## 2.2 Wahrscheinlichkeitsverbesserung nach Chor/Goldreich

[Benny Chor, Oded Goldreich: On the power of two-point based sampling. *J. Complexity* **5**(1):96–106 (1989)]

### 2.2.1 Sprachen bzw. Entscheidungsprobleme

Wir betrachten randomisierte Algorithmen mit einseitigem Fehler, in einer abstrakten Version. (Für konkrete Beispiele siehe Vorlesung „Randomisierte Algorithmen“.)

Sei $L \subseteq \{0,1\}^*$ eine Sprache. Sei $(\varepsilon_n)_{n\geq 0}$ eine Folge von Zahlen in $[0,1)$. Ein randomisierter Polynomialzeitalgorithmus für $L$ mit einseitiger Fehlerschranke $(\varepsilon_n)_{n\geq 0}$ ist gegeben durch ein Polynom $p(n)$ und eine polynomialzeitberechenbare Relation $R \subseteq \bigcup_{n\geq 0}\big(\{0,1\}^n \times \{0,1\}^{p(n)}\big)$ mit folgender Eigenschaft. Definiere

$$W_x := \{y \in \{0,1\}^{p(|x|)} \mid R(x,y) \text{ gilt}\}, \text{ für } x \in \{0,1\}^*.$$

Dann gilt:

$$\begin{cases} x \notin L \Rightarrow W_x = \emptyset, \\ x \in L \Rightarrow |W_x| \geq (1-\varepsilon_n)2^{p(n)}. \end{cases} \qquad (*)$$

In einer solchen Situation können wir ein gegebenes $x \in \{0,1\}^*$ darauf testen, ob es in $L$ liegt, wie folgt: Wähle ein Element $y$ von $\{0,1\}^{p(|x|)}$ uniform zufällig und teste (in polynomieller Zeit), ob $R(x,y)$ gilt. Falls ja, akzeptiere (Ausgabe 1), falls nein, verwirf (Ausgabe 0). Dieses Vorgehen weist folgendes Verhalten auf:

$$\begin{cases} x \notin L \Rightarrow \mathbf{Pr}(\text{Ausgabe ist „verwirf“}) = 1, \\ x \in L \Rightarrow \mathbf{Pr}(\text{Ausgabe ist „verwirf“}) \leq \varepsilon_n, \text{ für } n = |x|. \end{cases}$$

Man beachte die Verwandtschaft zum Verhalten von nichtdeterministischen Turingmaschinen. (Wenn $x \notin L$, wird nie akzeptiert. Wenn $x \in L$, wird mit positiver Wahrscheinlichkeit akzeptiert.)

Unter Umständen ist $\varepsilon_n$ zwar kleiner als 1, aber sehr nahe an 1, zum Beispiel $\varepsilon_n = 1 - \frac{1}{n^d}$ für eine Konstante $d$. Dann ist die Erfolgswahrscheinlichkeit nicht sehr überzeugend.

Unser Ziel ist, die Fehlerwahrscheinlichkeit auf eine Konstante kleiner als 1 zu drücken. Der Standardansatz ist Wiederholung: Wenn $x$ mit $n = |x|$ gegeben ist, wähle $k = k_n$ und wähle $y_1, \ldots, y_k$ unabhängig und uniform zufällig aus $\{0,1\}^{p(n)}$. Teste, ob $R(x,y_i)$ gilt, für $i = 1, \ldots, k$. Falls dies niemals gilt: verwirf $x$. Falls für mindestens ein $i$ die Relation $R(x,y_i)$ gilt, akzeptiere.

*Bemerkung*: Dies kann man auch dadurch modellieren, dass man eine neue Polynomialzeitrelation $R' \subseteq \bigcup_{n\geq 0}\big(\{0,1\}^n \times (\{0,1\}^{p(n)})^{k_n}\big)$ definiert. Details: Selber überlegen.

Man überzeugt sich leicht davon, dass nun gilt:

$$\begin{cases} x \notin L \Rightarrow \mathbf{Pr}(\text{Ausgabe ist „verwirf“}) = 1, \\ x \in L \Rightarrow \mathbf{Pr}(\text{Ausgabe ist „verwirf“}) \leq \varepsilon_n^{k_n}, \text{ für } n = |x|. \end{cases}$$

Wenn $\varepsilon_n = 1 - n^{-d}$ nahe an 1 ist, ist die neue Fehlerwahrscheinlichkeit $\leq (1-n^{-d})^{k_n} \leq e^{-k_n/n^d}$. Mit $k_n = n^d$ erhalten wir Fehlerwahrscheinlichkeit $\leq e^{-1}$. Die Anzahl der benötigten Zufallsbits: $\geq n^d \cdot p(n)$. Wir möchten gerne die Fehlerwahrscheinlichkeit auf eine Konstante $< 1$ senken, ohne mehr als $O(p(n))$ Zufallsbits zu benötigen.

**Hashklasse**: Für jedes $n \geq 1$ sei $\mathcal{H}_n$ eine zweifach unabhängige Klasse von Hashfunktionen von $\{0,1\}^{p(n)}$ nach $\{0,1\}^{p(n)}$. Beispiel: Konvolutionsklasse $\mathcal{H}^{\text{conv}}_{\ell,\ell}$ mit $\ell = p(n)$. Anzahl Zufallsbits: $3p(n) - 1$.

Bemerkung: Man könnte auch den endlichen Körper $\mathrm{GF}(2^{p(n)})$ verwenden. Nachteil: Für die Arithmetik benötigt man ein irreduzibles Polynom vom Grad $p(n)$ über $\mathbb{Z}_2$. Ohne große Verluste könnte man auch einen Primzahlkörper $\mathbb{Z}_{q_n}$ mit $2^{p(n)} < q_n < 2^{p(n)+1}$ verwenden.

Wir nummerieren $\{0,1\}^{p(n)}$ als $x_0, \ldots, x_{2^{p(n)}-1}$ durch (mittels der Binärdarstellung).

**Algorithmus $\mathcal{A}_2$**:
Wir wählen eine Hashfunktion $h$ aus $\mathcal{H}_n$ zufällig ($3p(n) - 1$ Zufallsbits), wählen $k = k_n$ passend (siehe unten) und testen für $i = 0, \ldots, k-1$, ob für einen der Pseudozufallswerte $X_i = h(x_i)$ die Eigenschaft $X_i \in W_x$ (d. h. $R(x, X_i)$) gilt. Falls dies so ist, ist die Ausgabe „akzeptiere“, sonst „verwirf“.

Rechenzeit: $k$-mal die Rechenzeit des Algorithmus für $R$.

Verhalten auf $x$ mit $|x| = n$:

$$\begin{cases} x \notin L \Rightarrow \mathbf{Pr}(\text{Ausgabe ist „verwirf“}) = 1, \\ x \in L \Rightarrow \mathbf{Pr}(\text{Ausgabe ist „verwirf“}) \leq ??? \end{cases}$$

Wir möchten gerne $k_n$ so einstellen, dass an der Stelle „???“ der Wert $\frac{1}{2}$ steht. Dazu müssen wir die Fehlerwahrscheinlichkeit analysieren. Sei $W_x \neq \emptyset$. Definiere:

$$Y_i := [X_i \in W_x], \text{ für } 0 \leq i < k, \text{ und } Z := Z_k := Y_0 + \cdots + Y_{k-1}.$$

Wir definieren $\varrho_x := |W_x|/2^{p(n)}$. Dann ist $\varrho_x \geq 1 - \varepsilon_n$, und

$$\mathbf{E}(Y_i) = \mathbf{Pr}(X_i \in W_x) = \varrho_x \text{ und } \mathbf{Var}(Y_i) = \varrho_x(1 - \varrho_x).$$

Damit folgt mit der Linearität des Erwartungswertes und der Additivität der Varianz bei zweifach unabhängigen Zufallsvariablen:

$$\mathbf{E}(Z_k) = k\varrho_x \text{ und } \mathbf{Var}(Z_k) = k\varrho_x(1-\varrho_x).$$

Wir wenden die Ungleichung von Chebychev-Cantelli[1] an und erhalten:

$$\begin{aligned}\mathbf{Pr}(\text{Ausgabe „verwirf“}) &= \mathbf{Pr}(Z_k = 0)\\ &= \mathbf{Pr}(Z_k \le \mathbf{E}(Z_k) - \mathbf{E}(Z_k)) \le \frac{\mathbf{Var}(Z_k)}{\mathbf{Var}(Z_k) + \mathbf{E}(Z_k)^2}\\ &= \frac{k\varrho_x(1-\varrho_x)}{k\varrho_x(1-\varrho_x) + (k\varrho_x)^2} = \frac{1-\varrho_x}{1-\varrho_x+k\varrho_x}\\ &< \frac{1}{1+k\varrho_x} \le \frac{1}{1+k(1-\varepsilon_n)}.\end{aligned}$$

Mit $k = k_n \ge c/(1-\varepsilon_n)$ erreichen wir, dass diese Wahrscheinlichkeitsschranke $\le \frac{1}{1+c}$ wird.

Beispiel: $\varepsilon_n = 1-\delta_n = 1-n^{-d}$. Dann genügen $k_n = n^d$ Wiederholungen, um die Fehlerwahrscheinlichkeit auf $\frac{1}{2}$ zu drücken. Im folgenden Satz ist der Fall kleiner, konstanter $c$, also zum Beispiel $c = 1$, der interessanteste.

**Satz 2.2.1** (Chor-Goldreich)**.** *Sei $R$ wie oben polynomialzeitberechenbar, und sei $L$ eine Sprache mit $(*)$. Dann kann man mit $3p(n)$ Zufallsbits einen randomisierten Algorithmus gestalten, der die Relation $R$ höchstens $c/(1-\varepsilon_n)$-mal auswertet, auf $x \notin L$ stets verwirft und $x \in L$ mit Wahrscheinlichkeit $\ge \frac{c}{1+c}$ akzeptiert.*

Andere Formulierung: Relation $R$ kann durch eine Relation $R' \subseteq \bigcup_{n\ge 0}\big(\{0,1\}^n \times \{0,1\}^{3p(n)}\big)$ ersetzt werden, mit folgenden Eigenschaften: Für $x \notin L$ gibt es kein $y$ mit $R'(x,y)$, und für $x \in L$ erfüllt mindestens die Hälfte der $y \in \{0,1\}^{3p(n)}$ die Relation $R'(x,y)$. Die Auswertezeit von $R'$ ist um den Faktor $1/(1-\varepsilon_n)$ höher als die von $R$.

### 2.2.2 Funktionen bzw. Berechnungsprobleme

Wir betrachten die Berechnung von Funktionen $f\colon \{0,1\}^* \to \{0,1\}^*$, mit der Längenspezifikation $|f(x)| = m_n$ für $|x| = n$. Ein randomisierter Algorithmus zur Berechnung von $f$ kann jetzt durch ein Polynom $p(n)$ und eine Funktion $F\colon \bigcup_{n\ge 0}\big(\{0,1\}^n \times \{0,1\}^{p(n)}\big) \to \{0,1\}^*$ modelliert werden, wobei $|F(x,y)| = m_n$ gilt für $|x| = n$ und $|y| = p(n)$. Dabei ist $F(x,y)$ in polynomieller Zeit berechenbar. Es sei $(\varepsilon_n)_{n\ge 0}$ eine Folge in $[0, \frac{1}{2})$. Mit

$$W_x := \{y \in \{0,1\}^{p(|x|)} \mid F(x,y) = f(x)\}, \text{ für } x \in \{0,1\}^*,$$

[1] $\mathbf{Pr}(V \le \mathbf{E}(V) - t) \le \mathbf{Var}(V)/(\mathbf{Var}(V) + t^2)$, für $t \ge 0$.

soll gelten:

$$|W_x| \geq (1-\varepsilon_n)2^{p(n)}. \tag{1}$$

Wenn wir zu gegebenem $x$ mit $|x| = n$ ein $y$ aus $\{0,1\}^{p(n)}$ zufällig wählen, dann ist die Wahrscheinlichkeit, das Ergebnis $\neq f(x)$ zu erhalten, höchstens $\varepsilon_n$.

Um die Fehlerwahrscheinlichkeit auf eine Konstante $< \frac{1}{2}$ zu verringern, würde man den Algorithmus für $F$ $k_n$-mal unabhängig zufällig wiederholen. Die notwendige Wiederholungszahl ist $O((\frac{1}{2}-\varepsilon_n)^{-2})$ (siehe Vorlesung „Randomisierte Algorithmen“, Kapitel „Wahrscheinlichkeitsverbesserung“); entsprechend steigt die Zahl an benötigten Zufallsbits. Der Chor-Goldreich-Ansatz schafft auch hier Abhilfe.

Definiere $X_i$, $i = 0, \ldots, 2^{p(n)}-1$, wie oben. Dann berechne $F(x, X_i)$ für $i = 0, \ldots, k-1$, für $k = k_n$. Die Resultate seien $r_0, \ldots, r_{k-1} \in \{0,1\}^{m(n)}$. Die Ausgabe ist $r^*$, wenn $r^*$ in $(r_0, \ldots, r_{k-1})$ mehr als $\frac{1}{2}k$-mal vorkommt. Sonst ist die Ausgabe beliebig. Wir nennen diesen Algorithmus $\mathcal{A}'$, und schätzen die Fehlerwahrscheinlichkeit von $\mathcal{A}'$ ab.

Wir definieren nun $Y_i := [F(x, X_i) = f(x)]$, für $0 \leq i < k$, und $Z_k := Y_0 + \cdots + Y_{k-1}$. Mit $\varrho_x := |W_x|/2^{p(n)} \geq 1 - \varepsilon_n$ gilt

$$\mathbf{E}(Z_k) = k\varrho_x \text{ und } \mathbf{Var}(Z_k) = k\varrho_x(1-\varrho_x),$$

wie vorher. Mit der Chebychev-Cantelli-Ungleichung erhalten wir:

$$\begin{aligned}
\mathbf{Pr}(\text{Ausgabe ist } \neq f(x)) &= \mathbf{Pr}(Z_k \leq \frac{1}{2}k) \\
&= \mathbf{Pr}(Z_k \leq \mathbf{E}(Z_k) - (\mathbf{E}(Z_k) - \frac{1}{2}k)) \leq \frac{\mathbf{Var}(Z_k)}{\mathbf{Var}(Z_k) + (\mathbf{E}(Z_k) - \frac{1}{2}k)^2} \\
&= \frac{k\varrho_x(1-\varrho_x)}{k\varrho_x(1-\varrho_x) + (k(\varrho_x - \frac{1}{2}))^2} \\
&\leq \frac{\varrho_x(1-\varrho_x)}{\varrho_x(1-\varrho_x) + k(\frac{1}{2}-\varepsilon_n)^2} \\
&\leq \frac{\frac{1}{4}}{\frac{1}{4} + k(\frac{1}{2}-\varepsilon_n)^2} = \frac{1}{1 + k(1-2\varepsilon_n)^2}.
\end{aligned}$$

(Dabei wurde benutzt: $t(1-t) \leq \frac{1}{4}$, für $0 \leq t \leq 1$, und $\frac{a}{b} \leq \frac{a+z}{b+z}$, falls $0 < a \leq b$ und $z \geq 0$.) Mit $k = k_n \geq 1/(1-2\varepsilon_n)^2$ erreichen wir, dass diese Wahrscheinlichkeitsschranke $\leq \frac{1}{2}$ wird.

Beispiel: $\varepsilon_n = \frac{1}{2}(1-n^{-d})$. Dann genügen $k_n = n^{2d}$ Wiederholungen, um die Fehlerwahrscheinlichkeit auf $\frac{1}{2}$ zu drücken.

Der Rest der Diskussion ist wie im Sprachenfall in 2.2.1.

## 2.3 Pseudozufallszahlengenerator nach Noam Nisan

[Noam Nisan, Pseudorandom generators for space bounded computation, *Combinatorica* **12** (4) (1992) 449–461.]

Genau wie in 2.2 wollen wir die Erfolgswahrscheinlichkeit bei randomisierten Algorithmen für Sprachen verbessern. Zu Eingabelänge $n$ gehört die Menge $U = U_n = \{0,1\}^{p(n)}$ der Zufallsstrings. Sei $q = |U| = 2^{p(n)}$. Anders als in 2.2 werden wir voraussetzen, dass die Fehlerwahrscheinlichkeit maximal $\frac{1}{2}$ ist.

Die Hashklasse $\mathcal{H} \subseteq \{h \mid h\colon U \to U\}$ sei 2-fach unabhängig. Man kann sich $U = \mathrm{GF}(2^{p(n)})$ mit linearen Funktionen vorstellen ($2p(n)$ Zufallsbits) oder $U = U_n = \{0,1\}^{p(n)}$ als $\mathbb{Z}_2$-Vektorraum mit der Konvolutions-Hashklasse $\mathcal{H}^{\mathrm{conv}}_{k,\ell}$ mit $k = \ell = p(n)$ (benötigt $3p(n) - 1$ Zufallsbits).

Wir beginnen mit einem technischen Lemma, das auch über unsere Anwendung hinaus sehr nützlich ist.

**Lemma 2.3.1** (Hash-Mixing-Lemma)**.** *Sei $\varepsilon = 1/q^{1/3}$. Dann gilt für alle $A, B \subseteq U$: Es gibt eine Menge $\mathcal{H}' = \mathcal{H}'(A,B) \subseteq \mathcal{H}$ mit $|\mathcal{H}'| \geq (1-\varepsilon)|\mathcal{H}|$, so dass alle $h \in \mathcal{H}'$ „$(A,B)$-gut“ sind, das heißt Folgendes erfüllen:*

$$\left|\mathbf{Pr}_{y \in U}(y \in A \wedge h(y) \in B) - \frac{|A|}{|U|} \cdot \frac{|B|}{|U|}\right| \leq \varepsilon.$$

*Interpretation*: Man betrachtet das Rechteck $A \times B \subseteq U \times U$. Wenn man $y$ und $z$ unabhängig und zufällig aus $U$ wählt, landet man mit Wahrscheinlichkeit genau $\frac{|A|}{|U|} \cdot \frac{|B|}{|U|}$ in diesem Rechteck. Man benötigt $2 \log |U|$ Zufallsbits. Ein rudimentärer Mechanismus zur Einsparung von Zufallsbits ist es, $y$ zufällig zu wählen und $z = h(y)$ zu setzen, für eine feste Hashfunktion $h$. Dann benötigt man nur $\log |U|$ Zufallsbits. Das Hash-Mixing-Lemma sagt, dass für jedes Paar $(A,B)$ fast alle Hashfunktionen $h \in \mathcal{H}$ die Eigenschaft haben, dass die Wahrscheinlichkeit, dass das Paar $(y, h(y))$ in $A \times B$ landet, sehr nahe an $\frac{|A|}{|U|} \cdot \frac{|B|}{|U|}$ liegt.

Vorsicht beim Beweis! Es werden einerseits Hashfunktionen aus $\mathcal{H}$ und andererseits Elemente $y$ aus $U$ zufällig gewählt. Diese beiden Zufallsexperimente muss man gut unterscheiden.

*Beweis* von Lemma 2.3.1: Die Mengen $A$ und $B$ seien fest gewählt. (Viele Größen im folgenden Beweis hängen von $A$ und $B$ ab. Dies wird in der Notation unterdrückt.) Man wählt $h \in \mathcal{H}$ zufällig und beweist, dass $h$ nur mit Wahrscheinlichkeit höchstens $\varepsilon$ *nicht*

$(A, B)$-gut ist.

$$\begin{aligned}
&\mathbf{Pr}_{h\in\mathcal{H}}(h \text{ ist nicht } (A,B)\text{-gut}) \\
&= \mathbf{Pr}_{h\in\mathcal{H}}\left(\left|\mathbf{Pr}_{y\in U}(h(y)\in B \mid y\in A)\cdot\mathbf{Pr}(y\in A) - \frac{|A|\cdot|B|}{|U|^2}\right| > \varepsilon\right) \\
&= \mathbf{Pr}_{h\in\mathcal{H}}\left(\left|\mathbf{Pr}_{y\in A}(h(y)\in B)\cdot\frac{|A|}{|U|} - \frac{|A|\cdot|B|}{|U|^2}\right| > \varepsilon\right) \\
&= \mathbf{Pr}_{h\in\mathcal{H}}\left(\left|\mathbf{Pr}_{y\in A}(h(y)\in B)\cdot|A| - \frac{|A|\cdot|B|}{|U|}\right| > \varepsilon\cdot|U|\right)
\end{aligned}$$

Für jedes $y \in U$ definieren wir eine Indikatorfunktion $Z_y(h) = [h(y) \in B]$ auf dem Wahrscheinlichkeitsraum $\mathcal{H}$. Für jedes $h \in \mathcal{H}$ definieren wir dann:

$$Y(h) := \sum_{y\in A} Z_y(h) = \mathbf{Pr}_{y\in A}(h(y)\in B)\cdot|A|.$$

Damit ist $Y$ eine auf $\mathcal{H}$ definierte Zufallsvariable. Es gilt:

$$\mathbf{Pr}_{h\in\mathcal{H}}(h \text{ ist nicht } (A,B)\text{-gut}) = \mathbf{Pr}_{h\in\mathcal{H}}\left(\left|Y - \frac{|A|\cdot|B|}{|U|}\right| > \varepsilon\cdot|U|\right).$$

Diese Wahrscheinlichkeit wollen wir mit der Chebychev-Ungleichung abschätzen. Dazu müssen wir $\mathbf{E}(Y)$ und $\mathbf{Var}(Y)$ bestimmen. Die Zufallsvariable $Y$ ist Summe von $|A|$ vielen paarweise unabhängigen Zufallsvariablen $Z_y$, die Erwartungswert $\varrho = |B|/|U|$ haben. Daraus folgt

$$\mathbf{E}(Y) = |A|\varrho \quad \text{und} \quad \mathbf{Var}(Y) = |A|\varrho(1-\varrho) < |A|\varrho.$$

(Details: Vorlesung „Randomisierte Algorithmen“, Kapitel zu „Paarweise unabhängige Suche“.) Die Chebychev-Ungleichung liefert:

$$\mathbf{Pr}_{h\in\mathcal{H}}\left(\left|Y - \frac{|A|\cdot|B|}{|U|}\right| > \varepsilon\cdot|U|\right) \le \frac{\mathbf{Var}(Y)}{(\varepsilon\cdot|U|)^2}.$$

Weil $\varepsilon = 1/|U|^{1/3}$ und $\mathbf{Var}(Y) < |A||B|/|U| \le |U|$, kann man die rechte Seite durch $1/|U|^{1/3} = \varepsilon$ abschätzen. – Damit ist das Hash-Mixing-Lemma bewiesen. □

Nun sei $W \subseteq U$ eine Teilmenge, $\varrho = \frac{|W|}{|U|}$ die Dichte von $W$ in $U$.

Wir wollen mit wenigen Zufallselementen eine lange Folge von „pseudozufälligen“ Elementen von $U$ generieren, die dann darauf getestet wird, ob ein Element von $W$ darunter ist.

**Festlegung:** Wir nehmen $\varrho \ge \frac{1}{2}$ an.

Man vergleiche Abschnitt 2.2, um zu sehen, wozu das gut ist. Die Menge $W$ steht für $W_x \subseteq U = \{0,1\}^{p(n)}$. Falls die Mengen $W_x$ zu klein sind, muss man zunächst das Chor-Goldreich-Verfahren aus Abschnitt 2.2 anwenden, um eine Erfolgswahrscheinlichkeit von mehr als $\frac{1}{2}$ herzustellen.

Sei $\ell \geq 1$. Aus $h_1, \ldots, h_\ell \in \mathcal{H}$ und $y \in U$ bilden wir:

$$G_\ell(h_1, \ldots, h_\ell, y) = (h_1^{a_1} \circ \cdots \circ h_\ell^{a_\ell}(y))_{a_1 \ldots a_\ell \in \{0,1\}^\ell}. \tag{2}$$

Dabei ist $h^0 = \mathrm{id}_U$ und $h^1 = h$. Das läuft darauf hinaus, startend mit $y$ jede Teilfolge der Funktionen $h_\ell, \ldots, h_1$ *in dieser Reihenfolge* nacheinander anzuwenden. Dies liefert $2^\ell$ Elemente in $U$. Wenn wir $h_1, \ldots, h_\ell$ und $y$ zufällig wählen, ergibt sich eine Pseudozufallsfolge der Länge $2^\ell$ in $U$.

*Beispiel*: $G_2(h_1, h_2, y) = (y, h_2(y), h_1(y), h_1(h_2(y))) \in U^4$ (für die vier Folgen $a_1 a_2 \in \{00, 01, 10, 11\}$).

$$\begin{aligned} &G_3(h_1, h_2, h_3, y) \\ &\quad = \quad (y, h_3(y), h_2(y), h_2(h_3(y)), h_1(y), h_1(h_3(y)), h_1(h_2(y)), h_1(h_2(h_3(y)))) \in U^8, \end{aligned}$$

für die acht Folgen $a_1 a_2 a_3 \in \{000, 001, 010, 011, 100, 101, 110, 111\}$.

Da jede Hashfunktion durch $3\log(|U|)$ Zufallsbits beschrieben wird, benötigen wir im Wesentlichen $3\ell + 1$ zufällige Objekte aus $U$, um $2^\ell$ pseudozufällige Objekte zu konstruieren.

**Satz 2.3.2.** *Sei $\varepsilon = q^{-1/3}$, wobei $q$ so groß ist, dass $\varepsilon \leq \frac{1}{8}$ gilt. Dann gilt:*

$$\mathbf{Pr}(G_\ell(h_1, \ldots, h_\ell, y) \in \overline{W}^{2^\ell}) \leq (1 - \varrho)^{2^\ell} + (\ell + 2) \cdot \varepsilon. \tag{3}$$

Die Bedingung „$G_\ell(h_1, \ldots, h_\ell, y) \in \overline{W}^{2^\ell}$“ in (3) heißt, dass die vom Pseudozufallsgenerator erzeugte Folge der Länge $2^\ell$ die Menge $W$ kein einziges Mal trifft. Die Wahrscheinlichkeit hierfür ist im voll zufälligen Fall $(1 - \varrho)^{2^\ell}$.

Wir skizzieren ein typisches Szenario, bei dem sich die Mächtigkeit dieses Ansatzes zur Pseudozufallszahlen-Erzeugung zeigt. In vielen Fällen (zum Beispiel bei gewissen Eingaben beim Miller-Rabin-Primzahltest) ist die Ausgangsfehlerwahrscheinlichkeit $1 - \varrho$ nicht besser als eine Konstante (z. B. $1 - \varrho \approx \frac{1}{4}$). Nun sei $k = 2^\ell$ die Wiederholungszahl. Bei vollständig zufälliger $k$-facher Wiederholung erhält man eine Schranke $(1-\varrho)^k$. Dies ist nicht wesentlich besser als $1/4^k$. Bei Benutzung der Folge $G_\ell(h_1, \ldots, h_\ell, y)$ als Pseudozufallszahlen-Generator für eine Eingabe $x$ ist nach dem Satz die Fehlerschranke $(1 - \varrho)^k + (\ell + 2)/q^{1/3}$. Meist ist $q$, die Größe des Raums $W_x$ der potenziellen Zeugen

für $x$, von der Größe $2^{p(|x|)}$ für ein Polynom $p(n)$. Dann ist $\log q = p(|x|)$ und $\frac{1}{8}\log q$ ebenfalls polynomiell in $|x|$ groß. Wenn nun $k = 2^\ell < \frac{1}{8}\log q$ gilt, haben wir

$$(1-\varrho)^k \approx \frac{1}{4^k} = 4^{-(\log q)/8} = 2^{-(\log q)/4} = 1/q^{1/4} > (\ell+2)/q^{1/3},$$

das heißt, dass in der Wahrscheinlichkeitsschranke in (3) der Term $(1-\varrho)^{2^\ell}$ dominierend ist.

*Beweis* des Satzes: Wir wollen einen Induktionsbeweis führen. Dazu definieren wir für $0 \leq i \leq \ell$ und für beliebige $h_1, \ldots, h_i \in \mathcal{H}$ folgende Teilmengen von $U$ mit denjenigen $y \in U$, die sich bezüglich $(h_1, \ldots, h_i)$ schlecht verhalten:

$$A_i(h_1, \ldots, h_i) := \{y \in U \mid G_i(h_1, \ldots, h_i, y) \in \overline{W}^{2^i}\}.$$

Man beachte, dass nach der Definition gilt:

$$\mathbf{Pr}_{h \in U}(G_i(h_1, \ldots, h_i, y) \in \overline{W}^{2^i}) = \frac{|A_i(h_1, \ldots, h_i)|}{|U|} \tag{4}$$

**Definition 2.3.3.** *Eine Folge $(h_1, \ldots, h_i) \in \mathcal{H}^i$ heißt i-gut, falls für alle $j \leq i$ bei zufälliger Wahl von $y$ aus $U$*

$$\mathbf{Pr}_{y \in U}(G_j(h_1, \ldots, h_j, y) \in \overline{W}^{2^j}) \leq (1-\varrho)^{2^j} + \varepsilon_j, \tag{5}$$

*gilt, wobei $\varepsilon_j = \varepsilon \cdot \min\{j, 2\}$. Weiter definieren wir:*

$$\mathcal{N}_i := \{(h_1, \ldots, h_i) \textit{ ist nicht i-gut}\}.$$

Folgendes ist das zentrale technische Lemma, das durch Induktion über $i = 0, \ldots, \ell$ bewiesen wird.

**Lemma 2.3.4.** *Unter den Bedingungen von Satz 2.3.2 gilt:*

$$\mathbf{Pr}_{h_1, \ldots, h_i \in \mathcal{H}}(\mathcal{N}_i) \leq i\varepsilon.$$

*Beweis*: Induktion über $i$. Der Induktionsanfang ($i = 0$) ist einfach: Es muss nur $j = 0$ betrachtet werden, das heißt, dass gar keine Hashfunktion vorkommt. Es gilt stets:

$$\mathbf{Pr}(G_i(h_1, \ldots, h_0, y) \in \overline{W}^{2^0}) = \mathbf{Pr}(y \in \overline{W}) = 1 - \varrho.$$

Nun sei die Behauptung für $i-1$ erfüllt. Es gilt

$$\mathbf{Pr}(\mathcal{N}_i) \leq \mathbf{Pr}(\mathcal{N}_i \mid \overline{\mathcal{N}_{i-1}}) \cdot \mathbf{Pr}(\overline{\mathcal{N}_{i-1}}) + \mathbf{Pr}(\mathcal{N}_{i-1}).$$

Nach Induktionsvoraussetzung ist der letzte Term nicht größer als $(i-1)\varepsilon$. Weiter ist $\mathbf{Pr}(\overline{\mathcal{N}_{i-1}}) \leq 1$. Damit haben wir:

$$\mathbf{Pr}(\mathcal{N}_i) \leq \mathbf{Pr}(\mathcal{N}_i \mid \overline{\mathcal{N}_{i-1}}) + (i-1)\varepsilon. \tag{6}$$

Was bleibt zu tun? Wir müssen den ersten Summanden abschätzen. Dazu wählen wir $h_1, \ldots, h_{i-1}$ beliebig, aber *fest* mit der Eigenschaft, dass $(h_1, \ldots, h_{i-1})$ $(i-1)$-gut ist. Nun wird nur noch $h_i$ zufällig gewählt, und für die Frage, ob $(h_1, \ldots, h_i)$ $i$-gut ist oder nicht, kommt es nur noch auf (5) für $j = i$ an, also auf die Wahrscheinlichkeit

$$\mathbf{Pr}_{h_i \in \mathcal{H}}(\mathcal{N}_i) \leq \mathbf{Pr}_{h_i \in H}\left( \frac{|A_i(\overbrace{h_1, \ldots, h_{i-1}}^{\text{fest, } (i-1)\text{-gut!}}, h_i)|}{|U|} > (1-\varrho)^{2^i} + \varepsilon_i \right). \tag{7}$$

Wir betrachten nun $A_i(h_1, \ldots, h_i)$. Ein $y$ ist in dieser Menge genau dann wenn $G_i(h_1, \ldots, h_i, y) \in \overline{W}^{2^i}$, das heißt aber:

$$G_{i-1}(h_1, \ldots, h_{i-1}, y) \in \overline{W}^{2^{i-1}} \text{ und } G_{i-1}(h_1, \ldots, h_{i-1}, h_i(y)) \in \overline{W}^{2^{i-1}}.$$

Das bedeutet nichts anderes als

$$(y, h_i(y)) \in A_{i-1}(h_1, \ldots, h_{i-1}) \times A_{i-1}(h_1, \ldots, h_{i-1}). \tag{8}$$

Wir wenden das Hash-Mixing-Lemma 2.3.1 auf $A = B = A_{i-1}(h_1, \ldots, h_{i-1})$ an und erhalten, gilt, dass für mindestens einen Anteil von $1-\varepsilon$ aller $h_i \in \mathcal{H}$

$$\left| \frac{|\{y \mid (y, h_i(y)) \in A_{i-1}(\ldots) \times A_{i-1}(\ldots)\}|}{|U|} - \frac{|A_{i-1}(\ldots)| \cdot |A_{i-1}(\ldots|)}{|U|^2} \right| \leq \varepsilon \tag{9}$$

gilt.

**Behauptung:** Jedes $h_i$, das (9) erfüllt, führt dazu, dass (5) auch für $j = i$ erfüllt ist, dass also $(h_1, \ldots, h_i)$ $i$-gut ist.

*Beweis der Behauptung*: Betrachte ein $h_i$ mit (9). Weil $(h_1, \ldots, h_{i-1})$ $(i-1)$-gut ist, gilt

$$|A_{i-1}(\ldots)|/|U| \leq (1-\varrho)^{2^{i-1}} + \varepsilon_{i-1}.$$

Daraus folgt mit (9):

$$\begin{aligned} &\mathbf{Pr}_{y \in U}((y, h_i(y)) \in A_{i-1}(\ldots) \times A_{i-1}(\ldots)) \\ &\quad \leq \frac{|A_{i-1}(\ldots)| \cdot |A_{i-1}(\ldots)|}{|U|^2} + \varepsilon \leq ((1-\varrho)^{2^{i-1}} + \varepsilon_{i-1})^2 + \varepsilon. \end{aligned} \tag{10}$$

Wenn $i = 1$ ist, ist $i - 1 = 0$, also $\varepsilon_{i-1} = 0$, und wir können (10) durch $(1-\varrho)^2 + \varepsilon$ abschätzen. Wenn $i = 2$ ist, ist $i - 1 = 1$, also $\varepsilon_{i-1} \leq \varepsilon$, und wir erhalten für (10) die Schranke

$$((1-\varrho)^2 + \varepsilon)^2 + \varepsilon = (1-\varrho)^4 + 2(1-\varrho)^2\varepsilon + \varepsilon^2 + \varepsilon < (1-\varrho)^4 + 2\varepsilon,$$

wobei wir $\varrho \geq \frac{1}{2}$ und $\varepsilon \leq \frac{1}{2}$ benutzt haben. Wenn schließlich $i \geq 3$ ist, erhalten wir für (10) die Schranke

$$\begin{aligned}
&((1-\varrho)^{2^{i-1}} + 2\varepsilon)^2 + \varepsilon \\
&\quad\leq\ (1-\varrho)^{2^i} + 2(1-\varrho)^{2^{i-1}} \cdot 2\varepsilon + (2\varepsilon)^2 + \varepsilon \\
&\quad\leq\ (1-\varrho)^{2^i} + (2(1-\varrho)^4 \cdot 2 + 4\varepsilon + 1)\varepsilon \\
&\quad\leq\ (1-\varrho)^{2^i} + 2\varepsilon,
\end{aligned}$$

unter Verwendung der Ungleichungen $\varrho \geq \frac{1}{2}$ und $\varepsilon \leq \frac{1}{8}$. Damit ist $h_i$ $i$-gut, und die Behauptung ist bewiesen.

Aus der Behauptung folgt, dass mindestens ein Anteil von $1 - \varepsilon$ der $h_i$ Ungleichung (5) erfüllt. Also ist $\mathbf{Pr}(\mathcal{N}_i \mid \overline{\mathcal{N}_{i-1}}) \leq \varepsilon$, und mit (6) erhalten wir $\mathbf{Pr}(\mathcal{N}_i) \leq (i-1)\varepsilon + \varepsilon = i\varepsilon$. □

Nun können wir Satz 2.3.2 beweisen. Wir wählen $(h_1, \ldots, h_\ell)$ zufällig. Nach Lemma 2.3.4 ist die Wahrscheinlichkeit, dass diese Folge nicht $\ell$-gut ist, ist höchstens $\ell\varepsilon$. Nun gehen wir davon aus, dass $(h_1, \ldots, h_\ell)$ $\ell$-gut ist, und wählen $y$ zufällig. Nach der Definition von „$\ell$-gut“ erhalten wir dabei mit Wahrscheinlichkeit höchstens $(1-\varrho)^{2^i} + 2\varepsilon$ ein $y$, das $G_\ell(h_1, \ldots, h_\ell, y) \in \overline{W}^{2^\ell}$ erfüllt. Die gesamte Wahrscheinlichkeit, eine schlechte Pseudo-Zufallsfolge zu erzeugen, ist also höchstens

$$\ell\varepsilon + (1-\varrho)^{2^\ell} + 2\varepsilon = (1-\varrho)^{2^\ell} + (\ell+2)\varepsilon,$$

wie behauptet. □