (x \in \mathrm{SAT} \wedge R_j(x) \notin L_f) \vee (x \notin \mathrm{SAT} \wedge R_j(x) \in L_f)$.

Wie in Fall 1 zeigt man, dass $f(n) \leq 2j + 1$ für alle n.

Nicht **NP**-vollständige Mengen innerhalb von **NP** $\setminus$ **P**

Sei also $f(n) = m$ für alle $n \geq n_0$.

Aus der Definition von L_f (Folie 3) folgt:

1. Ist m gerade, so gilt $|L_f \,\Delta\, \mathrm{SAT}| < \infty$ und L_f ist **NP**-vollständig.
2. Ist m ungerade, so ist L_f endlich und damit $L_f \in \mathbf{P}$.

Aus der Vorraussetzung $\mathbf{P} \neq \mathbf{NP}$ folgt nun:

- Wenn $m = 2j$, dann ist $L(M_j) = L_f$, und diese Sprache ist nach (1) **NP**-vollständig.

  Widerspruch zu $L(M_j) \in \mathbf{P}$.

- Wenn $m = 2j + 1$, dann ist R_j eine Reduktion von SAT auf L_f.

  Widerspruch zu (2), da L_f endlich ist und wegen $\mathbf{P} \neq \mathbf{NP}$ die Sprache SAT nicht auf eine Sprache in **P** reduziert werden kann. $\square$

Dünne Mengen und der Satz von Mahaney

Eine Menge $L \subseteq \Sigma^*$ ist dünn, wenn ein Polynom $p(n)$ mit

$$\forall n \geq 0 : \left| \left\{ w \in L \mid |w| = n \right\} \right| \leq p(n).$$

existiert.

Z. B. ist jede Sprache über einem unären Alphabet $\{a\}$ dünn.

Satz von Mahaney, 1982

Gilt **P** $\neq$ **NP**, so gibt es keine dünne Sprache, die **NP**-schwierig (unter Polynomialzeitreduktionen) ist.

Beweis des Satzes von Mahaney

Beweis: Es wird zunächst eine Menge SAT' definiert:

$$\mathrm{SAT}' = \left\{ \langle F, x \rangle \,\middle|\, \begin{array}{l} F \text{ ist boolesche Formel in den Variablen } x_1, \ldots, x_n, \\ x \in \{0,1\}^n \text{ und } \exists y \in \{0,1\}^n : (y \geq x \wedge F(y) = 1) \end{array} \right\}$$

Hierbei ist

- $\geq$ die lexikographische Ordnung auf $\{0,1\}^*$ und
- $F(y) = 1$ bedeutet, dass der Bitstring y als Belegung der Variablen $x_1, \ldots, x_n$ interpretiert wird und F sich unter dieser Belegung zu wahr auswertet.

Behauptung: SAT' ist **NP**-vollständig.

1. $\mathrm{SAT}' \in$ **NP**: Rate Belegung $y \geq x$ und überprüfe, ob $F(y) = 1$.
2. SAT' ist **NP**-schwierig:
   O.B.d.A. enthalte F die Variablen $x_1, \ldots, x_n$. Wir reduzieren SAT auf SAT' mit der Abbildung $F \mapsto \langle F, 0^n \rangle$. Es gilt offensichtlich:

$$F \in \mathrm{SAT} \Leftrightarrow \langle F, 0^n \rangle \in \mathrm{SAT}'$$

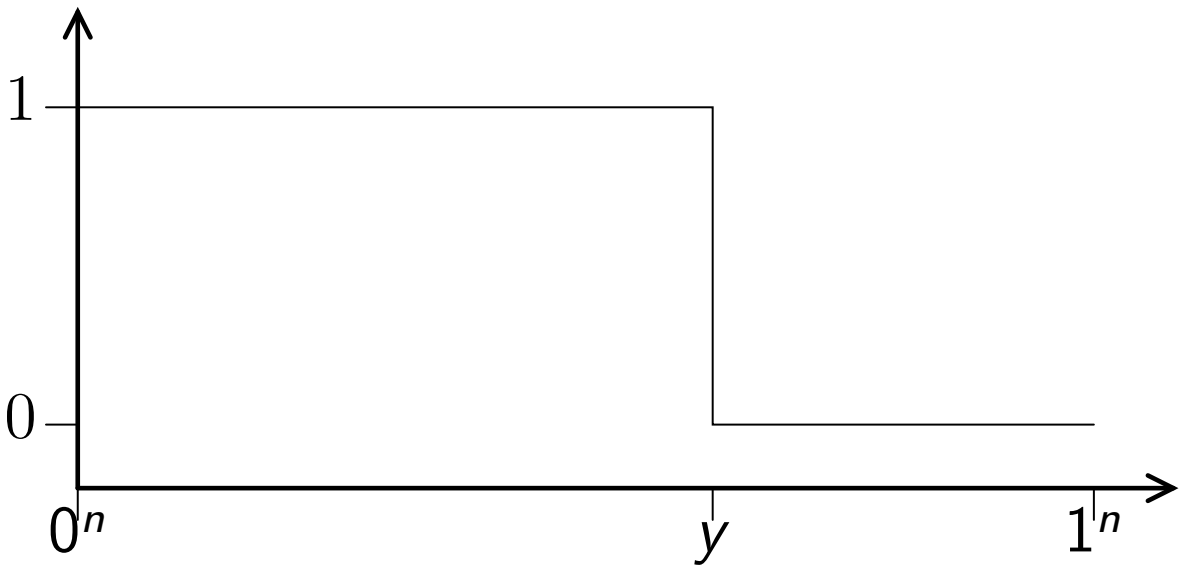
Beweis des Satzes von Mahaney

Für eine Formel F in den Variablen $x_1, \ldots, x_n$ definieren wir eine Funktion $f_F : [0, \ldots, 2^n - 1] \to \{0,1\}$ durch

$$f_F(x) \quad = \quad \begin{cases} 1, \text{ falls } \langle F, x \rangle \in \mathrm{SAT}' \\ 0, \text{ sonst} \end{cases}$$

Der Definitionsbereich von f_F wird hier mit $\{0,1\}^n$ identifiziert.

Der Graph von f_F ist eine Stufenfunktion:



Der Wert y ist hier die (lexikographisch) größte erfüllende Belegung für F.

Beweis des Satzes von Mahaney

Annahme: Sei $L \subseteq \Gamma^*$ eine dünne **NP**-schwierige Sprache.

Da $\mathrm{SAT}' \in \mathbf{NP}$, gibt es eine Polynomialzeitreduktion $H : \Sigma^* \to \Gamma^*$ von SAT' auf L:

$$\langle F, x \rangle \in \mathrm{SAT}' \Leftrightarrow H(\langle F, x \rangle) \in L$$

Die Länge einer Eingabe $\langle F, x \rangle$ definieren wir im folgenden zu $n = |F|$ und wir können o.B.d.A. annehmen, dass die Formel F nur Variablen aus der Menge $\{x_1, \ldots, x_n\}$ enthält.

Da L nach Annahme eine dünne Sprache ist, gibt es ein Polynom $p(n)$ mit

$$\left| \left\{ H(\langle F, x \rangle) \mid \langle F, x \rangle \in \mathrm{SAT}' \wedge |\langle F, x \rangle| \leq n \right\} \right| \leq p(n). \tag{1}$$

Wir zeigen, dass unter obigen Annahmen $\mathrm{SAT} \in \mathbf{P}$, d. h. $\mathbf{P} = \mathbf{NP}$ gilt.

Sei F eine boolesche Formel mit $n = |F|$, in der nur Variablen aus der Menge $\{x_1, \ldots, x_n\}$ vorkommen.

Beweis des Satzes von Mahaney

Sei $I = [0, \ldots, 2^n - 1]$.

Wir werden den Verlauf der Stufenfunktion $f_F : I \to \{0, 1\}$ berechnen.

Aus (1) folgt

$$\left|\left\{H(\langle F, x\rangle) \mid \langle F, x\rangle \in \mathrm{SAT}' \ \wedge \ x \in I\right\}\right| \leq p(n). \tag{2}$$

Wir unterteilen nun I in $2 \cdot p(n)$ (in etwa) gleichgroße Teilintervalle.

Jedes Intervall wird durch die linke Grenze und die Länge repräsentiert.

Die linken Grenzen seien $y_1 < y_2 < \cdots < y_{2p(n)}$.

Für die linken Grenzen y_i wird $H(\langle F, y_i\rangle)$ berechnet und in einer Tabelle abgelegt.

Beweis des Satzes von Mahaney

Angenommen es gilt $H(\langle F, y_i \rangle) = H(\langle F, y_j \rangle)$ für $i < j$.

Dann gilt $\langle F, y_i \rangle \in \mathrm{SAT}' \Leftrightarrow \langle F, y_j \rangle \in \mathrm{SAT}'$ und damit $f_F(y_i) = f_F(y_j)$, d.h.

$$f_F(y_i) = f_F(y_{i+1}) = \cdots = f_F(y_{j-1}) = f_F(y_j).$$

Sollte F erfüllbar sein, kann die größte erfüllende Belegung y von F in keinem der Intervalle mit den linken Grenzen $y_i, y_{i+1}, \ldots, y_{j-1}$ liegen.

Ansonsten: $y_i \leq y < y_j \;\Rightarrow\; f_F(y_i) \neq f_F(y_j) \;\Rightarrow\; H(\langle F, y_i \rangle) \neq H(\langle F, y_j \rangle)$

Kollisionsauflösung

Lösche alle Teilintervalle mit linken Grenzen $y_i, y_{i+1}, \ldots, y_{j-1}$.

Dabei wird das Intervall, das die größte erfüllende Belegung y enthält (sofern überhaupt eine erfüllende Belegung existiert), nicht gelöscht.

Beweis des Satzes von Mahaney

Seien $z_1 < z_2 < \ldots < z_l$ die verbleibenden linken Intervallgrenzen.

Am Ende dieser Prozedur sind die Werte $H(\langle F, z_i \rangle)$ für alle linken Grenzen der verbleibenden Intervalle verschieden.

Angenommen F ist erfüllbar und für die größte erfüllende Belegung y von F gilt $z_{p(n)+1} \leq y$.

Dann gilt $z_i \leq y$ und somit $\langle F, z_i \rangle \in \mathrm{SAT}'$ für alle $1 \leq i \leq p(n) + 1$.

Also gilt $H(\langle F, z_i \rangle) \in L$ für alle $1 \leq i \leq p(n) + 1$.

Dies widerspricht (2)!

Also gehört y zu keinem der Intervalle mit den linken Grenzen $z_{p(n)+1}, \ldots, z_l$.

Wir löschen daher auch diese Intervalle.

Danach bleiben nur noch höchsten $p(n)$ Intervalle übrig.

Beweis des Satzes von Mahaney

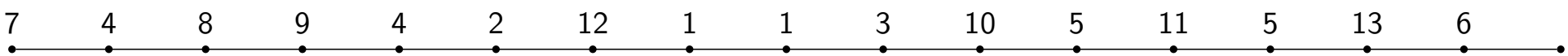
Jetzt wird eine Intervallhalbierung bei den verbleibenden Intervallen durchgeführt.

Dies erzeugt maximal $2 \cdot p(n)$ Intervalle, die nach dem gleichen Verfahren wiederum auf maximal $p(n)$ Intervalle verringert werden. Dann erfolgt eine erneute Intervallhalbierung usw.

Nach höchstens n Halbierungen gibt es nur noch 1-Punkt-Intervalle (das Intervall I enthält 2^n Elemente) und das Verfahren der Intervallhalbierungen wird abgebrochen.

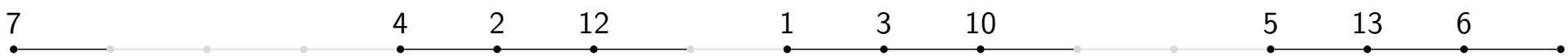
Beweis des Satzes von Mahaney

Example: $p(n) = 8$



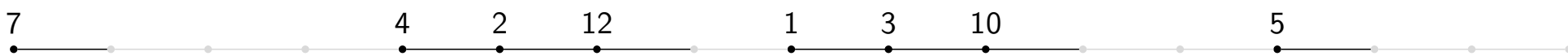
Beweis des Satzes von Mahaney

Example: $p(n) = 8$



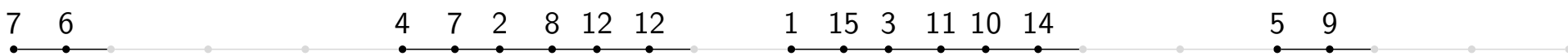
Beweis des Satzes von Mahaney

Example: $p(n) = 8$



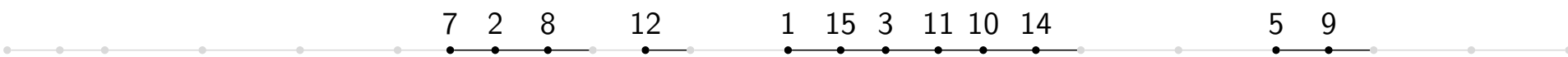
Beweis des Satzes von Mahaney

Example: $p(n) = 8$



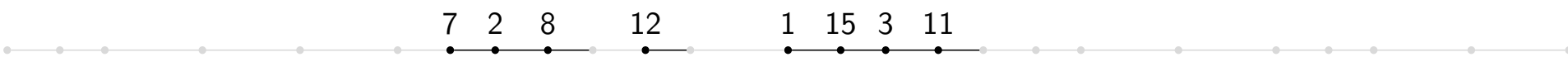
Beweis des Satzes von Mahaney

Example: $p(n) = 8$



Beweis des Satzes von Mahaney

Example: $p(n) = 8$



Beweis des Satzes von Mahaney

Die bis zu diesem Zeitpunkt verbrauchte Zeit ist polynomial beschränkt.

Seien $z_1, \ldots, z_l$ $(l \leq p(n))$ die linken Grenzen der verbleibenden 1-Punkt-Intervalle.

Jetzt kann in polynomieller Zeit für jedes i, $1 \leq i \leq l$, der Wert $F(z_i)$ berechnet werden.

Ist $F(z_i) = 0$ für alle i, so ist F unerfüllbar. Sonst ist F erfüllbar (das größte z_i mit $F(z_i) = 1$ ergibt die größte erfüllende Belegung).

Damit können wir in Polynomialzeit entscheiden, ob $F \in \mathrm{SAT}$. □

Orakel-Turingmaschinen & relative Schwierigkeit von $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$

Eine nichtdeterministische Orakel-Turingmaschine (kurz OTM) $M^?$ ist eine nichtdeterministische Turingmaschine mit folgenden Besonderheiten:

- $M^?$ hat drei ausgezeichneten Zuständen $o_J, o_N, o_?$ sowie
- ein ausgezeichnetes Arbeitsband — das Orakelband — auf welches nur geschrieben wird.
- Das Orakelband enthält zu jedem Zeitpunkt einen String $w(o)$ über einem Alphabet Γ.
- Befindet sich $M^?$ im Zustand $o_?$, so kann $M^?$ unabhängig von den gelesenen Bandsymbolen nichtdeterministisch in den Zustand o_N oder den Zustand o_J gehen. Der Inhalt des Orakelbands wird dabei gelöscht.

Eine OTM $M^?$ ist deterministisch, falls sich $M^?$ auf allen Zuständen außer $o_?$ deterministisch verhält.

Orakel-Turingmaschinen

Zeit- und Platzschranken werden für OTMs wie für normale Turingmaschinen definiert.

Wir betrachten dafür eine OTM als eine nichtdeterministische Turingmaschine (d. h. auf allen Berechnungspfaden muss die Zeitschranke bzw. Platzschranke eingehalten werden).

Sei $A \subseteq \Gamma^*$ eine beliebige (nicht notwendigerweise entscheidbare) Menge.

Wir definieren Berechnungen einer Maschine M^A auf eine Eingabe $v \in \Sigma^*$ wie folgt:

- Auf Zuständen in $Q \setminus \{o_?\}$ verhält sich M^A wie $M^?$.
- Befindet sich $M^?$ im Zustand $o_?$, so ist der Folgezustand (unabhängig von gelesenen Bandsymbolen) o_J (bzw. o_N), falls aktuell $w(o) \in A$ (bzw. $w(o) \notin A$) gilt. Der Inhalt des Orakelbands wird wieder gelöscht.

Relative Komplexitätsklassen

Wir definieren die Komplexitätslassen

$$\mathbf{P}^A = \{L(M^A) \mid M^? \text{ ist eine det. polynomial zeitbeschränkte OTM}\}$$
$$\mathbf{NP}^A = \{L(M^A) \mid M^? \text{ ist eine nichtdet. polynomial zeitbeschränkte OTM}\}$$

Für eine Komplexitätsklasse $\mathcal{C}$ sein $\mathbf{P}^{\mathcal{C}} = \bigcup_{A \in \mathcal{C}} \mathbf{P}^A$ und analog für $\mathbf{NP}^{\mathcal{C}}$.

Bemerkung:

1. $\mathbf{NP} \cup \mathbf{CoNP} \subseteq \mathbf{P}^{\mathrm{SAT}} \subseteq \mathbf{PSPACE}$. Es ist offen, ob „=“ gilt.
2. $A \in \mathbf{P}^A$ für jede Sprache A, damit kann insbesondere $\mathbf{P}^A$ unentscheidbare Sprachen enthalten.
3. Ist A entscheidbar, so ist $\mathbf{NP}^A$ eine Familie entscheidbarer Sprachen.
4. Trivialerweise gilt $\mathbf{P}^A \subseteq \mathbf{NP}^A$.

Polynomiale Turing-Reduzierbarkeit

Seien A und B Sprachen. Dann ist A polynomial Turing-reduzierbar auf B (kurz $A \leq_T^P B$), falls $A \in \mathbf{P}^B$ gilt.

Mit dieser Definition gelten folgende Aussagen:

Lemma 2

Wenn $A \leq_m^P B$, dann $A \leq_T^P B$

Beweis: Sei f eine Polynomialzeitreduktion f von A auf B.

Berechne bei Eingabe x das Wort $f(x)$ und frage das Orakel für B, ob $f(x) \in B$ ($\Leftrightarrow x \in A$) gilt. □

Polynomiale Turing-Reduzierbarkeit

Lemma 3

Wenn $A \leq_T^P B$ und $B \in \mathbf{P}$, dann auch $A \in \mathbf{P}$ (d.h. $\mathbf{P}^{\mathbf{P}} = \mathbf{P}$).

Beweis: Es sei

- $M_1^?$ eine deterministische polynomial zeitbeschränkte OTM mit $A = L(M_1^B)$ (Zeitschranke $p_1(n)$) und
- M_2 eine deterministische polynomial zeitbeschränkte TM mit $B = L(M_2)$ (Zeitschranke $p_2(n)$).

Eine polynomial zeitbeschränkte deterministische TM für A arbeitet bei Eingabe x ($|x| = n$) wie folgt:

Polynomiale Turing-Reduzierbarkeit

- Lasse die Maschine $M_1^?$ auf der Eingabe x laufen.
- Wird im Laufe der Berechnung das Orakel gefragt, ob $w(o) \in B$ gilt, so muss $|w(o)| \leq p_1(n)$ gelten.
- Indem wir die Maschine M_2 mit Eingabe $w(o)$ starten, können wir in Zeit $p_2(p_1(n))$ überprüfen, ob $w(o) \in B$ gilt, und danach im Zustand o_J oder o_N die Berechnung von $M_1^?$ fortführen. □

Eine Welt wo **P** = **NP**

Der gleiche Beweis (wobei $M_1^?$ eine nichtdeterministische polynomial zeitbeschränkte OTM ist und M_2 eine deterministische polynomial platzbeschränkte TM ist) zeigt:

Lemma 4

$\mathbf{NP}^{\mathbf{PSPACE}} = \mathbf{PSPACE}$

Satz 5

Es gibt ein Orakel $A \subseteq \Sigma^*$ in **PSPACE** mit $\mathbf{P}^A = \mathbf{NP}^A$.

Beweis:

Betrachte eine **PSPACE**-vollständige Sprache, etwa $A = \mathrm{QBF}$.

Dann gilt $\mathbf{PSPACE} \subseteq \mathbf{P}^{\mathrm{QBF}} \subseteq \mathbf{NP}^{\mathrm{QBF}} \overset{\text{(L. 4)}}{\subseteq} \mathbf{PSPACE}$.

Also gilt $\mathbf{PSPACE} = \mathbf{P}^{\mathrm{QBF}} = \mathbf{NP}^{\mathrm{QBF}}$. □

Eine Welt wo $\mathbf{P} \neq \mathbf{NP}$

Satz 6 (Baker, Gill, Solovay, 1975)

Es gibt ein entscheidbares Orakel $B \subseteq \{0,1\}^*$ mit $\mathbf{P}^B \neq \mathbf{NP}^B$.

Beweis:

Das Orakel $B \subseteq \{0,1\}^*$ wird so definiert, dass gilt:

$$\forall n \geq 0 : \left|B \cap \left\{w \in \{0,1\}^* \mid |w| = n\right\}\right| \leq 1$$

Für eine Sprache $B \subseteq \{0,1\}^*$ sei

$$L_B = \left\{1^n \;\middle|\; \exists w \in B \text{ mit } |w| = n\right\}.$$

Dann gilt offensichtlich: $L_B \in \mathbf{NP}^B$.

Noch zu zeigen: Es gibt eine entscheidbare Sprache B mit $L_B \notin \mathbf{P}^B$.

Eine Welt wo **P** $\neq$ **NP**

Sei $M_1^?, M_2^?, \ldots$ eine effektive Aufzählung aller deterministischen OTMs (mit Eingabealphabet $\{1\}$) mit zusätzlicher polynomieller Zeitschranke.

Wir nehmen an, dass jedes $M_i^?$ in der Aufzählung unendlich oft vorkommt.

Dies kann z.B. dadurch erreicht werden, dass $M_{ij}^? := M_i^?$ aus einer ursprünglichen Aufzählung definiert wird und dann die OTMs aus $\{M_{ij}^? \mid i, j \in \mathbb{N}\}$ aufgezählt werden.

Wir definieren rekursiv das Orakel $B = \bigcup_{i \geq 0} B_i$ durch Mengen $B_0 \subseteq B_1 \subseteq B_2 \subseteq \cdots$ mit $B_i \subseteq \{w \in \{0,1\}^* \mid |w| \leq i\}$ und eine Ausnahmemengen $X_0 \subseteq X_1 \subseteq X_2 \subseteq \cdots$:

Eine Welt wo **P** $\neq$ **NP**

- Für $i = 0$ setze $B_0 = X_0 = \varnothing$.
- Für $i > 0$ simulieren wir die Rechnung von $M_i^?$ auf Input 1^i für $i^{\lceil \log i \rceil}$ Schritte.

  Wir nehmen an, dass B_{i-1} und X_{i-1} mit $B_{i-1} \cap X_{i-1} = \varnothing$ schon definiert sind.
  - Setze $X_i = X_{i-1}$.
  - Befragt $M_i^?$ im Laufe der Rechnung ein Wort w mit $|w| < i$, so wird die Rechnung entsprechend dem Orakel B_{i-1} fortgesetzt.
  - Befragt $M_i^?$ im Laufe der Rechnung ein Wort w mit $|w| \geq i$, so setze die Rechnung im nein-Zustand o_N fort und setze $X_i := X_i \cup \{w\}$.
  - Akzeptiert $M_i^?$ innerhalb der Zeitschranke $i^{\lceil \log i \rceil}$ die Eingabe 1^i oder kommt $M_i^?$ innerhalb der Zeitschranke $i^{\lceil \log i \rceil}$ zu keiner Entscheidung, so setze $B_i := B_{i-1}$.
  - Verwirft $M_i^?$ innerhalb der Zeitschranke $i^{\lceil \log i \rceil}$ die Eingabe 1^i, so betrachte das lexikographisch erste Wort b_i in $\{0,1\}^i \setminus X_i$. Setze dann

$$B_i = \begin{cases} B_{i-1} \cup \{b_i\} & \text{falls } b_i \text{ existiert} \\ B_{i-1} & \text{sonst} \end{cases}$$

Eine Welt wo $\mathbf{P} \neq \mathbf{NP}$

Es gilt mit $B = \bigcup_{i \geq 0} B_i$ und $X = \bigcup_{i \geq 0} X_i$:

- $B_i \subseteq \{w \in \{0,1\}^* \mid |w| \leq i\}$
- B ist entscheidbar, denn für $|w| = i$ gilt: $w \in B \Leftrightarrow w \in B_i$.
- $B_i \cap X_i = \varnothing$ für alle $i \geq 0$ und damit $B \cap X = \varnothing$.

Wir zeigen jetzt $L_B \notin \mathbf{P}^B$.

Angenommen, es wäre $L_B = L(M^B)$ für eine deterministische, polynomial zeitbeschränkte OTM $M^?$.

Dann gibt es ein Polynom $p(n)$ so, dass $M^?$ $p(n)$-zeitbeschränkt ist.

Wähle jetzt i genügend groß so, dass $\forall n \geq i : p(n) \leq n^{\lceil \log i \rceil}$ und $M^? = M_i^?$ gilt.

Dies ist möglich, da $M^?$ in der Aufzählung beliebig oft vorkommt.

Eine Welt wo **P** $\neq$ **NP**

1. Fall: $1^i \in L_B = L(M_i^B)$, d.h. M_i^B akzeptiert 1^i innerhalb der Zeitschranke $p(i) \leq i^{\lceil \log i \rceil}$.

Dann gilt $B_{i-1} = B_i$ nach Konstruktion von B und damit $1^i \notin L_B$ nach Definition von L_B. Widerspruch!

2. Fall: $1^i \notin L_B = L(M_i^B)$, d.h. M_i^B verwirft 1^i innerhalb der Zeitschranke $p(i) \leq i^{\lceil \log i \rceil}$.

Ist $\{0,1\}^i \setminus X_i \neq \emptyset$, so existiert ein $b_i \in B$ mit $|b_i| = i$ und damit $1^i \in L_B$. Widerspruch!

Es ist daher noch zu zeigen, dass für alle zuvor genügend groß gewählten i gilt: $\{0,1\}^i \setminus X_i \neq \emptyset$

Bei der Berechnung von $X_0, X_1, \ldots, X_i$ werden maximal

$$\sum_{j=1}^{i} j^{\lceil \log j \rceil} \leq i \cdot i^{\lceil \log i \rceil} \leq 2^{\log i + \lceil \log i \rceil^2}$$

Orakelanfragen gestellt.

Eine Welt wo $\mathbf{P} \neq \mathbf{NP}$

X_i enthält also maximal $2^{\log i + \lceil \log i \rceil^2}$ Wörter.

Für alle zuvor genügend groß gewählten i gilt nun

$$2^{\log i + \lceil \log i \rceil^2} < 2^i,$$

weshalb $\{0,1\}^i \setminus X_i$ nicht leer ist. □

Man kann sogar zeigen, dass die Menge aller Orakel $B \subseteq \{0,1\}^*$ mit $\mathbf{P}^B = \mathbf{NP}^B$ Maß 0 innerhalb der Menge aller Sprachen über $\{0,1\}$ hat (Bennett, Gill, 1981).

Eine Welt wo **P** $\neq$ **NP**

Was sagen Satz 5 und 6 über das **P**-versus-**NP**-Problem aus?

Viele Beweistechniken in der Komplexitätstheorie sind relativierbar.

Das bedeutet: Zeigt man für Komplexitätsklassen $\mathcal{C}_1$ und $\mathcal{C}_2$ mittels einer relativierbaren Beweistechnik $\mathcal{C}_1 = \mathcal{C}_2$ (bzw. $\mathcal{C}_1 = \mathcal{C}_2$), so kann der Beweis zu einem Beweis von $\mathcal{C}_1^A = \mathcal{C}_2^A$ (bzw. $\mathcal{C}_1^A = \mathcal{C}_2^A$) für ein beliebiges Orakel A verallgemeinert werden.

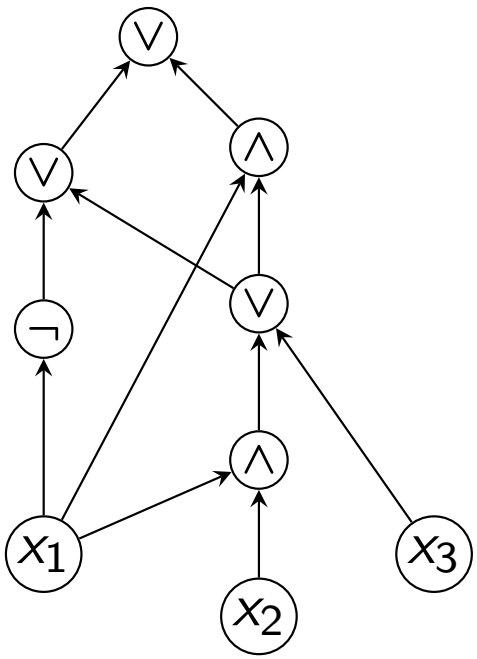
Wegen Satz 5 und 6 kann das **P**-versus-**NP**-Problem nicht mittels relativierbarer Beweistechniken gelöst werden.

Monotone Schaltkreise und der Satz von Razborov

Sei $C = (C_n)_{n \geq 0}$ eine Familie von booleschen Schaltkreisen, wobei C_n genau n Eingabegatter $x_1, \ldots, x_n$ hat.

Dann berechnet C_n eine boolesche Funktion $f_n : \{0,1\}^n \to \{0,1\}$ und C definiert die Sprache $L(C) = \bigcup_{n \geq 0} \{w \in \{0,1\}^n \mid f_n(w) = 1\}$.

Beispiel: Sei C_3 der folgende Schaltkreis:



Es gilt $f_3(1,0,0) = 0$ und $f_3(a,b,c) = 1$ für alle $(a,b,c) \in \{0,1\}^3 \setminus \{(1,0,0)\}$.

Also: $L(C) \cap \{0,1\}^3 = \{0,1\}^3 \setminus \{100\}$.

Monotone Schaltkreise und der Satz von Razborov

Die Familie $C = (C_n)_{n \geq 0}$ ist polynomiell, falls es ein Polynom $p(n)$ gibt, so dass C_n nur höchstens $p(n)$ viele Gatter hat.

Aus der Konstruktion im Satz von Cook (bzw. der **P**-Vollständigkeit von Circuit Value) folgt sehr einfach:

Lemma 7

Zu jeder Sprache $L \in \mathbf{P}$ gibt es eine polynomielle Familie $C = (C_n)_{n \geq 0}$ von Schaltkreisen mit $L = L(C)$.

Eine Strategie zum Beweis von $\mathbf{P} \neq \mathbf{NP}$ könnte also sein:

Zeige für eine **NP**-vollständige Sprache L, dass es keine polynomielle Familie $C = (C_n)_{n \geq 0}$ von Schaltkreisen gibt mit $L = L(C)$.

Beachte: Es gibt nicht-entscheidbare Sprachen mit polynomiellen Schaltkreisen.

Monotone Schaltkreise und der Satz von Razborov

Es ist jedoch leider extrem schwierig für ein konkretes Problem (Sprache) eine exponentielle Schranke für die Größe von Schaltkreisen zu zeigen.

Razborov gelang dies für das **NP**-vollständige Problem CLIQUE unter der Einschränkung auf monotone Schaltkreise.

Eine boolesche Funktion $f : \{0,1\}^n \to \{0,1\}$ heißt monoton, falls $f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n)$ gilt, sofern $x_i \leq y_i$ für alle $1 \leq i \leq n$.

D.h., durch einen Wechsel von 0 nach 1 in der Eingabe kann die Ausgabe nicht auf 0 zurückfallen.

Ein Schaltkreis heißt monoton, falls er keine NOT-Gatter besitzt.

Ein monotoner Schaltkreis berechnet eine monotone Funktion.

Umgekehrt gibt es zu jeder monotonen Funktion $f : \{0,1\}^n \to \{0,1\}$ (mit $n \geq 1$) einen monotonen Schaltkreis mit höchstens $2^n + 2^{n-1} - 2$ Gattern, der f berechnet.

Monotone Schaltkreise und der Satz von Razborov

Das folgende Problem CLIQUE ist **NP**-vollständig (Übung):

Eingabe: ungerichteter Graph $G = (V, E)$ und ein $k \geq 2$.

Frage: Gibt es in G eine Clique der Größe k, d.h. gibt es $C \subseteq V$ mit: $|C| \geq k$ und $\{u, v\} \in E$ für alle $u, v \in C$ mit $u \neq v$.

Ein ungerichteter Graph $G = (\{1, \ldots, n\}, E)$ wird im folgenden kodiert durch Bits $g_{ij} \in \{0, 1\}$ mit: $g_{ij} = 1 \iff \{i, j\} \in E$.

Sei $\mathrm{cl}_{n,k}$ die folgende monotone boolesche Funktion:

$$\mathrm{cl}_{n,k}((g_{ij})_{1 \leq i < j \leq n}) = 1 \iff \text{der durch } (g_{ij})_{1 \leq i < j \leq n} \text{ definierte Graph hat eine Clique der Größe } k$$

Satz (Razborov, 1985)

Es gibt eine Konstante $c_0 > 0$, sodass für jedes genügend große n und $k = \sqrt[4]{n}$ jeder monotone Schaltkreis für $\mathrm{cl}_{n,k}$ mindestens $r = n^{c_0 \cdot \sqrt[8]{n}}$ viele Gatter besitzt.

Sonnenblumen

Für den Beweis des Satzes von Razborov benötigen wir eine Eigenschaft von Sonnenblumen.

Definition

Eine Sonnenblume der Größe $p \geq 2$ mit Kern Z ist eine Menge $\{X_1, \ldots, X_p\}$ von p Mengen, für die gilt: $X_i \cap X_j = Z$ für alle $1 \leq i < j \leq p$. Die Mengen $X_i \setminus Z$ bezeichnen wir als Blütenblätter.

Lemma (Erdős, Rado)

Sei $p \geq 2$, $\ell \geq 1$ und $\mathcal{F} = \{X_1, \ldots, X_m\}$ eine Menge von m verschiedenen Mengen mit $m > (p-1)^\ell \cdot \ell!$ und $|X_i| \leq \ell$ für alle $1 \leq i \leq m$. Dann enthält $\mathcal{F}$ eine Sonnenblume der Größe p.

Sonnenblumen

Beweis: Induktion nach ℓ.

Induktionsanfang: $\ell = 1$.

$\mathcal{F}$ enthält dann $m > p - 1$ paarweise disjunkte Mengen der Kardinalität ≤ 1.

Dann bilden p viele dieser Mengen eine Sonnenblume der Größe p mit Kern $Z = \emptyset$.

Induktionsschritt: $\ell > 1$.

Sei $\mathcal{D} \subseteq \mathcal{F}$ eine maximale Teilmenge paarweise disjunkter Mengen.

Fall 1. $|\mathcal{D}| \geq p$.

Wähle analog zu $\ell = 1$ den Kern $Z = \emptyset$.

Fall 2. $|\mathcal{D}| \leq p - 1$.

Dann gilt $|D| \leq (p-1) \cdot \ell$ für $D = \bigcup \{X \mid X \in \mathcal{D}\}$.

Sonnenblumen

Da $\mathcal{D}$ maximal ist, gilt $\forall X \in \mathcal{F} \setminus \mathcal{D} : X \cap D \neq \emptyset$.

Aus $|\mathcal{F} \setminus \mathcal{D}| \geq m - (p-1)$ und $|D| \leq (p-1) \cdot \ell$ folgt, dass es ein $d \in D$ und mindestens

$$\frac{m-(p-1)}{(p-1)\cdot\ell} > \frac{(p-1)^\ell \cdot \ell! - (p-1)}{(p-1)\cdot\ell} = (p-1)^{\ell-1} \cdot (\ell-1)! - \frac{1}{\ell}$$

viele Mengen $X \in \mathcal{F} \setminus \mathcal{D}$ mit $d \in X$ gibt.

Für eine weitere Menge $D_0 \in \mathcal{D}$ gilt $d \in D_0$.

Also gibt es mehr als $(p-1)^{\ell-1} \cdot (\ell-1)!$ Mengen $X \in \mathcal{F}$ mit $d \in X$.

Sei $\mathcal{F}' = \{X \setminus \{d\} \mid d \in X \in \mathcal{F}\}$.

Nach Induktionshypothese enthält $\mathcal{F}'$ eine Sonnenblume $\{Y_1 \setminus \{d\}, \ldots, Y_p \setminus \{d\}\}$ der Größe p mit Kern Z'.

Damit ist $\{Y_1, \ldots, Y_p\}$ eine Sonnenblume der Größe p in $\mathcal{F}$ mit Kern $Z = Z' \cup \{d\}$.

Beweis des Satzes von Razborov

Annahme

Für jede Konstante $c_0 > 0$ gibt es für unendlich viele n einen monotonen Schaltkreis $C_{n,k}$ für die Funktion $\mathrm{cl}_{n,k}$ (mit $k = \sqrt[4]{n}$), welcher weniger als $r = n^{c_0 \cdot \sqrt[8]{n}}$ viele Gatter besitzt.

Sei $c_0 = 1/5$ im folgenden.

Wir werden für große n die obige Annahme zum Widerspruch führen.

Sei $C_{n,k}$ ein Schaltkreis für $\mathrm{cl}_{n,k}$, so wie in der Annahme gefordert.

Wir legen zunächst einige Parameter fest:

$$k := \sqrt[4]{n} \qquad p := \ell \cdot \log n$$
$$\ell := \sqrt[8]{n} \qquad M := (p-1)^{\ell} \cdot \ell!.$$

Der Logarithmus wird dabei zur Basis 2 gebildet und die Werte auf- oder abgerundet (die Rechnung ist genügend robust).

Beweis des Satzes von Razborov

Wir beschriften nun die Gatter von $C_{n,k}$ mit Mengen $\{X_1, \ldots, X_m\}$, wobei $X_i \subseteq \{1, \ldots, n\}$.

Sei $\mathcal{B}_g$ die Beschriftung für ein Gatter g.

Dann definieren wir für einen Graphen $G = (\{1, \ldots, n\}, E)$ den Wert $\mathcal{B}_g(G) \in \{0, 1\}$ durch:

$$\mathcal{B}_g(G) = 1 \quad \Longleftrightarrow \quad \exists X \in \mathcal{B}_g : \binom{X}{2} \subseteq E$$

Hierbei ist $\binom{X}{2} = \{\{u, v\} \mid u, v \in X, u \neq v\}$.

D.h.: $\mathcal{B}_g(G) = 1 \iff \mathcal{B}_g$ enthält eine Clique von G.

Beweis des Satzes von Razborov

Eine Beschriftung $\mathcal{B}$ ist zulässig, falls $|\mathcal{B}| \leq M$ und $|X| \leq \ell$ für alle $X \in \mathcal{B}$.

Z. B. ist $\mathcal{B} = \varnothing$ zulässig.

Die folgende reduce-Prozedur wandelt eine beliebige Beschriftung $\mathcal{B}$ in eine zulässige Beschriftung um:

1. Ersetze $\mathcal{B}$ durch $\mathcal{B}' = \{X \in \mathcal{B} \mid |X| \leq \ell\}$.
2. Solange noch $|\mathcal{B}'| > M$ gilt, tue folgendes:

   Suche eine beliebige Sonnenblume $\{X_1, \ldots, X_p\} \subseteq \mathcal{B}'$ mit Kern Z (existiert auf Grund des Lemmas von Erdős und Rado) und ersetze $\{X_1, \ldots, X_p\}$ durch Z.

   Diesen Vorgang bezeichnen wir als Pflücken der Sonnenblume.

Die zulässige Beschriftung, die wir so erhalten, nennen wir reduce($\mathcal{B}$).

Beweis des Satzes von Razborov

Nun definieren wir eine zulässige Beschriftung der Gatter von $C_{n,k}$ induktiv:

1. Für ein Eingabegatter g_{ij} setzen wir $\mathcal{B}_{g_{ij}} = \{\{i, j\}\}$.

   Dies ist eine zulässige Beschriftung, falls n genügend groß ist.

2. Sei g ein Gatter mit den Eingangsgattern f und h.

   Induktiv seien f mit $\mathcal{B}_f$ und h mit $\mathcal{B}_h$ jeweils zulässig beschriftet.

   Die Beschriftung $\mathcal{B}_g$ von g definieren wir als

$$\mathcal{B}_g = \begin{cases} \text{reduce}(\mathcal{B}_f \cup \mathcal{B}_h), & \text{falls } g = f \vee h \\ \text{reduce}(\mathcal{B}_f \cdot \mathcal{B}_h), & \text{falls } g = f \wedge h \end{cases}$$

   Dabei ist $\mathcal{B}_f \cdot \mathcal{B}_h = \{X \cup Y \mid X \in \mathcal{B}_f, Y \in \mathcal{B}_h\}$.

Am Ausgangsgatter *out* von $C_{n,k}$ erhalten wir so eine zulässige Beschriftung $\mathcal{B}_{out}$.

Beweis des Satzes von Razborov

Mit $C_{n,k}(g, G)$ bezeichnen wir den Wert des Gatters g in $C_{n,k}$ bei Eingabe G (ein Graph mit n Knoten).

Es gilt also $C_{n,k}(out, G) = C_{n,k}(G) = 1$ genau dann, wenn G eine Clique der Größe k enthält.

Behauptung 1: Für jeden Graphen $G = (\{1, \ldots, n\}, E)$ und jedes Eingabegatter g_{ij} gilt $\mathcal{B}_{g_{ij}}(G) = C_{n,k}(g_{ij}, G)$.

Beweis von Behauptung 1: Es gilt

$$\begin{aligned} \mathcal{B}_{g_{ij}}(G) = 1 &\iff \{i,j\} \text{ ist Clique von } G \\ &\iff \{i,j\} \in E \\ &\iff C_{n,k}(g_{ij}, G) = 1. \end{aligned}$$

Beweis des Satzes von Razborov

Behauptung 2: $\mathcal{B}_{out} \neq \emptyset$, falls n genügend groß ist.

Beweis von Behauptung 2:

Für $P \subseteq \{1, \ldots, n\}$ mit $|P| = k$ definieren wir den positiven Testgraphen

$$G_P = (\{1, \ldots, n\}, \binom{P}{2}).$$

Also gilt $C_{n,k}(G_P) = 1$, da P eine Clique der Größe k ist.

Es gibt $\binom{n}{k}$ viele positive Testgraphen G_P.

Wir zeigen, dass mindestens ein $P \subseteq \{1, \ldots, n\}$ mit $|P| = k$ und $\mathcal{B}_{out}(G_P) = 1$ existiert.

Dies impliziert dann $\mathcal{B}_{out} \neq \emptyset$.

Um einen Widerspruch abzuleiten, nehmen wir an:

$$\forall P \subseteq \{1, \ldots, n\} \text{ mit } |P| = k : \mathcal{B}_{out}(G_P) = 0 \neq 1 = C_{n,k}(G_P).$$

Beweis des Satzes von Razborov

Wir stellen uns die Gatter von $C_{n,k}$ nun so aufgelistet vor, dass ein Gatter g stets nach seinen Eingangsgattern h und f kommt (topologische Sortierung).

Für jede Teilmenge $P \subseteq \{1, \ldots, n\}$ mit $|P| = k$ gibt es ein kleinstes Gatter g (bzgl. unserer Auflistung) mit

$$\mathcal{B}_g(G_P) = 0 \quad \text{und} \quad C_{n,k}(g, G_P) = 1.$$

Wegen Behauptung 1 kann g kein Eingabegatter g_{ij} sein.

Seien f und h die Eingangsgatter von g.

Also gilt: $\mathcal{B}_f(G_P) \geq C_{n,k}(f, G_P)$ und $\mathcal{B}_h(G_P) \geq C_{n,k}(h, G_P)$

Beweis des Satzes von Razborov

Angenommen, g ist ein OR-Gatter.

Dann gilt $1 = C_{n,k}(g, G_P) = C_{n,k}(f, G_P) \vee C_{n,k}(h, G_P)$.

Gelte o.B.d.A. $C_{n,k}(f, G_P) = 1$.

Mit $\mathcal{B}_f(G_P) \geq C_{n,k}(f, G_P) = 1$ folgt $\mathcal{B}_f(G_P) = 1$.

Sei etwa $X \in \mathcal{B}_f$ und $X \subseteq P$.

$\rightsquigarrow$ $X \in \mathcal{B}_f \cup \mathcal{B}_h$

$\rightsquigarrow$ $\exists X' \in \mathcal{B}_g = \text{reduce}(\mathcal{B}_f \cup \mathcal{B}_h) : X' \subseteq X \subseteq P$.

$\rightsquigarrow$ $\mathcal{B}_g(G_P) = 1$ — ein Widerspruch.

Also muss g ein AND-Gatter sein, d.h. $\mathcal{B}_g = \text{reduce}(\mathcal{B}_f \cdot \mathcal{B}_h)$.

Aus $C_{n,k}(g, G_P) = 1$ folgt $C_{n,k}(f, G_P) = C_{n,k}(h, G_P) = 1$.

$\rightsquigarrow$ $\mathcal{B}_f(G_P) = \mathcal{B}_h(G_P) = 1$.

Sei etwa $X \in \mathcal{B}_f$, $Y \in \mathcal{B}_h$, $X \subseteq P$ und $Y \subseteq P$.

$\rightsquigarrow$ $X \cup Y \subseteq P$ und $X \cup Y \in \mathcal{B}_f \cdot \mathcal{B}_h$.

Beweis des Satzes von Razborov

Wegen $\mathcal{B}_g(G_P) = 0$ kann keine Teilmenge von $X \cup Y$ in $\mathcal{B}_g$ sein.

Also gilt $|X \cup Y| \geq \ell + 1$ (sonst wäre nach jedem Pflücken einer Sonnenblume noch eine Teilmenge von $X \cup Y$ in der aktuellen Beschriftung).

Wir haben somit gezeigt: Wenn g das kleinste Gatter mit $\mathcal{B}_g(G_P) = 0$ und $C_{n,k}(g, G_P) = 1$ ist, dann

- ist g ein AND-Gatter und
- es existiert $Z \in \mathcal{B}_f \cdot \mathcal{B}_h$ mit $Z \subseteq P$ und $|Z| \geq \ell + 1$.

Definiere für ein AND-Gatter g:

$$\mathcal{P}_g = \{P \subseteq \{1, \ldots, n\} \mid |P| = k, g = \text{kleinste Gatter mit } \mathcal{B}_g(G_P) = 0 \text{ und } C_{n,k}(g, G_P) = 1\}$$

Beachte: Für verschiedene AND-Gatter g und h gilt $\mathcal{P}_g \cap \mathcal{P}_h = \emptyset$.

Beweis des Satzes von Razborov

Außerdem gilt für jedes AND-Gatter g: $|\mathcal{P}_g| \leq M^2 \cdot \binom{n-\ell-1}{k-\ell-1}$.

Begründung: Jede Menge $P \in \mathcal{P}_g$ lässt sich schreiben als $P = Z \uplus P'$ mit

- $Z \in \mathcal{B}_f \cdot \mathcal{B}_h$, $|Z| \geq \ell + 1$ ($\leq M^2$ Möglichkeiten) und
- $P' \subseteq \{1, \ldots, n\} \setminus Z$, $|P'| = k - |Z|$.

  Hierfür gibt es $\binom{n-|Z|}{k-|Z|} \leq \binom{n-\ell-1}{k-\ell-1}$ viele Möglichkeiten.

Andererseits gibt es für jede k-elementige Teilmenge P ein AND-Gatter g mit $P \in \mathcal{P}_g$. Da $C_{n,k}$ weniger als r viele AND-Gatter hat, folgt:

$$r \cdot M^2 \cdot \binom{n-\ell-1}{k-\ell-1} > (\text{Anzahl AND-Gatter von } C_{n,k}) \cdot M^2 \cdot \binom{n-\ell-1}{k-\ell-1}$$

$$\geq \sum_{g \text{ ist AND-Gatter von } C_{n,k}} |\mathcal{P}_g| = \binom{n}{k}$$

Beweis des Satzes von Razborov

Also gilt:

$$
\begin{aligned}
r &> \frac{1}{M^2} \cdot \frac{\binom{n}{k}}{\binom{n-\ell-1}{k-\ell-1}} = \frac{1}{M^2} \cdot \frac{n! \cdot (k-\ell-1)!}{k! \cdot (n-\ell-1)!} \\
&= \frac{1}{M^2} \cdot \frac{n(n-1)\cdots(n-\ell)}{k(k-1)\cdots(k-\ell)} \geq \frac{1}{M^2} \cdot \left(\frac{n-\ell}{k}\right)^{\ell+1} \\
&= \frac{1}{(p-1)^{2\ell} \cdot \ell!^2} \cdot \left(\frac{n-\ell}{k}\right)^{\ell+1} \geq \left(\frac{n-\ell}{p^2 \cdot \ell^2 \cdot k}\right)^{\ell+1} \\
&= \left(\frac{n-n^{1/8}}{n^{1/4} \cdot (\log n)^2 \cdot n^{1/4} \cdot n^{1/4}}\right)^{\ell+1} = \left(\frac{n-n^{1/8}}{n^{3/4} \cdot (\log n)^2}\right)^{\ell+1} \\
&= \left(\frac{n^{1/4}}{(\log n)^2} - \frac{1}{n^{5/8} \cdot (\log n)^2}\right)^{\ell+1} \geq n^{1/5 \cdot (\ell+1)} \geq n^{c_0 \sqrt[8]{n}}
\end{aligned}
$$

wobei $\geq$ gilt, falls n genügend groß ist.

Dies ist ein Widerspruch zu $r = n^{c_0 \sqrt[8]{n}}$, was Behauptung 2 beweist.

Beweis des Satzes von Razborov

Für die weitere Argumentation benötigen wir negative Testgraphen.

Eine Färbung $c : \{1, \ldots, n\} \to \{1, \ldots, k-1\}$ der Knotenmenge definiert den Graphen $G_c = (\{1, \ldots, n\}, E)$ mit $E = \{\{i, j\} \mid c(i) \neq c(j)\}$.

An den Eingabegattern gilt: $C_{n,k}(g_{ij}, G_c) = 1 \iff c(i) \neq c(j)$.

Der Graph G_c ist $(k-1)$-färbbar, also gilt $C_{n,k}(G_c) = 0$.

Es gibt $(k-1)^n$ Färbungen, aber sehr viel weniger negative Testgraphen.

Sei $c : \{1, \ldots, n\} \to \{1, \ldots, k-1\}$ im folgenden eine zufällig und gleichverteilt gewählte Färbung (jede Färbung hat Wahrscheinlichkeit $(k-1)^{-n}$

Für $Z \subseteq \{1, \ldots, n\}$ sei $R(Z)$ das Ereignis, dass zwei Knoten aus Z gleich gefärbt werden (repeated color):

$$R(Z) \iff \exists i, j \in Z : i \neq j \wedge c(i) = c(j)$$

Beweis des Satzes von Razborov

Für eine Beschriftung $\mathcal{B}$ gilt damit:

$$\mathcal{B}(G_c) = 1 \iff \mathcal{B} \text{ enthält eine Clique von } G_c \iff \exists Z \in \mathcal{B} : \neg R(Z).$$

Behauptung 3: Für jede zulässige Beschriftung $\mathcal{B} \neq \varnothing$ gilt $\Pr(\mathcal{B}(G_c) = 1) \geq 1/2$.

Beweis von Behauptung 3:

Für $i, j \in \{1, \ldots, n\}$ mit $i \neq j$ gilt $\Pr(c(i) = c(j)) = \frac{(k-1)^{n-2} \cdot (k-1)}{(k-1)^n} = \frac{1}{k-1}$.

Sei $Z \in \mathcal{B}$ mit $|Z| \leq \ell$. Dann gilt

$$\Pr(R(Z)) \leq \binom{|Z|}{2} \cdot \frac{1}{k-1} \leq \frac{\ell(\ell-1)}{2(k-1)} = \frac{\ell(\ell-1)}{2(\ell^2-1)} = \frac{\ell}{2(\ell+1)} \leq \frac{1}{2}.$$

Also gilt $\Pr(\neg R(Z)) \geq 1/2$, was Behauptung 3 zeigt.

Beweis des Satzes von Razborov

Beachte:

- Aus Behauptung 2 ($\mathcal{B}_{out} \neq \emptyset$) und 3 folgt $\Pr(\mathcal{B}_{out}(G_c) = 1) \geq 1/2$.
- Andererseits gilt $C_{n,k}(G_c) = 0$ für alle Färbungen c.

Beweis des Satzes von Razborov

Behauptung 4: Sei $\{Z_1, \ldots, Z_p\}$ eine Sonnenblume mit Kern Z und $|Z_i| \leq \ell$ für alle $1 \leq i \leq p$. Dann gilt

$$\Pr(R(Z_1) \wedge \cdots \wedge R(Z_p) \wedge \neg R(Z)) \leq (1/2)^p.$$

Beweis von Behauptung 4: Sei inj_Z die Menge aller injektiven Färbungen $f : Z \to \{1, \ldots, k-1\}$.

Dann gilt ($c{\restriction}_Z$ ist die Einschränkung der zufälligen Färbung c auf Z):

$$\begin{aligned}
\Pr\left(\bigwedge_{i=1}^{p} R(Z_i) \wedge \neg R(Z)\right) &= \Pr\left(\bigvee_{f \in \mathrm{inj}_Z} \left(\bigwedge_{i=1}^{p} R(Z_i) \wedge c{\restriction}_Z = f\right)\right) \\
&= \sum_{f \in \mathrm{inj}_Z} \Pr\left(\bigwedge_{i=1}^{p} R(Z_i) \wedge c{\restriction}_Z = f\right) \\
&= \sum_{f \in \mathrm{inj}_Z} \Pr\left(\bigwedge_{i=1}^{p} R(Z_i) \;\middle|\; c{\restriction}_Z = f\right) \cdot \Pr(c{\restriction}_Z = f)
\end{aligned}$$

Beweis des Satzes von Razborov

Beachte: Unter der Voraussetzung $c{\restriction}_Z = f$ gilt

$$R(Z_i) \iff R(Z_i \setminus Z) \ \vee \ c(Z_i \setminus Z) \cap f(Z) \neq \varnothing.$$

Die Mengen $Z_1 \setminus Z, \ldots, Z_p \setminus Z$ sind paarweise disjunkt. Daher sind die Ereignisse $R(Z_1), \ldots, R(Z_p)$ unter der Voraussetzung $c{\restriction}_Z = f$ unabhängig.

Damit erhalten wir:

$$\sum_{f \in \mathrm{inj}_Z} \Pr\left(\bigwedge_{i=1}^{p} R(Z_i) \;\middle|\; c{\restriction}_Z = f \right) \cdot \Pr(c{\restriction}_Z = f)$$

$$= \sum_{f \in \mathrm{inj}_Z} \Pr(c{\restriction}_Z = f) \cdot \prod_{i=1}^{p} \Pr(R(Z_i) \mid c{\restriction}_Z = f)$$

$$\leq \sum_{f \in \mathrm{inj}_Z} \Pr(c{\restriction}_Z = f) \cdot \prod_{i=1}^{p} \Pr(R(Z_i))$$

$$= \prod_{i=1}^{p} \Pr(R(Z_i)) \cdot \sum_{f \in \mathrm{inj}_Z} \Pr(c{\restriction}_Z = f) \ \leq \ (1/2)^p,$$

Beweis des Satzes von Razborov

Beachte hierbei:

- Für $|Z_i| = y$ und $|Z| = z \leq y$ zeigen elementare Umformungen

$$\Pr(\neg R(Z_i)) = \frac{\binom{k-1}{y} \cdot y!}{(k-1)^y} \leq \frac{\binom{k-1-z}{y-z} \cdot (y-z)!}{(k-1)^{y-z}} = \Pr(\neg R(Z_i) \mid c{\restriction}_Z = f).$$

  Also gilt

$$\begin{aligned} \Pr(R(Z_i)) = 1 - \Pr(\neg R(Z_i)) &\geq 1 - \Pr(\neg R(Z_i) \mid c{\restriction}_Z = f) \\ &= \Pr(R(Z_i) \mid c{\restriction}_Z = f) \end{aligned}$$

- $\sum_{f \in \mathrm{inj}_Z} \Pr(c{\restriction}_Z = f) \leq 1$

- $\Pr(R(Z_i)) \leq 1/2$, falls $|Z_i| \leq \ell$ (siehe Beweis von Behauptung 3)

Dies beweist Behauptung 4.

Beweis des Satzes von Razborov

Wir wissen bereits, dass für mindestens die Hälfte aller Färbungen c gilt: $\mathcal{B}_{out}(G_c) = 1 \neq 0 = C_{n,k}(G_c)$.

Sei c eine Färbung mit $\mathcal{B}_{out}(G_c) = 1$ und $C_{n,k}(G_c) = 0$.

Sei g das kleinste Gatter mit $\mathcal{B}_g(G_c) = 1$ und $C_{n,k}(g, G_c) = 0$.

Wegen Behauptung 1 kann g kein Eingabegatter g_{ij} sein.

Seien also f und h die Eingabegatter von g.

Also gilt $\mathcal{B}_f(G_c) \leq C_{n,k}(f, G_c)$ und $\mathcal{B}_h(G_c) \leq C_{n,k}(h, G_c)$.

Beweis des Satzes von Razborov

Fall 1: g ist ein AND-Gatter.

$\rightsquigarrow\ 0 = C_{n,k}(g, G_c) = C_{n,k}(f, G_c) \wedge C_{n,k}(h, G_c)$.

Gelte o.B.d.A. $C_{n,k}(f, G_c) = 0$.

Mit $\mathcal{B}_f(G_c) \leq C_{n,k}(f, G_c)$ folgt $\mathcal{B}_f(G_c) = 0$.

Also ist keine Menge aus $\mathcal{B}_f$ eine Clique von G_c, d. h. $\forall X \in \mathcal{B}_f : R(X)$.

Also gilt für die Menge $\mathcal{B}'_g = \mathcal{B}_f \cdot \mathcal{B}_h$: $\forall Z \in \mathcal{B}'_g : R(Z)$.

Beweis des Satzes von Razborov

Fall 2: g ist ein OR-Gatter.

$\rightsquigarrow$ $0 = C_{n,k}(g, G_c) = C_{n,k}(f, G_c) \vee C_{n,k}(h, G_c)$.

$\rightsquigarrow$ $C_{n,k}(f, G_c) = C_{n,k}(h, G_c) = 0$.

$\rightsquigarrow$ $\mathcal{B}_f(G_c) = \mathcal{B}_h(G_c) = 0$

$\rightsquigarrow$ $\forall X \in \mathcal{B}_f : R(X)$ und $\forall Y \in \mathcal{B}_h : R(Y)$

Also gilt für die Menge $\mathcal{B}'_g = \mathcal{B}_f \cup \mathcal{B}_h$: $\forall Z \in \mathcal{B}'_g : R(Z)$.

In beiden Fällen gilt also $\forall Z \in \mathcal{B}'_g : R(Z)$.

Nun entsteht die Beschriftung $\mathcal{B}_g$ aus $\mathcal{B}'_g$ durch folgende 2 Schritte:

(1) Entfernen aller Mengen mit mehr als ℓ Elementen.

(2) Wiederholtes Pflücken von Sonnenblumen der Größe p.

Nach Schritt (1) gilt immer noch $R(Z)$ für alle Mengen in der Beschriftung.

Beweis des Satzes von Razborov

Da am Ende aber $\mathcal{B}_g(G_c) = 1$ gilt, muss durch eine der Pflückungen eine Clique von G_c in die Beschriftung kommen.

Insgesamt gibt es am Gatter g nur maximal $M^2/(p-1)$ viele Pflückungen ($\mathcal{B}'_g$ enthält $\leq M^2$ viele Elemente und jede Pflückung reduziert die Anzahl der Mengen in der Beschriftung um $p-1$).

Betrachte die erste Pflückung bei der eine Clique in die Beschriftung von g kommt.

Angenommen die Sonnenblume $\{Z_1, \ldots, Z_p\}$ wird dabei durch den Kern Z ersetzt, d. h.:

- Der Kern Z ist eine Clique von G_c: $\neg R(Z)$
- $Z_1, \ldots, Z_p$ sind keine Cliquen von G_c: $R(Z_1), \ldots, R(Z_p)$

Nach Behauptung 4 ($\Pr(R(Z_1) \wedge \ldots \wedge R(Z_p) \wedge \neg R(Z)) \leq (1/2)^p$) kann dies nur für höchstens $(1/2)^p \cdot (k-1)^n$ viele Färbungen c gelten.

Beweis des Satzes von Razborov

Da es weniger als r Gatter in $C_{n,k}$ gibt, erhalten wir

$$\begin{aligned}
\frac{1}{2}(k-1)^n &\leq \text{Anzahl der Färbungen } c \text{ mit } \mathcal{B}_{out}(G_c) = 1 \\
&\leq \left(\frac{1}{2}\right)^p \cdot (k-1)^n \cdot \frac{M^2}{p-1} \cdot \text{Anzahl Gatter von } C_{n,k} \\
&< \left(\frac{1}{2}\right)^p \cdot (k-1)^n \cdot \frac{M^2}{p-1} \cdot r
\end{aligned}$$

Dies bedeutet für genügend große n:

$$\begin{aligned}
r &> \frac{p-1}{M^2} \cdot 2^{p-1} = \frac{(p-1)\cdot 2^{p-1}}{((p-1)^\ell \cdot (\ell!))^2} \geq \frac{2^p}{(p\cdot\ell)^{2\ell}} \\
&= \frac{n^\ell}{(p\cdot\ell)^{2\ell}} = \left(\frac{n}{p^2\cdot\ell^2}\right)^\ell = \left(\frac{\sqrt{n}}{(\log n)^2}\right)^\ell \geq n^{1/5\ell} = n^{c_0 \sqrt[8]{n}} = r.
\end{aligned}$$

Dies ist ein Widerspruch. Damit ist der Satz von Razborov bewiesen. □

Beweis des Satzes von Razborov

Razborovs Resultat für CLIQUE hat ernsthafte Hoffnungen auf einen Beweis für $\mathbf{P} \neq \mathbf{NP}$ erzeugt.

Insbesondere wurde die folgende Vermutung aufgestellt:

Vermutung

Jede monotone Sprache $L \subseteq \{0,1\}^*$ in $\mathbf{P}$ kann durch eine polynomielle Familie von monotonen Schaltkreisen entschieden werden.

Zusammen mit Razborovs Resultat würde dies CLIQUE $\notin \mathbf{P}$ und damit $\mathbf{P} \neq \mathbf{NP}$ beweisen.

Leider ist die obige Vermutung falsch!

MATCHING ist das Problem, ob ein gegebener bipartiter Graph ein perfektes Matching hat. Dieses Problem liegt in $\mathbf{P}$ und ist monton.

Razborov selbst konnte seine Techniken für CLIQUE erweitern und zeigen, dass MATCHING nicht durch eine polynomielle Familie von monotonen Schaltkreisen entschieden werden kann.

Randomisierte Turingmaschinen

Definition

Eine randomisierte Turingmaschine (RTM) ist eine deterministische Turingmaschine M mit einem zusätzlichen Eingabeband (das Zufallsband), welches mit einem nach rechts unendlichen String über dem Alphabet $\{0,1\}$ gefüllt ist.

M ist $t(n)$-zeitbeschränkt, falls M bei Eingabe x und jeder Belegung des Zufallsbandes nach höchstens $t(|x|)$ Schritten terminiert.

Wir können dann das Zufallsband auf die Länge $t(|x|)$ einschränken.

Dann akzepiert M eine Eingabe x mit der Wahrscheinlichkeit

$$\operatorname*{Prob}_{r \in \{0,1\}^{t(|x|)}} [M \text{ akzeptiert bei Eingabe } x \text{ und Zufallsbandbelegung } r].$$

Randomisierte Komplexitätsklassen

Eine Sprache L gehört zur Klasse **RP** (randomized polynomial time), falls es eine polynomial zeitbeschränkte RTM M gibt, so dass für alle Eingaben x gilt:

- Wenn $x \in L$, dann gilt Prob$[M$ akzeptiert $x] \geq 1/2$.
- Wenn $x \notin L$, dann gilt Prob$[M$ akzeptiert $x] = 0$

Bemerkung: Die Wahrscheinlichkeit $1/2$ kann man durch jede Konstante $1 - \varepsilon$ ersetzen.

Eine Sprache L gehört zur Klasse **BPP** (bounded error probabilistic polynomial time), falls es eine polynomial zeitbeschränkte RTM M gibt, so dass für alle Eingaben x gilt:

- Wenn $x \in L$, dann gilt Prob$[M$ akzeptiert $x] \geq 2/3$.
- Wenn $x \notin L$, dann gilt Prob$[M$ akzeptiert $x] \leq 1/3$.

Bemerkung: Die Wahrscheinlichkeiten $1/3$ und $2/3$ kann man durch ε und $1 - \varepsilon$ für eine beliebige Konstante ersetzen.

Interaktive Beweissysteme

Definition IPS (Goldwasser, Micali, Rackoff 1985)

Ein interaktives Beweissystem (IPS) ist ein Paar (A, B) mit:

- A (Alice) ist eine beliebige Funktion

$$A : \bigcup_{i \geq 0} (\{0,1\}^*)^{1+2i} \to \{0,1\}^*.$$

- B (Bob) ist eine randomisierte Turingmaschine mit Ausgabe, dessen Eingabe von der Form $(x, \ldots)$ ($\ldots$ steht für eine Folge von Bitstrings) ist. Bobs Rechenzeit ist durch $q(|x|)$ für ein Polynom $q(n)$ beschränkt.
- Alice und Bob erhalten die gleiche Eingabe x und kommunizieren über ein gemeinsames Arbeitsband (Kommunikationsband) in $p(|x|)$ vielen Runden für ein Polynom $p(n)$.

Interaktive Beweissysteme

Fortsetzung Definition IPS (Goldwasser, Micali, Rackoff 1985)

- Runde i beginnt mit der Botschaft

$$a_i = A(x, a_1, b_1, \ldots, a_{i-1}, b_{i-1})$$

  von Alice an Bob, die auf das Kommunikationsband geschrieben wird.

- Danach sendet Bob die Nachricht

$$b_i = B(x, a_1, b_1, \ldots, a_{i-1}, b_{i-1}, a_i, r_1, \ldots, r_i)$$

  als Antwort an Alice. Hierbei ist r_i der Zufallsstring für Runde i.

- In der letzten Runde $p(|x|)$ entscheidet Bob, ob die Eingabe x akzeptiert oder abgelehnt wird.

Interaktive Beweissysteme

Bemerkungen:

- Da Bob in Polynomialzeit arbeitet, sind Bobs Nachrichten b_i an Alice automatisch polynomiell in der Länge von x beschränkt. Für Alice muss dies nicht unbedingt gelten.

  Aber: Ist Bob $q(n)$-zeitbeschränkt (für ein Polynom $q(n)$), so kann Bob nur die ersten $q(|x|)$ vielen Bits jeder Nachricht a_i von Alice an Bob lesen.

  Also können wir o.B.d.A. verlangen, dass $|a_i| \leq q(|x|)$ gilt. Wir setzen dies im folgenden stets voraus.

- Bob teilt seine Zufallsstrings r_i nicht Alice mit (private coins).

  Später werden wir aber sehen, dass man auch verlangen kann, dass Bob seine Zufallsstrings r_i Alice mitteilt (public coins).

Interaktive Beweissysteme

- Die letzte Nachricht $b_{p(n)}$ von Bob an Alice ist eigentlich redundant.

  Später wird es bequem sein, wenn Bob das letzte Bit von $b_{p(n)}$ auf 1 (bzw. 0) setzt, falls er x am Ende akzeptiert (bzw. ablehnt).

- Der Wahrscheinlichkeitsraum wird durch die Zufallsstrings $r_1, r_2, \ldots, r_{p(|x|)}$ erzeugt. Diese befinden sich zu Beginn auf dem Zufallsband. In Runde i hat Bob Zugriff auf das Anfangsstück $r_1, r_2, \ldots, r_i$.

Interaktive Beweissysteme und die Klasse **IP**

Definition

Ein IPS (A, B) entscheidet eine Sprache $L \subseteq \Sigma^*$, falls für alle Eingaben $x \in \Sigma^*$ gilt:

- Ist $x \in L$, so akzeptiert Bob die Eingabe mit einer Wahrscheinlichkeit $\geq (1 - 2^{-|x|})$.
- Ist $x \notin L$, so gibt es kein IPS (A', B) in dem Bob die Eingabe mit einer höheren Wahrscheinlichkeit als $2^{-|x|}$ akzeptiert.

IP ist die Menge der Sprachen, die durch ein IPS entschieden werden können:

$$\mathbf{IP} \quad = \quad \{L \subseteq \Sigma^* \mid \exists \text{ IPS } (A, B) : (A, B) \text{ entscheidet } L\}$$

Intuition zu interaktiven Beweissystemen

- Bob wird häufig auch als der Verifizierer bezeichnet, Alice als der Beweiser.
- Mittels eines Kommunikationsprotokolls soll herausgefunden werden, ob $x \in L$ oder nicht.
- Vollständigkeit: Wenn $x \in L$, dann überzeugt ein Beweiser, der sich an das Protokoll hält, den Verifizierer mit hoher Wahrscheinlichkeit $(1 - 2^{-|x|})$.
- Korrektheit: Wenn $x \notin L$, dann überzeugt jeder Beweiser (auch solche, die sich nicht an das Protokoll halten), den Verifizierer nur mit geringer Wahrscheinlichkeit $(2^{-|x|})$.

Die Klasse **IP**

Bemerkung: NP $\subseteq$ **IP**: Alice kann die erfolgreiche Rechnung einer **NP**-Maschine Bob zur Überprüfung vorlegen.

Für ein konkretes Problem aus **NP** kann Alice auch eine Lösung angeben: Für SAT übermittelt Alice beispielsweise eine erfüllende Belegung an Bob. Bob wertet die Formel unter dieser Belegung aus und akzeptiert, wenn sich als Wert der Formel „wahr“ ergibt.

Lemma 8

IP ist unter polynomialer Zeitreduktion abgeschlossen:

$$L \leq_m^p L' \text{ und } L' \in \mathbf{IP} \implies L \in \mathbf{IP}$$

Beweis: Gelte $L \leq_m^p L' \in \mathbf{IP}$ und sei f die Reduktion von L auf L'.

Dann berechnen sowohl Bob und Alice bei Eingabe x zunächst $f(x)$ und lassen dann das Protokoll für L' laufen. □

Beispiel: Nicht-Isomorphie von Graphen

Das Problem, ob zwei ungerichtete Graphen isomorph sind, liegt in **NP**.

Es ist aber weder bekannt, ob es in **P** liegt, noch, ob es **NP**-vollständig ist.

Es gibt allerdings Hinweise, die darauf hindeuten, dass dieses Problem nicht **NP**-schwierig ist.

Es ist nicht bekannt, ob das Problem der Nicht-Isomorphie von Graphen in **NP** liegt.

Satz 9 (Goldreich, Micali, Wigderson 1987)

Nicht-Isomorphie von Graphen gehört zu **IP**.

Beispiel: Nicht-Isomorphie von Graphen

Beweis:

Die Eingabe besteht aus 2 Graphen $G_0 = (V_0, E_0)$ und $G_1 = (V_1, E_1)$. o.B.d.A. $V_0 = V_1 = \{1, \ldots, n\}$ (wenn $|V_0| \neq |V_1|$, dann kann Bob sofort akzeptieren).

Das Protokoll zwischen Alice und Bob arbeitet wie folgt:

Bob wählt zunächst m Permutationen $\pi_i \in \mathrm{Perm}(\{1, \ldots, n\})$ und m Bits $b_i \in \{0, 1\}$ zufällig $(1 \leq i \leq m)$.

Dann übermittelt Bob an Alice in einer einzigen Runde die Liste $(\pi_1(G_{b_1}), \ldots, \pi_m(G_{b_m}))$.

Bob verlangt von Alice jetzt einen Bitstring $b'_1 b'_2 \cdots b'_m$ und akzeptiert, falls $b_i = b'_i$ für alle $i \in \{1, \ldots, m\}$.

Beispiel: Nicht-Isomorphie von Graphen

Fall 1: $G_0 \not\cong G_1$.

Dann kann Alice den Bitstring $b_1 b_2 \cdots b_m$ aus $(\pi_1(G_{b_1}), \ldots, \pi_m(G_{b_m}))$ rekonstruieren.

Also wird Bob mit Wahrscheinlichkeit 1 akzeptieren.

Fall 2: $G_0 \cong G_1$.

In diesem Fall sieht Alice lediglich eine Liste von m vielen Zufallspermutation von G_0.

Alice muss also einen von Bob zufällig gewählten und ihr nicht bekannten String der Länge m "erraten".

Die Wahrscheinlichkeit, dass ihr das gelingt, ist nur 2^{-m}, d. h. Bobs Antwort lautet (für jede Alice) "Isomorph" mit Wahrscheinlichkeit $1 - 2^{-m}$. □

Zero-Knowledge-Beweise

Das obige Protokoll für Nicht-Isomorphie von Graphen hat eine interessante zusätzliche Eigenschaft:

Falls $G_0 \cong G_1$, so teilt Alice Bob keinen Isomorphismus zwischen G_0 und G_1 mit. Mehr noch, Bob erhält keinerlei Information (zero-knowledge) über solch einen Isomorphismus.

Solche Beweissysteme nennt man auch Zero-Knowledge-Beweissysteme, sie spielen in der Kryptographie eine wichtige Rolle.

IP ⊆ PSPACE

Satz 10
IP ⊆ PSPACE

Beweis: Sei $L \in \mathbf{IP}$, o.B.d.A. $L \subseteq \{0,1\}^*$.

Fixiere eine fest gewählte randomisierte polynomiell zeitbeschränkte Turingmaschine B (Bob) und eine Eingabe $x \in \Sigma^*$ mit $|x| = n$.

Sei $p(n)$ das Polynom, das die Anzahl der Runden in dem Protokoll bestimmt.

O.B.d.A sei $q(n)$ ein Polynom, so dass alle Nachrichten von Alice sowie Bob $(a_1, b_1 \ldots, a_{p(n)}, b_{p(n)})$ sowie alle Zufallsstrings $r_1, \ldots, r_{p(n)}$ Länge genau $q(n)$ haben.

Eine mögliche Alice ist dann gegeben durch eine Funktion

$$A : \bigcup_{i=0}^{p(n)-1} \{0,1\}^{n+i\cdot 2q(n)} \to \{0,1\}^{q(n)}.$$

IP ⊆ PSPACE

Wir konstruieren nun eine optimale Alice in polynomiellen Platz.

Legt man eine mögliche Alice A sowie den Zufallsstring

$$\bar{r} = r_1 \cdots r_{p(n)} \in \{0,1\}^{p(n)q(n)}$$

fest, so ist dadurch die Folge der ausgetauschten Nachrichten

$$P(A, \bar{r}) := a_1 b_1 \cdots a_{p(n)} b_{p(n)}$$

(das zu A und r gehörende Protokoll) eindeutig bestimmt.

Falls Bob am Ende des Protokolls $P(A, \bar{r})$ die Eingabe x akzeptiert, so sagen wir, dass $(A, \bar{r})$ akzeptierend ist.

Ein Protokollpräfix ist ein Tupel $P \in \{0,1\}^{i \cdot q(n)}$ mit $0 \leq i \leq 2p(n)$.

$(A, \bar{r})$ passt zu diesem Protokollpräfix P, falls P ein Anfangsstück (Präfix) von $P(A, \bar{r})$ ist.

IP ⊆ PSPACE

Für eine mögliche Alice A und das Protokollpräfix P definieren wir

$$f(A,P) = |\{\bar{r} \in \{0,1\}^{p(n)q(n)} \mid (A,\bar{r}) \text{ ist akzeptierend und passt zu } P\}|.$$

Eine mögliche Alice ist optimal bzgl. P, falls $f(A,P) = f(P)$ für das folgende Maximum:

$$f(P) = \max\{f(A,P) \mid A \text{ ist eine mögliche Alice}\}.$$

Für den leeren Protokollpräfix λ gilt

$$f(A,\lambda) = |\{\bar{r} \in \{0,1\}^{p(n)q(n)} \mid (A,\bar{r}) \text{ ist akzeptierend}\}|.$$

und damit:

$$\frac{f(A,\lambda)}{2^{p(n)q(n)}} = \text{Prob}[\text{das IPS } (A,B) \text{ akzeptiert } x]$$

$$\frac{f(\lambda)}{2^{p(n)q(n)}} = \max\{\text{Prob}[(A,B) \text{ akzeptiert } x] \mid A \text{ ist eine mögliche Alice}\}$$

IP ⊆ PSPACE

Also gilt:

$$\begin{aligned} x \in L &\implies f(\lambda) \geq (1 - 2^{-n}) \cdot 2^{p(n)q(n)} \\ x \notin L &\implies f(\lambda) < 2^{-n} \cdot 2^{p(n)q(n)}. \end{aligned}$$

Ziel: Berechnung von $f(\lambda)$ in polynomiellen Platz.

Wir berechnen die Werte $f(P)$ rekursiv für jedes Protokollpräfix P.

Fall 1: $P = a_1 b_1 \cdots a_{p(n)} b_{p(n)}$ ein vollständiges Protokoll.

Fall 1.1: Bob lehnt am Ende des Protokolls P die Eingabe x ab (d.h. das letzte Bit von $b_{p(n)}$ ist 0).

Dann gilt $f(A, P) = 0$ für jede mögliche Alice und damit $f(P) = 0$.

IP ⊆ PSPACE

Fall 1.2: Bob akzeptiert am Ende des Protokolls P die Eingabe x.
(d.h. das letzte Bit von $b_{p(n)}$ ist 1).

Dann gilt für jede mögliche Alice

$$\begin{aligned} f(A,P) &= \left|\{\bar{r} \in \{0,1\}^{p(n)q(n)} \mid (A,\bar{r}) \text{ ist akzeptierend und passt zu } P\}\right| \\ &= \left|\{\bar{r} \in \{0,1\}^{p(n)q(n)} \mid P(A,\bar{r}) = P = a_1 b_1 \cdots a_{p(n)} b_{p(n)}\}\right| \end{aligned}$$

Dieser Wert wird maximal für jede mögliche Alice A mit

$$A(x, a_1, b_1, \ldots, a_{j-1}, b_{j-1}) = a_j$$

für alle $1 \leq j \leq p(n)$. Also gilt:

$$f(P) = \left|\{\bar{r} \in \{0,1\}^{p(n)q(n)} \mid \bar{r} = r_1 \cdots r_{p(n)} \text{ und für alle } 1 \leq j \leq p(n) \text{ gilt } B(x, a_1, b_1, \ldots, a_{j-1}, b_{j-1}, a_j, r_1, \ldots, r_j) = b_j\}\right|.$$

IP ⊆ PSPACE

Fall 2: Das Protokollpräfix ist von der Form $P = a_1 b_1 \cdots a_{i-1} b_{i-1} a_i$ mit $1 \leq i \leq p(n)$.

Induktionshyp.: Alle Werte $f(Pb_i)$ $(b_i \in \{0,1\}^{q(n)})$ sind bereits bekannt.

Behauptung 1: $f(P) = \sum_{b_i \in \{0,1\}^{q(n)}} f(Pb_i)$.

Wir zeigen zunächst $f(P) \leq \sum_{b_i \in \{0,1\}^{q(n)}} f(Pb_i)$

Beobachtung: Seien

- A und A' zwei mögliche Alices,
- $\bar{r} \in \{0,1\}^{p(n)q(n)}$ ein Zufallsstring und
- $b_i, b_i' \in \{0,1\}^{q(n)}$ Nachrichten (von Bob an Alice) mit $b_i \neq b_i'$.

Dann kann nicht $(A, \bar{r})$ passend zu Pb_i sein und gleichzeitig $(A', \bar{r})$ passend zu Pb_i' sein.

IP ⊆ PSPACE

Daher gilt für jede mögliche Alice A:

$$\begin{aligned} f(A,P) &= \left|\{\bar{r} \in \{0,1\}^{p(n)q(n)} \mid (A,\bar{r}) \text{ ist akzept. und passt zu } P\}\right| \\ &= \left| \biguplus_{b_i \in \{0,1\}^{q(n)}} \{\bar{r} \in \{0,1\}^{p(n)q(n)} \mid (A,\bar{r}) \text{ ist akzept.und passt zu } Pb_i\} \right| \\ &= \sum_{b_i \in \{0,1\}^{q(n)}} \left|\{\bar{r} \in \{0,1\}^{p(n)q(n)} \mid (A,\bar{r}) \text{ ist akzept. und passt zu } Pb_i\}\right| \\ &= \sum_{b_i \in \{0,1\}^{q(n)}} f(A,Pb_i). \end{aligned}$$

Sei nun A eine optimale Alice bzgl. P, d.h. $f(A,P) = f(P)$:

$$f(P) = f(A,P) = \sum_{b_i \in \{0,1\}^{q(n)}} f(A,Pb_i) \leq \sum_{b_i \in \{0,1\}^{q(n)}} f(Pb_i)$$

IP $\subseteq$ PSPACE

Nun zeigen wir $\sum_{b_i \in \{0,1\}^{q(n)}} f(Pb_i) \leq f(P)$.

Für jedes $b_i \in \{0,1\}^{q(n)}$ sei A_{b_i} eine Alice mit $f(Pb_i) = f(A_{b_i}, Pb_i)$.

Dann definieren wir eine Alice A' wie folgt:

- In den ersten i Runden schickt A' die durch den Protokollpräfix P festegelegten Nachrichten an Bob.
- Ist b_i Bobs Nachricht an Alice in Runde i, so verhält sich A' danach wie A_{b_i}.

IP ⊆ PSPACE

Damit gilt:

$$\sum_{b_i \in \{0,1\}^{q(n)}} f(Pb_i)$$

$$= \sum_{b_i \in \{0,1\}^{q(n)}} f(A_{b_i}, Pb_i)$$

$$= \sum_{b_i \in \{0,1\}^{q(n)}} \left|\{\bar{r} \in \{0,1\}^{p(n)q(n)} \mid (A_{b_i}, \bar{r}) \text{ ist akzept. und passt zu } Pb_i\}\right|$$

$$= \left| \biguplus_{b_i \in \{0,1\}^{q(n)}} \{\bar{r} \in \{0,1\}^{p(n)q(n)} \mid (A_{b_i}, \bar{r}) \text{ ist akzept. und passt zu } Pb_i\} \right|$$

$$\leq \left|\{\bar{r} \in \{0,1\}^{p(n)q(n)} \mid (A', \bar{r}) \text{ ist akzept. und passt zu } P\}\right|$$

$$= f(A', P)$$

$$\leq f(P)$$

IP ⊆ PSPACE

Fall 3: Das Protokollpräfix ist von der Form $P = a_1 b_1 \cdots a_{i-1}, b_{i-1}$ mit $1 \leq i \leq p(n)$.

Alice kann nun eine Antwort $a_i \in \{0,1\}^{q(n)}$ wählen.

Induktionshypothese: alle Werte $f(Pa_i)$ sind bereits bekannt.

Behauptung 2: $f(P) = \max\{f(Pa_i) \mid a_i \in \{0,1\}^{q(n)}\}$.

$$\begin{aligned} f(P) &= \max\{f(A,P) \mid A \text{ ist eine mögliche Alice}\} \\ &= \max\{f(A,Pa_i) \mid a_i \in \{0,1\}^{q(n)},\ A \text{ ist eine mögliche Alice}\} \\ &= \max_{a_i \in \{0,1\}^{q(n)}} \max\{f(A,Pa_i) \mid A \text{ ist eine mögliche Alice}\} \\ &= \max_{a_i \in \{0,1\}^{q(n)}} f(Pa_i) \end{aligned}$$

Behauptung 1 und 2 liefern einen rekursiven Algorithmus zur Berechnung von $f(\lambda)$.

IP ⊆ PSPACE

Platzbedarf des rekursiven Algorithmus:

- Die Rekursionstiefe ist $2p(n)$ (polynomiell).
- Für jeden lokalen Aufruf müssen folgende lokalen Variablen gespeichert werden: ein Protokollpräfix, ein String aus $\{0,1\}^{q(n)}$ (ein a_i oder b_i) sowie eine Zahl $\leq 2^{p(n)q(n)}$ (aktueller f-Wert) .

  Hierfür reicht polynomieller Platz.
- Für einen terminalen Aufruf mit $P = a_1 b_1 \cdots a_{p(n)} b_{p(n)}$ muss für alle Zufallsstrings $r = (r_1, \ldots, r_{p(n)})$ überprüft werden, ob r zu P passt.

  Hierfür muss für alle $1 \leq i \leq p(n)$ die Bedingung

$$b_i = B(x, a_1, b_1, \ldots, a_i, r_i)$$

  überprüft werden. Dies ist in Polynomialzeit möglich (uns reicht die Aussage, dass dies in polynomiellen Platz möglich ist). □

Der Satz von Shamir: **IP** = **PSPACE**

Satz 11 (Shamir, 1990)

IP = **PSPACE**.

Beweis: Noch zu zeigen: **PSPACE** $\subseteq$ **IP**.

Da **IP** unter Polynomialzeitreduktionen abgeschlossen ist und QBF **PSPACE**-vollständig ist, genügt es zu zeigen: QBF $\in$ **IP**.

Wir werden sogar ein IPS konstruieren, in dem Bob seine Zufallszahlen Alice mitteilt und am Ende (mit Wahrscheinlichkeit 1) „ja“ sagt, falls $x \in$ QBF gilt.

Sei also die Eingabe eine geschlossene quantifizierte Formel F über $0, 1, \wedge, \vee, \forall, \exists$ und den Literalen $\tilde{x_1}, \tilde{x_2}, \ldots$ mit $\tilde{x_i} \in \{x_i, \overline{x_i}\}$ gegeben.

O.B.d.A. wird Negation nur auf Variablen und nicht auf größere Teilformeln angewendet.

Beweis des Satzes von Shamir (Arithmetisierung)

F kann wie folgt in einen arithmetischen Ausdruck $a(F)$ umgeformt werden: Wir ersetzen

$\overline{x}$	durch	$(1-x)$
$\vee$	durch	$+$
$\wedge$	durch	$\cdot$ (Multiplikation)
$\forall x$	durch	$\prod_{x\in\{0,1\}}$
$\exists x$	durch	$\sum_{x\in\{0,1\}}$

Mit dieser Übersetzung gilt:

$$F \in \text{QBF} \iff a(F) > 0$$
$$F \notin \text{QBF} \iff a(F) = 0$$

Beweis des Satzes von Shamir (Arithmetisierung)

Beispiel:

Sei $F = \forall x \exists y \Big((x \vee \overline{y}) \wedge \exists z (\overline{x} \wedge z) \Big)$. Dann ist

$$
\begin{aligned}
a(F) &= \prod_{x=0,1} \Big(\sum_{y=0,1} \Big((x + (1-y)) \cdot \sum_{z=0,1} ((1-x) \cdot z) \Big) \Big) \\
&= \prod_{x=0,1} \Big(\sum_{y=0,1} \Big((x + (1-y)) \cdot (1-x) \Big) \Big) \\
&= \prod_{x=0,1} \Big((1-x^2) + (x-x^2) \Big) \\
&= 0
\end{aligned}
$$

Das bedeutet, dass die Auswertung der Formel F „falsch“ ergibt.

Beweis des Satzes von Shamir (Arithmetisierung)

Für die Negation $\neg F = \exists x \forall y \Big((\overline{x} \wedge y) \vee \forall z (x \vee \overline{z}) \Big)$ gilt:

$$
\begin{aligned}
a(\neg F) &= \sum_{x=0,1} \Big(\prod_{y=0,1} \Big((1-x) \cdot y + \prod_{z=0,1} (x+1-z) \Big) \Big) \\
&= \sum_{x=0,1} \prod_{y=0,1} \Big((1-x) \cdot y + x^2 + x \Big) \\
&= \sum_{x=0,1} (x^2 + x) \cdot (1 + x^2) \\
&= 4
\end{aligned}
$$

Die Auswertung der Formel $\neg F$ ergibt also „wahr“.

Beweis des Satzes von Shamir

Es stellt sich die Frage, wie groß die Zahl $a(F)$ werden kann.

Die Länge $|F|$ einer Formel definieren wir induktiv wie folgt:

$$\begin{aligned} |0| = |1| = |x| = |\overline{x}| &= 1 \\ |F \wedge G| = |F \vee G| &= |F| + |G| \\ |\forall x F| = |\exists x F| &= 1 + |F| \end{aligned}$$

Lemma 12

Für eine geschlossene quantifizierte Formel F gilt: $a(F) \leq 2^{2^{|F|}}$.

Beweis:

Zunächst eliminieren wir Quantoren, indem wir jedes Vorkommen von $\forall x G$ $(\exists x G)$ in F durch $G_{x=0} \wedge G_{x=1}$ $(G_{x=0} \vee G_{x=1})$ ersetzen.

Beweis des Satzes von Shamir

Sei F' die resultierende Formel.

Durch Induktion über den Aufbau von F zeigt wir $|F'| \leq 2^{|F|}$:

Für F von der Form 0, 1, x oder $\overline{x}$ gilt $F = F'$ und damit

$$|F'| = 1 \leq 2^1 = 2^{|F|}.$$

Für $F = G \circ H$ mit $\circ \in \{\wedge, \vee\}$ gilt

$$|F'| = |G' \circ H'| = |G'| + |H'| \leq 2^{|G|} + 2^{|H|} \leq 2^{|G|} \cdot 2^{|H|} = 2^{|G|+|H|} = 2^{|F|}.$$

Für $F = \forall x G$ gilt

$$|F'| = |G'_{x=0} \wedge G'_{x=1}| = |G'_{x=0}| + |G'_{x=1}| = 2|G'| \leq 2 \cdot 2^{|G|} = 2^{1+|G|} = 2^{|F|}.$$

Der Fall $F = \exists x G$ ist analog.

Beweis des Satzes von Shamir

Nun zeigen wir $a(F') \leq 2^{|F'|}$ durch Induktion über den Aufbau einer Quantoren-freien Formel F':

Für $F' = 0$ oder $F' = 1$ gilt $a(F') \leq 1 \leq 2^1 = 2^{|F'|}$.

Für $F' = G' \vee H'$ gilt

$$a(G' \vee H') = a(G') + a(H') \leq 2^{|G'|} + 2^{|H'|} \leq 2^{|G'|} \cdot 2^{|H'|} = 2^{|G'|+|H'|} = 2^{|F'|}.$$

Für $F' = G' \wedge H'$ gilt

$$a(G' \wedge H') = a(G') \cdot a(H') \leq 2^{|G'|} \cdot 2^{|H'|} = 2^{|G'|+|H'|} = 2^{|F'|}.$$

Insgesamt erhalten wir: $a(F) = a(F') \leq 2^{|F'|} \leq 2^{2^{|F|}}$. □

Beweis des Satzes von Shamir

Beispiel: Sei $F = \forall x_1 \ldots \forall x_k \exists y \exists z (y \vee z)$. Dann ist

$$
\begin{aligned}
a(F) &= \prod_{x_1=0,1} \cdots \prod_{x_k=0,1} \sum_{y=0,1} \sum_{z=0,1} (y+z) \\
&= \prod_{x_1=0,1} \cdots \prod_{x_k=0,1} \sum_{y=0,1} (2y+1) \\
&= \prod_{x_1=0,1} \cdots \prod_{x_k=0,1} (4) \\
&= 4^{2^k} \\
&\leq 2^{2^{k+4}} \\
&= 2^{2^{|F|}}
\end{aligned}
$$

Beweis des Satzes von Shamir

Problem: Die Zahl $2^{2^{|F|}}$ benötigt selbst bei binärer Kodierung noch $2^{|F|}$ Bits und kann deshalb nicht in einem IPS übermittelt werden.

Lösung: Rechnen modulo einer Primzahl.

Lemma 13

Für $n \geq 4$ gibt es im Intervall $[2^n, 2^{2n}]$ mindestens 2^n Primzahlen.

Beweis: Sei $\pi(n)$ die Anzahl der Primzahlen im Intervall $[2, n]$.

Mit elementaren Methoden kann man zeigen: $\pi(n) \geq \frac{n}{\log_2 n} - 2$, siehe z. B. Theorem 3.6.3 in *Primality Testing in Polynomial Time* von M. Dietzfelbinger, Springer 2004.

Also gibt es im Intervall $[2^n, 2^{2n}]$ mindestens

$$\left(\frac{2^{2n}}{2n} - 2\right) - (2^n - 2) = 2^n \cdot \left(\frac{2^n}{2n} - 1\right) \geq 2^n$$

viele Primzahlen, falls $n \geq 4$. □

Beweis des Satzes von Shamir

Sei im Folgenden $n = |F|$ und seien $p_1, \ldots, p_k$ die Primzahlen zwischen 2^n und 2^{2n} (d.h. $k \geq 2^n$).

Dann gilt:

$$m = p_1 \cdot p_2 \cdots p_k > (2^n)^{(2^n)} = 2^{n \cdot 2^n} \geq 2^{2^n}$$

und damit $m > a(F)$, da $a(F) \leq 2^{2^n}$.

Damit gilt:

$$F \notin \text{QBF} \iff a(F) \equiv 0 \bmod m \quad (\text{da } a(F) = 0)$$
$$F \in \text{QBF} \iff \exists \text{ Primzahl } p \in [2^n, 2^{2n}] : a(F) \not\equiv 0 \bmod p$$

Die zweite Zeile folgt aus der Tatsache, dass $m = p_1 \cdot p_2 \cdots p_k > a(F)$ gilt und $a(F)$ damit nicht alle p_i als Teiler haben kann, falls $a(F) > 0$.

Beweis des Satzes von Shamir

Falls nun $F \in \text{QBF}$ gilt, dann berechnet Alice die kleinste Primzahl $p \geq 2^n$ mit der Eigenschaft $a(F) \not\equiv 0 \bmod p$.

Sie sendet dann p an Bob, der überprüfen kann, ob p auch wirklich eine Primzahl ist (PRIM $\in \mathbf{P}$).

Beachte: p hat höchstens $2n$ viele Bits.

Alle weiteren Rechnungen werden von nun an modulo p durchgeführt.

O.B.d.A. beginne F mit einem Quantor: $F = QxF'(x)$ mit $Q \in \{\forall, \exists\}$.

Jetzt wird aus dem arithmetischen Ausdruck $a(F)$ das Polynom $a(F'(x))$ berechnet, indem in $a(F)$ das erste Zeichen $\prod$ bzw. $\sum$ gestrichen wird.

Problem: Der Grad dieses Polynoms kann exponentiell in n sein.

Lösung: einfache Formeln.

Beweis des Satzes von Shamir (einfache Formeln)

Definition (einfache Formel)

Eine Formel ist einfach, wenn für jede Teilformel $Qx\,G$ ($Q \in \{\forall, \exists\}$) jedes Vorkommen von x in G nur innerhalb eines $\forall$-Quantors liegt.

Lemma 14

Jede Formel lässt sich in polynomieller Zeit in eine äquivalente einfache Formel umwandeln.

Beweis:

Jede Teilformel $\forall y : G(x_1, \ldots, x_k, y)$ (wobei $x_1, \ldots, x_k, y$ alle freien Variablen in G sind) wird ersetzt durch:

$$\forall y\ \exists y_1 \cdots \exists y_k : \bigwedge_{i=1}^{k} x_i \leftrightarrow y_i \wedge G(y_1, \ldots, y_k, y).$$

□

Beweis des Satzes von Shamir (einfache Formeln)

Beispiel:

$F = \forall x_1 \forall x_2 \forall x_3 \exists x_4 G(x_1, x_2, x_3, x_4)$ (G Quantoren-frei) wird zu

$$\forall x_1 \forall x_2 \exists y_1 \big(x_1 \leftrightarrow y_1 \wedge \forall x_3 \exists z_1 \exists z_2 \big(y_1 \leftrightarrow z_1 \wedge x_2 \leftrightarrow z_2 \wedge \exists x_4 G(z_1, z_2, x_3, x_4) \big) \big)$$

Wenn die Ausgangsformel in Pränexnormalform ist ($Q_1 x_1 Q_2 x_2 \cdots Q_k x_k G$ mit G Quantoren-frei), so erhalten wir eine Formel F mit:

- F beginnt mit einem Quantor (falls $k \geq 1$).
- F ist einfach.
- Wenn $F_i = Q_i x_i G_i$ eine Teilformel von F ist, dann ist G_i entweder Quantoren-frei oder lässt sich schreiben als $G_i = H_i \wedge F_{i+1}$ mit H_i Quantoren-frei und $F_{i+1} = Q_{i+1} x_{i+1} G_{i+1}$.

Beweis des Satzes von Shamir (einfache Formeln)

Lemma 15

Sei $G = Qx : G'$ ($Q \in \{\exists, \forall\}$) eine einfache Formel der Länge n. Dann ist $a(G')$ ein Polynom welches Grad $\leq 2n$ in x hat.

Beweis:

Wir ersetzen zunächst in G' jede Teilformel der Gestalt $\forall y : H$, in der x frei vorkommt, durch $H_{y=0} \wedge H_{y=1}$.

Dies verdoppelt die Länge der Formel nur, da alle Teilformeln der Gestalt $\forall y : H$, in denen x frei vorkommt, disjunkt zueinander liegen.

Unsere neue (äquivalente) Formel hat also die Länge $2n$.

Es genügt somit zu zeigen: Sei G' eine Formel, so dass keine Teilformel der Gestalt $\forall y : H$ mit x frei in H existiert. Dann ist $a(G')$ ein Polynom in den freien Variablen von G' mit $\mathrm{Grad}_x(a(G')) \leq |G'|$.

Beweis des Satzes von Shamir (einfache Formeln)

Diese Aussage zeigen wir durch Induktion über den Aufbau von G':

Fall 1: G' ist eine Konstante oder eine (evtl. negierte) Variable: klar.

Fall 2: $G' = G_1 \vee G_2$. Dann gilt:

$$\mathrm{Grad}_x(a(G')) \leq \max\{\mathrm{Grad}_x(a(G_1)), \mathrm{Grad}_x(a(G_2))\} \leq \max\{|G_1|, |G_2|\} \leq |G'|$$

Fall 3: $G' = G_1 \wedge G_2$. Dann gilt:

$$\mathrm{Grad}_x(a(G')) = \mathrm{Grad}_x(a(G_1)) + \mathrm{Grad}_x(a(G_2)) \leq |G_1| + |G_2| = |G'|$$

Fall 4: $G' = \exists y : G_1$, o.B.d.A $y \neq x$. Dann gilt:

$$\begin{aligned} \mathrm{Grad}_x(a(G')) &\leq \max\{\mathrm{Grad}_x(a(G_1)_{y=0}), \mathrm{Grad}_x(a(G_1)_{y=1})\} \\ &\leq \mathrm{Grad}_x(a(G_1)) \\ &\leq |G_1| \leq |G'|. \end{aligned}$$

Fall 5: $G' = \forall y : G_1$. Dann kann x nicht frei in G_1 vorkommen.

Also gilt $\mathrm{Grad}_x(a(G')) = 0 \leq |G'|$. □

Beweis des Satzes von Shamir: das Protokoll

Erinnerung : Alice hat Bob bereits eine Primzahl $p \in [2^n, 2^{2n}]$ geschickt.
Es gilt $a(F) \not\equiv 0 \bmod p$, falls $F \in \mathrm{QBF}$.

Nun werden insgesamt m Runden gespielt, falls $m \leq n$ die Anzahl der Quantoren in F ist.

Sei $F = F_1 = Q_1 x_1 : G_1(x_1)$ und $p_1(x_1) = a(G_1(x_1)) \bmod p$
(ein Polynom in x_1 vom Grad höchstens $2n$).

Runde 1:

- Alice sendet Bob den Wert $a_1 = a(F_1) \bmod p$.
- Bob gibt "$F \notin \mathrm{QBF}$" aus, falls $a_1 = 0$, sonst verlangt er einen Beweis für die Korrektheit von a_1.
- Diesen gibt Alice, indem sie Bob das Polynom $p_1(x_1)$ mitteilt.
- Falls $Q_1 = \forall$: Bob überprüft, ob $a_1 \equiv p_1(0) \cdot p_1(1) \bmod p$.
- Falls $Q_1 = \exists$: Bob überprüft, ob $a_1 \equiv p_1(0) + p_1(1) \bmod p$.

Beweis des Satzes von Shamir: das Protokoll

- Nun wählt Bob zufällig eine Zahl $r_1 \in \mathbb{F}_p$, teilt diese Zahl Alice mit (public coins) und berechnet $p_1(r_1) \pmod p$.
- Der arithmetische Ausdruck $a(G_1(x_1))_{x_1=r_1}$ lässt sich schreiben als

$$a(G_1(x_1))_{x_1=r_1} = b_1 \cdot a(F_2(x_1))_{x_1=r_1},$$

  wobei F_2 der Teilausdruck von G_1 ist, der mit dem ersten Quantor von G_1 beginnt. Der Wert $b_1 \in \mathbb{F}_p$ kann von Bob aus G_1 und r_1 berechnet werden.
- Falls $b_1 = 0$: Bob akzeptiert, falls $p_1(r_1) = 0$ gilt, sonst lehnt er ab.
- Falls $b_1 \neq 0$: Bob berechnet $a_2 = p_1(r_1) \cdot b_1^{-1}$.

Beachte: Wenn Alice in Runde 1 das korrekte Polynom $p_1(x_1) = a(G_1(x_1))$ angibt, gilt $a(F_2(x_1))_{x_1=r_1} \equiv a_2 \pmod p$.

Beweis des Satzes von Shamir: das Protokoll

Sei $F_2(x_1) = Q_2 x_2 : G_2(x_1, x_2)$ und $p_2(x_2) = a(G_2(x_1, x_2))_{x_1=r_1} \pmod p$.

Runde 2:

- Bob will von Alice einen Beweis für $a(F_2(x_1))_{x_1=r_1} \equiv a_2 \pmod p$.
- Alice sendet hierfür Bob das Polynom $p_2(x_2)$.
- Bob überprüft wieder $a_2 \equiv p_2(0) \cdot p_2(1) \bmod p$ (falls $Q_2 = \forall$) bzw. $a_2 \equiv p_2(0) + p_2(1) \bmod p$ (falls $Q_2 = \exists$).
- Bob wählt die Zufallszahl r_2, teilt r_2 Alice mit und berechnet $p_2(r_2)$.
- Sei $a(G_2(x_1, x_2))_{x_1=r_1, x_2=r_2} = b_2 \cdot a(F_3(x_1, x_2))_{x_1=r_1, x_2=r_2}$, wobei F_3 der Teilausdruck von G_2 ist, der mit dem ersten Quantor von G_2 beginnt.
- Falls $b_2 = 0$: Bob akzeptiert, falls $p_2(r_2) = 0$, sonst lehnt er ab.
- Falls $b_2 \neq 0$: Bob berechnet $a_3 = p_2(r_2) \cdot b_2^{-1}$.
- In Runde 3 muss Alice $a(F_3(x_1, x_2))_{x_1=r_1, x_2=r_2} \equiv a_3 \bmod p$ beweisen.

Beweis des Satzes von Shamir: das Protokoll

Diese Protokoll wird für höchstens $m =$ (Anzahl der Quantoren in F) viele Runden durchgeführt.

Sei $F_m(x_1, \ldots, x_{m-1}) = Q_m x_m : G_m(x_1, \ldots, x_m)$ und

$$p_m(x_m) = a(G_m(x_1, \ldots, x_{m-1}, x_m))_{x_1=r_1,\ldots,x_{m-1}=r_{m-1}} \bmod p$$

mit G_m Quantoren-frei.

Runde m:

- Bob will von Alice einen Beweis für $a(F_m(x_1, \ldots, x_{m-1}))_{x_1=r_1,\ldots,x_{m-1}=r_{m-1}} \equiv a_m \pmod{p}$.
- Alice sendet hierfür Bob das Polynom $p_m(x_m)$.
- Bob überprüft wieder $a_m \equiv p_m(0) \cdot p_m(1) \bmod p$ (falls $Q_m = \forall$) bzw. $a_m \equiv p_m(0) + p_m(1) \bmod p$ (falls $Q_m = \exists$).
- Bob wählt eine weitere Zufallszahl r_m und akzeptiert genau dann, wenn $p_m(r_m) = a(G_m(x_1, \ldots, x_m))_{x_1=r_1,\ldots,x_m=r_m} \pmod{p}$.

Beweis des Satzes von Shamir: Analyse des Protokolls

Offensichtlich: Wenn $F \in \text{QBF}$, dann akzeptiert Bob mit Wahrscheinlichkeit 1.

Gelte nun $F \notin \text{QBF}$, d. h. $a(F) \equiv 0 \bmod p$.

Mit welcher Wahrscheinlickkeit kann eine Alice Bob vom Gegenteil $F \in \text{QBF}$ überzeugen?

Angenommen, Bob gibt schließlich "$F \in \text{QBF}$" aus.

Alice muss in Runde 1 einen Wert $a_1 \not\equiv a(F_1) \equiv 0 \bmod p$ an Bob schicken, sonst würde Bob "$F \notin \text{QBF}$" ausgeben.

Da Bob $a_1 \equiv p_1(0) \cdot p_1(1) \bmod p$ (falls $Q_1 = \forall$) bzw $a_1 \equiv p_1(0) + p_1(1) \bmod p$ (falls $Q_1 = \exists$) prüft, muss auch das von Alice an Bob gesendete Polynom $p_1(x_1)$ falsch sein, d. h. $p_1(x_1) \neq a(G_1(x_1))$.

Beweis des Satzes von Shamir: Analyse des Protokolls

Dann ist $p_1(x_1) - a(G_1(x_1))$ ein Polynom vom Grad $\leq 2n$ und hat damit höchstens $2n$ Nullstellen in dem Körper $\mathbb{F}_p$.

Es gilt also $p_1(r_1) = a(G_1(x_1))_{x_1=r_1}$ an höchstens $2n$ Stellen $r_1 \in \mathbb{F}_p$.

Wegen $p \geq 2^n$ folgt $\operatorname{Prob}_{r_1 \in \mathbb{F}_p}(p_1(r_1) \neq a(G_1(x_1))_{x_1=r_1}) \geq 1 - \frac{2n}{2^n}$.

Falls $p_1(r_1) \neq a(G_1(x_1))_{x_1=r_1}$ und $b_1 = 0$ in der ersten Runde, so wird Bob auf jeden Fall ablehnen.

Falls $p_1(r_1) \neq a(G_1(x_1))_{x_1=r_1}$ und $b_1 \neq 0$ in der ersten Runde (und das Protokoll in Runde 2 geht), so gilt $a_2 \neq a(F_2(x_1))_{x_1=r_1}$.

In diesem Fall muss Alice Bob mit Runde 2 beginnend von der falschen Identität $a_2 = a(F_2(x_1))_{x_1=r_1}$ überzeugen.

Diese Situation wiederholt sich ständig (höchstens $m \leq n$ mal).

Beweis des Satzes von Shamir: Analyse des Protokolls

In Runde i muss Alice Bob von einer Identität

$$a_i = a(F_i(x_1, \ldots, x_{i-1}))_{x_1=r_1,\ldots,x_{i-1}=r_{i-1}} \tag{3}$$

überzeugen.

Hierzu schickt sie ihm ein Polynom $p_i(x_i)$ (von Grad höchstens $2n$).

Bob stellt sicher, dass wenn (3) nicht gilt (Alice schummelt), dann auch

$$p_i(x_i) = a(G_i(x_1, \ldots, x_{i-1}, x_i))_{x_1=r_1,\ldots,x_{i-1}=r_{i-1}}$$

nicht gilt.

Dann wird Bob wieder mit Wahrscheinlichkeit mindestens $1 - \frac{2n}{2^n}$ ein $r_i \in \mathbb{F}_p$ mit $p_i(r_i) \neq a(G_i(x_1, \ldots, x_i))_{x_1=r_1,\ldots,x_i=r_i}$ wählen.

In diesem Fall geht es in Runde $i+1$ mit
$a_{i+1} \neq a(F_{i+1}(x_1, \ldots, x_i))_{x_1=r_1,\ldots,x_{i-1}=r_i}$.

Beweis des Satzes von Shamir: Analyse des Protokolls

Andererseits: Wenn Bob einmal (etwa in Runde i) eine Zufallszahl r_i mit

$$p_i(r_i) = a(G_i(x_1, \ldots, x_{i-1}, x_i))_{x_1=r_1,\ldots,x_{i-1}=r_{i-1},x_i=r_i}$$

trifft, dann gilt $a_{i+1} = a(F_{i+1}(x_1, \ldots, x_i))_{x_1=r_1,\ldots,x_{i-1}=r_i}$. Alice kann dann ohne erneute Täuschungsmanöver weiter spielen und Bob wird am Ende fälschlicherweise "$F \in$ QBF" ausgeben.

Es werden jedoch maximal $m \leq n$ viele Runden durchgeführt.

Wegen $(1-x)^n \geq 1 - nx$ für $0 \leq x \leq 1$ gilt:

$$\text{Prob}[\text{Bob gibt "}F \notin \text{QBF}\text{" aus}] \geq \left(1 - \frac{2n}{2^n}\right)^m \geq \left(1 - \frac{2n}{2^n}\right)^n \geq 1 - n \cdot \frac{2n}{2^n},$$

Also: $\text{Prob}[\text{Bob gibt "}F \in \text{QBF}\text{" aus}] \leq \frac{2n^2}{2^n}$.

Wird das Protokoll zweimal durchlaufen, so folgt für genügend große n:

$$\text{Prob}[\text{Bob gibt "}F \in \text{QBF}\text{" aus}] \leq \left(\frac{2n^2}{2^n}\right)^2 = \frac{4n^4}{2^{2n}} < 2^{-n} \qquad \square$$

Bemerkungen zu **IP** = **PSPACE**

- Die Inklusion **PSPACE** $\subseteq$ **IP** und damit **IP** = **PSPACE** gilt auch dann, wenn die Zufallsbits von Bob öffentlich sind, also Alice bekannt sind.
- Die Inklusion **PSPACE** $\subseteq$ **IP** gilt auch dann, wenn man Alice auf eine polynomiell platzbeschränkte Turingmaschine einschränkt.
- Schränkt man die Anzahl der Runden in interaktiven Beweissystemen auf eine Konstante ein, so erhält man die Komplexitätsklasse **AM** (Arthur-Merlin-Spiele). Es wird **AM** $\subsetneq$ **PSPACE** vermutet.
- **IP** = **PSPACE** ist nicht relativierbar: Es gibt ein Orakel A mit $\mathbf{IP}^A \neq \mathbf{PSPACE}^A$.

  In der Tat gilt dies sogar für fast alle Orakel A.

Erweiterungen von **IP**

- Ersetzt man Alice durch zwei Beweiser, mit denen Bob kommunizieren kann, so erhält man die Komplexitätsklasse **MIP**.

  Die beiden Beweiser können hierbei nicht miteinander kommunizieren.

  Analogie: Die beiden Beweiser sind zwei Verdächtige, die Bob (ein Polizeibeamter) von ihrer Unschuld zu überzeugen versuchen. Bob verhört die beiden Verdächtigen in getrennten Räumen.

  Babai, Fortnow, Lund 1991: **MIP** = **NEXPTIME**, wobei **NEXPTIME** = $\bigcup_{k\geq 1}$ **NTIME**(2^{n^k}).

- Werden diese beiden Beweiser zu Beginn mit einer beliebigen Zahl von verschränkten qubits ausgestattet, so erhält man die Klasse **MIP***.

  Ji, Natarajan, Vidick, Wright, Yuen 2020: **MIP*** = **RE** (die Klasse aller rekursiv aufzählbaren Sprachen!).

PRIM $\in$ **NP**

Es sei PRIM $= \{w \in 1\{0,1\}^* \mid w$ ist Binärkodierung einer Primzahl$\}$.

Im August 2002 bewiesen Agrawal, Kayal und Saxena PRIM $\in$ **P**.

Wir zeigen hier den folgenden schwächeren Satz:

Theorem 16
PRIM $\in$ **NP** $\cap$ **CoNP**.

Beweis:

(1) PRIM $\in$ **CoNP** ist einfach:

Bei Eingabe $w \in 1\{0,1\}^*$ raten wir $u, v \in 1\{0,1\}^+$ mit $|u|, |v| \leq |w|$ und überprüfen, ob $w = u \cdot v$ gilt.

Beachte: Aus $u, v \in 1\{0,1\}^*$ berechnet man das Produkt $u \cdot v$ in Polynomialzeit mit der Schulmultiplikationsmethode.

PRIM $\in$ **NP**

(2) PRIM $\in$ **NP**: Wir benötigen den kleinen Satz von Fermat:

Kleiner Satz von Fermat (ohne Beweis)

Eine Zahl $p \in \mathbb{N}$ ist eine Primzahl genau dann, wenn eine Zahl $x \in \{0, \ldots, p-1\}$ existiert mit:

$$x^{p-1} \equiv 1 \bmod p \text{ und } \forall i \in \{1, \ldots, p-2\} : x^i \not\equiv 1 \bmod p$$

Eine nichtdeterministischer Algorithmus, der $p \in$ PRIM überprüft:

1. Akzeptiere, falls $p = 2$ oder $p = 3$, sonst weiter mit (2).
2. Rate eine Primfaktorzerlegung für $p - 1 = p_1 p_2 \cdots p_k$, $k \geq 2$.
3. Überprüfe dabei, ob $p_j \geq 2$ und rekursiv, ob $bin(p_j) \in$ PRIM, für jedes $j \in \{1, \ldots, k\}$.
4. Rate nichtdeterministisch ein $x \in \{0, \ldots, p-1\}$ und überprüfe, ob $x^{p-1} \equiv 1 \bmod p$ gilt.
5. Verifiziere für alle $j \in \{1, \ldots, k\}$ die Ungleichung $x^{\frac{p-1}{p_j}} \not\equiv 1 \bmod p$.

PRIM $\in$ **NP**

(A) Wenn der Algorithmus die Eingabe p akzeptiert, dann ist $p \in$ PRIM.

Angenommen der Algorithmus akzeptiert die Eingabe $p > 3$.

Sei $x \in \{0, \ldots, p-1\}$ der in (4) geratene Wert. Es gilt $x^{p-1} \equiv 1 \bmod p$.

Angenommen es gilt $x^i \equiv 1 \bmod p$ für ein $i \in \{1, \ldots, p-2\}$.

Sei m der größte gemeinsame Teiler von $p-1$ und i.

Wegen $i \leq p-2$ gilt $m < p-1$, d.h. m ist echter Teiler von $p-1$

Euklid $\quad\Longrightarrow\quad$ $\exists \alpha, \beta \in \mathbb{Z} : m = \alpha \cdot (p-1) + \beta \cdot i$.

$\Longrightarrow\quad$ $x^m = x^{\alpha \cdot (p-1) + \beta \cdot i} = (x^{p-1})^\alpha \cdot (x^i)^\beta \equiv 1 \bmod p$

Da m ein echter Teiler von $p-1$ ist, gibt es ein p_j mit:
m ist Teiler von $\frac{p-1}{p_j}$.

$\Longrightarrow\quad$ $x^{\frac{p-1}{p_j}} \equiv 1 \bmod p$ Widerspruch zu (5)

PRIM $\in$ **NP**

(B) Wenn $p \in$ PRIM dann kann der Algorithmus bei Eingabe p akzeptieren.

Dies folgt direkt aus dem kleinen Satz von Fermat.

(C) Der Algorithmus kann so implementiert werden, dass er in Polynomialzeit arbeitet.

Betrachte für eine Primzahl p eine erfolgreiche Rechnung des Algorithmus.

Diese Rechnung kann durch einen Beweisbaum (für p) beschrieben werden, der für jeden rekursiven Aufruf einen Knoten enthält.

An den Blättern des Beweisbaums wird nur Schritt (1) ausgeführt, an den inneren Knoten des Beweisbaums werden die Schritte (1) – (5) ausgeführt.

PRIM $\in$ **NP**

Behauptung: Wenn p eine Primzahl ist, so gibt es einen Beweisbaum mit höchstens $2 \cdot \lfloor \log_2(p) \rfloor - 1$ vielen Knoten.

Beweis der Behauptung:

$p = 2$ oder $p = 3$: Dann gibt es einen Beweisbaum mit nur einem Knoten und $2 \cdot \lfloor \log_2(2) \rfloor - 1 = 2 \cdot \lfloor \log_2(3) \rfloor - 1 = 1$.

Sei nun $p > 3$ eine Primzahl und $p - 1 = p_1 \cdots p_k$, $k \geq 2$, eine Primfaktorzerlegung.

Der Beweisbaum für p besteht aus den Beweisbäumen für $p_1, \ldots, p_k$ und dem Wurzelknoten.
Induktion ergibt für die Anzahl der Knoten des Beweisbaums für p:

$$\begin{aligned} 1 + \sum_{i=1}^{k} (2 \cdot \lfloor \log_2(p_i) \rfloor - 1) &\leq 2 \cdot \lfloor \log_2(p_1 \cdot \ldots \cdot p_k) \rfloor - (k-1) \\ &\leq 2 \cdot \lfloor \log_2(p) \rfloor - 1, \text{ da } k \geq 2. \end{aligned}$$

PRIM $\in$ **NP**

Unser **NP**-Algorithmus, der $p \in \text{PRIM}$ überprüft, geht nun wie folgt vor:

- Rate einen Beweisbaum mit höchstens $2 \cdot \lfloor \log_2(p) \rfloor - 1$ vielen Knoten.

  An jedem Knoten des Beweisbaums steht eine Zahl $q \leq p$.

  Jedes Blatt ist mit 2 oder 3 beschriftet.

  Ist ein Knoten mit q beschriftet und sind die Kinder des Knotens mit $q_1, \ldots, q_k$ beschriftet, so muss gelten: $k \geq 2$ und $q - 1 = q_1 \cdot q_2 \cdots q_k$ (kann in Polynomialzeit überprüft werden).

- für jeden mit q beschrifteten inneren Knoten v wird noch eine Zahl $y \in \{0, \ldots, q-1\}$ geraten und überprüft, ob gilt: $y^{q-1} \equiv 1 \bmod q$ und $y^{(q-1)/q_j} \not\equiv 1 \bmod q$ für jedes mit q_j beschriftete Kind von v.

  Dies kann wieder in Polynomialzeit überprüft werden: $y^{q-1} \bmod q$ (sowie $y^{(q-1)/q_j} \bmod q$) kann durch iteriertes Quadrieren berechnet werden: Berechne nacheinander $y^1 \bmod q$, $y^2 \bmod q$, $y^4 \bmod q, \ldots$, $y^{2^i} \bmod q$, wobei 2^i die größte Zweierpotenz $\leq q - 1$ ist. $\square$