Seminar Interaktive Beweissysteme

Arthur-Merlin-Games

Teil 1

Ralf Berger
Lars Siggelkow

14.05.2002
21.05.2002

1. Einleitung

Nachdem wir die Klasse IP näher untersucht haben, wenden wir uns einer speziellen restriktiveren Form dieser Protokolle zu - den sogenannten Arthur-Merlin-Spielen. Sie wurden erstmals 1979 von *Babai* vorgestellt.
Der Name ist der Arthus-Sage entlehnt und veranschaulicht das Grundprinzip interaktiver Beweise. Zum einen gibt es den allmächtigen Zauberer Merlin (Prover), zum anderen den probabilistischen, polynomial zeitbeschränkten und misstrauischen König Arthur (Verifier).
Zu einer gegebenen Sprache $L \subseteq \{0,1\}^*$ und einem Wort $x \in \{0,1\}^*$ versucht nun Merlin Arthur von $x \in L$ zu überzeugen . Wie schon bei IP geschieht dies rundenbasiert durch wechselseitigen Austausch von Nachrichten, wobei Arthur den Vorteil des Münzwurfs hat, d.h. ihm stehen in jeder Runde beliebig viele Zufallsbits für seine Entscheidungen zur Verfügung. Da Merlin sicherlich ein cleverer Zauberer ist, wird er versuchen, seinen König zu betrügen, ihm also auch für $x \notin L$ das Gegenteil beweisen wollen. Dies muss Arthur zu verhindern wissen. Dessen Vorteil in IP lag im wesentlichen darin, dass er seine Zufallsbits vor dem Prover geheim halten konnte. Diese Beschränkung wird bei den Arthur-Merlin-Spielen aufgegeben.
Arthur muss seine Zufallsbits sofort und unverändert auf das Kommunikationsband schreiben. Diese Art des Zufalls in interaktiven Beweisen wird auch mit „public coins“ bezeichnet.

Ein Arthur-Merlin-Spiel gestaltet sich also wie folgt:

- Arthur und Merlin machen abwechselnd einen Zug
- Zwei Züge ergeben eine Runde
- ein Zug für Arthur besteht aus:
  - dem Zugriff auf die Zufallsbits
  - ggf. lesen des Kommunikationsbandes
  - durchführen der Berechnung
  - akzeptieren / verwerfen oder
  - Zufallsbits und Anfrage auf das Kommunikationsband schreiben
- ein Zug für Merlin besteht dementsprechend aus:
  - lesen des Kommunikationsbandes
  - durchführen der Berechnung
  - Antwort auf das Kommunikationsband schreiben
- die Anzahl der Runden ist durch ein Polynom **t(|x|)** in der Länge der Eingabe x beschränkt, welche selbst nur polynomiell groß sein kann.

$AM(t(n))$ bezeichnet ein Protokoll über maximal t(n) Runden, bei dem Arthur anfängt
$MA(t(n))$ bezeichnet ein Protokoll über maximal t(n) Runden, bei dem Merlin anfängt
$AM(poly) = MA(poly) = \bigcup_{k>0} AM(n^k)$.

Abkürzend schreibt man AM für AM(1), MA für MA(1) usw.

Definition 1.
Eine Sprache L ist in $AM[t(n)]$, falls ein $AM(t(n))$-Protokoll existiert, mit:

$$\exists Merlin: \quad x \in L \Rightarrow \text{Prob}[Arthur \text{ akzeptiert } x] > \frac{2}{3}$$
$$\forall Merlin: \quad x \notin L \Rightarrow \text{Prob}[Arthur \text{ akzeptiert } x] < \frac{1}{3} \quad .$$

Definition 2.
Eine Sprache L ist in $AM[t(n)]_1$ (one-sided-$AM[t(n)]$), falls ein $AM(t(n))$-Protokoll existiert, mit:

$$\exists Merlin: \quad x \in L \Rightarrow \text{Prob}[Arthur \text{ akzeptiert } x] = 1$$
$$\forall Merlin: \quad x \notin L \Rightarrow \text{Prob}[Arthur \text{ akzeptiert } x] < \frac{1}{3} \quad .$$

Wie immer spielen die genauen Zahlenwerte für die Wahrscheinlichkeit auf Grund der möglichen Wahrscheinlichkeitsverstärkung keine Rolle.

Man kann zeigen, dass $AM[t(n)]_1 = AM[t(n)]$.

Viel interessanter ist jedoch die Frage, wie sich AM zu IP und dem Rest der Polynomialzeithierarchie verhält. Dass $AM \subseteq IP$ gelten muss, ist klar. Wie wirkt sich aber der Verlust der geheimen Zufallsbits auf die Mächtigkeit von AM aus? Der einzig verbliebene Vorteil von Arthur (der ja $x \in L$ richtig entscheiden können muss) ist jetzt, dass Merlin sein Verhalten nicht im Voraus simulieren kann. Überraschenderweise konnten *Goldwasser* und *Sipser* 1989 zeigen, dass dies kein echter Nachteil ist und daher gilt:
$AM(poly) = IP = PSPACE$.

2. Das Graph-Nichtisomorphie-Problem

Als Beispiel für ein Problem aus AM wählen wir das Graph-Nichtisomorphie-Problem. Wir haben den Beweis (das Protokoll) für $\overline{GI} \in IP$ bereits kennen gelernt. Der ganze Beweis beruhte auf der Tatsache, dass der Verifier seine Zufallsbits, mit denen er einen Graph und eine Permutation auf der Knotenmenge auswählt, vor dem Prover geheim halten konnte. Um so erstaunlicher ist es, dass man ein Protokoll angeben kann, bei dem der Verifier seine Zufallsbits getrost dem Prover übermitteln kann. Auf der Idee dieses Protokolls beruht auch der allgemeinere Beweis von *Goldwasser* und *Sipser*, dass private und öffentliche Zufallsbits äquivalent sind.

Theorem 1. $\overline{\mathbf{GI}} \in \mathbf{AM}$

Grundidee des AM-Protokolls / des Beweises ist folgende.
Man findet eine weitere Charakterisierung der Isomorphiebeziehung $G_1 \cong G_2$. Nun wählt Arthur per Zufall ein Verfahren aus, mittels dessen Merlin ein Indiz für die eben erwähnte Eigenschaft finden muss. Je nach dem, ob $G_1 \cong G_2$ oder nicht, wird ihm das statistisch selten oder oft gelingen.

Für den Beweis müssen wir einige Eigenschaften der Gruppentheorie bemühen.
Gegeben seien zwei Graphen G_1 und G_2, deren Knotenmengen V = {1,...,n} o.B.d.A identisch sind.

Def 3. Aut(H)
Für einen Graphen H = (V, E) mit V={1,...,n} bezeichne $Aut(H)$ die Menge seiner Automorphismen φ.

Weiter sei $\varphi \in S_n$ eine Permutation über n. $\varphi(H)$ bezeichne den Graphen $(V, \varphi(E))$ mit $\varphi(E) = \{\{\varphi(u), \varphi(v)\} \mid \{u,v\} \in E\}$.

Es gilt dann:

$$\varphi(G) = G \quad \Leftrightarrow \quad \varphi \in Aut(G) \tag{1}$$

$$\{G' \mid G' \cong G\} = \{\varphi(G) \mid \varphi \in S_n\} \tag{2}$$

$$\|\{G' \mid G' \cong G\}\| = \frac{n!}{\|Aut(G)\|} \tag{3}$$

(1) und (2) sind relativ offensichtlich.
Beweis zu (3):

Aus der Definition der Automorphismen folgt direkt, dass Aut(G) eine Untergruppe von S_n bezüglich der Komposition ∘ bildet.
Daraus folgt:

$$\{\varphi \in S_n \mid \varphi(G) = \varphi_1(G)\} = \varphi_1 \circ Aut(G)$$

$$\Rightarrow \quad \text{ex. } m = \frac{n!}{\|Aut(G)\|} \text{ Permutationen } \varphi_1, \varphi_2, .. \varphi_m \in S_n \text{ mit:}$$

$$S_n = \varphi_1 \circ Aut(G) \mathbin{\mathring{\cup}} \varphi_2 \circ Aut(G) \ldots \mathbin{\mathring{\cup}} \varphi_m \circ Aut(G)$$

$$\Rightarrow \quad \{G' \mid G' \cong G\} = \{\varphi_1(G), \ldots, \varphi_m(G)\},\ \varphi_i(G) \neq \varphi_j(G) \text{ für } i \neq j$$

$$\Rightarrow \quad \|\{G' \mid G' \cong G\}\| = m \qquad \square$$

Stellt man (3) um, erhält man die Beziehung, dass das Produkt aus der Anzahl isomorpher Graphen zu G und der Anzahl der Automorphismen von G konstant (nämlich n!) ist. Diese Eigenschaft werden wir im Beweis ausnutzen.
Betrachten wir also folgende Menge N:

$$\begin{aligned} N(G_1, G_2) &= \{(H, \varphi) \mid (H \cong G_1 \text{ oder } H \cong G_2),\ \varphi \in Aut(H)\} \\ &= \{(H, \varphi) \mid H \cong G_1,\ \varphi \in Aut(H)\} \cup \\ &\quad \{(H, \varphi) \mid H \cong G_2,\ \varphi \in Aut(H)\} \end{aligned} \tag{4}$$

Mit (3) können wir jetzt die Mächtigkeit der Menge N angeben. Wenn $G_1 \cong G_2$, sind die beiden Teilmengen aus (4) identisch, wenn nicht, sind sie disjunkt. Daher gilt:

$$\left\| N(G_1, G_2) \right\| = \begin{cases} n! & G_1 \cong G_2 \\ 2n! & G_1 \ncong G_2 \end{cases} \tag{5}$$

Wir haben also eine Eigenschaft auf den zu untersuchenden Graphen gefunden, die mit der Nichtisomorphie korreliert, die Arthur jedoch noch nicht direkt ausnutzen kann.

Der relative Unterschied (2:1) in der Größe von N wird für den Beweis nicht ausreichen. Indem wir aber ein endliches Kreuzprodukt betrachten, können wir dieses Verhältnis beliebig vergrößern. Es wird sich zeigen, dass das fünffache Kreuzprodukt von (5) ausreichend ist.

$$X(G_1, G_2) = N(G_1, G_2) \times N(G_1, G_2) \times N(G_1, G_2) \times N(G_1, G_2) \times N(G_1, G_2) \tag{6}$$

$$\left\| X(G_1, G_2) \right\| = \begin{cases} (n!)^5 & G_1 \cong G_2 \\ 2^5 (n!)^5 & G_1 \ncong G_2 \end{cases} \tag{7}$$

Wegen $\log n! \approx n \log n$ können wir auch jedes Element von X durch einen Binärstring polynomieller Länge p(n) kodieren . Den String $0^{p(n)}$ können wir o.B.d.A. als gültige Kodierung eines Elementes ausschließen.

Bevor wir fortfahren, müssen wir kurz den Begriff der Hashfunktion einführen:

Def 4. Hashfunktion

Als universelle Hashfunktionen bezeichnet man die Klasse H von Funktionen h, die eine Schlüsselmenge A in einen Adressraum B abbilden, wobei gelten muss:

$$\text{Prob}_{h \in_R H}\left[h(x_1) = h(x_2)\right] = \|B\|^{-1}.$$

Einfachstes Beispiel ist ***Lin(n,k)***, die Klasse aller linearen Funktionen von $\{0,1\}^n$ in $\{0,1\}^k$. Jedes $h \in \text{Lin(n,k)}$ lässt sich eindeutig durch die Binärmatrix $\mathbf{A}_n \in \{0,1\}^{k \times n}$ beschreiben, mit $h(\vec{y}) = \vec{y} \cdot \mathbf{A}_n$. Man sieht leicht, dass sich eine solche Hashfunktion effizient generieren lässt (durch Raten einer Matrix A erhält man unter Gleichverteilung ein h aus H), ebenso ist die Anwendung der Funktion auf einen Bitvektor $\vec{y}$ (Matrizenmultiplikation) effizient möglich.

Lemma 1.

Sei A eine Menge mit $\varnothing \neq A \subseteq \{0,1\}^n \setminus \{0\}$.

Sei h eine zufällig unter Gleichverteilung ausgewählte Hashfunktion $h \in_R Lin(n,k)$.

Sei S die Zufallsvariable $S = \left\| \left\{ y \in A \mid h(y) = 0^k \right\} \right\|$. Dann gilt:

$$\text{E}(S) = \|A\| \cdot 2^{-k} \tag{8}$$

$$Var(S) = \|A\| \cdot 2^{-k} \cdot \left(1 - 2^{-k}\right) < \text{E}(S) \tag{9}$$

Beweis siehe VL Komplexitätstheorie WS 2001/02.

Nun können wir unser Wissen über Hashfunktionen auf unsere Charakterisierung X von $\overline{GI}$ anwenden. Dazu definieren wir zwei Zufallsvariablen:

- $S = \left\| \left\{ y \in X \mid h(y) = 0^k \right\} \right\|$
- $Z: \exists x \in X: h(x) = 0^k$

  Das Ereignis Z wird jenes sein, welches Merlin versucht herbeizuführen, um Artur von $G_1 \ncong G_2$ zu überzeugen.

Wählen wir jetzt eine Hashfunktion h mit $k = \left\lceil \log\left(2^2 \cdot (n!)^5\right) \right\rceil$, erhalten wir mit (7) und (8) für den Erwartungswert der Zufallsvariable S:

$$G_1 \cong G_2 \Rightarrow \mathrm{E}(S) = \|X\| \cdot 2^{-k} = (n!)^5 \cdot 2^{-\left\lceil \log\left(2^2 \cdot (n!)^5\right) \right\rceil} < 2^{-2} = 1/4 \tag{10}$$

$$G_1 \ncong G_2 \Rightarrow \mathrm{E}(S) = \|X\| \cdot 2^{-k} = 2^5 \cdot (n!)^5 \cdot 2^{-\left\lceil \log\left(2^2 \cdot (n!)^5\right) \right\rceil} > 2^{-2} = 4 \tag{11}$$

Um nun auch die Wahrscheinlichkeit für das Ereignis Z zu berechnen, benötigen wir noch eine weitere Beziehung:

Tschebyschev-Ungleichung
Sei X eine stetige Zufallsvariable mit dem Erwartungswert μ und der Streuung σ^2, dann gelten folgende Ungleichungen:

$$\mathrm{Prob}\left[|X| \geq a\right] \leq E\left[X^2\right]/a^2 \tag{12}$$

$$\mathrm{Prob}\left[|X - \mu| \geq a\right] \leq \mathrm{Var}[X]/a^2 \tag{13}$$

$$\mathrm{Prob}\left[|X - \mu| \geq k\sigma\right] \leq 1/k^2 \tag{14}$$

Betrachte:

- $G_1 \cong G_2$:

$$\begin{aligned}
\mathrm{Prob}[Z] &= \mathrm{Prob}[S \geq 1] && \text{laut Def.} \\
&= \mathrm{Prob}\left[\sqrt{S} \geq 1\right] \\
&\leq E\left[\left(\sqrt{S}\right)^2\right] && \text{mit (12)} \\
&= E[S] \\
\mathrm{Prob}[Z] &\leq 1/4 && \text{mit (10)}
\end{aligned}$$

- $G_1 \ncong G_2$:

$$\begin{aligned}
\mathrm{Prob}[Z] &= \mathrm{Prob}[S \geq 1] && \text{laut Def.} \\
&= 1 - \mathrm{Prob}[S = 0] \\
&\geq 1 - \mathrm{Prob}\left[|S - E[E]| \geq E[S]\right] \\
&\geq 1 - \left(\mathrm{Var}[S] \,/\, \left(E[S]\right)^2\right) && \text{mit (13)} \\
&\geq 1 - \left(E[S] \,/\, \left(E[S]\right)^2\right) && \text{mit (9)} \\
&\geq 1 - E[S] \\
\mathrm{Prob}[Z] &\geq 3/4 && \text{mit (11)}
\end{aligned}$$

Wir haben jetzt eine Eigenschaft gefunden, die mit dem wie üblich geforderten Wahrscheinlichkeitsunterschied direkt mit der Graph-Nichtisomorphie einhergeht. Außerdem kann sie leicht vom Verifier überprüft werden. Daraus ergibt sich folgendes AM-Protokoll:

- Eingabe G_1 und G_2 auf einer n-elementigen Knotenmenge
- **Arthur** wählt zufällig eine Hashfunktion h aus $\mathrm{Lin}\left(n, k = \left\lceil \log\left(2^2 \cdot (n!)^5\right)\right\rceil\right)$ und sendet diese (polynomiell in n kodiert) an Merlin
- **Merlin** rät sich (sofern existent) ein $x \in X$ mit $h(x) = 0^k$ und sendet x und den Beweis für $x \in X$ (den entsprechenden Isomorphismus) an Arthur.
- **Arthur** überprüft (in Polynomialzeit) $h(x) = 0^k$ und akzeptiert wenn alles korrekt ist; kann Merlin ihm kein x präsentieren, so verwirft er.

Ist $G_1 \ncong G_2$, dann ist die Menge X sehr groß und es ist sehr wahrscheinlich (Prob > 3/4), dass ein x existiert, mit $h(x) = 0^k$. Ist $G_1 \cong G_2$, ist dies eher unwahrscheinlich (Prob < 1/4).

$\Rightarrow \overline{\mathbf{GI}} \in \mathbf{AM}$

□

Wir hatten zu Beginn erwähnt, dass $AM[t(n)]_1 = AM[t(n)]$.
Statt des allgemeinen Beweises, soll die Äquivalenz am nun schon gut bekannten Beispiel der Graph-Nichtisomorphie gezeigt werden.

Theorem 2. $\overline{\mathbf{GI}} \in$ one - sided - AM

Das Verfahren für Theorem 1 hilft uns hier nicht weiter, da egal wie groß man die Menge N bzw. X auch aufbläht, das Eintreten des Ereignisses Z wird für den positiven Fall $G_1 \ncong G_2$ zwar immer wahrscheinlicher, aber niemals sicher.
Wir brauchen also eine andere Eigenschaft, die auf einer entsprechend großen Menge mit Sicherheit erfüllt ist. Dafür bieten sich Kollisionen an.
Wieder werden wir die Klasse *LIN[m,k]* o.B. als eine universelle Hashklasse verwenden.

Zunächst definieren wir den Begriff der Kollision in verschiedenen Kontexten:

Def 5. Kollision
Sei h eine zufällig unter Gleichverteilung aus einer universellen Hash-Klasse gewählte Hash-Funktion. Sei Y eine Menge von Schlüsseln und x ein weiterer Schlüssel. Wir bezeichnen das Ereignis, das dieser Schlüssel durch h auf dieselbe Adresse abgebildet wird, wie ein Element aus Y als **Kollision(x, Y, h)**.

$$Kollision(x, Y, h) = \exists y \in Y : x \neq y \wedge h(x) = h(y)$$

$$\begin{aligned} \mathrm{Prob}\left[Kollision(x, Y, h)\right] &\leq \sum_{y \in Y, x \neq y} \left[h(x) = h(y)\right] \\ &\leq \sum_{y \in Y, x \neq y} \left[\|B\|^{-1}\right] \\ &\leq \|Y\| / \|B\| = \beta \end{aligned}$$

β wird als Belegungsfaktor bezeichnet

Betrachten wir nicht einen einzigen Schlüssel x, sondern eine ganze Menge X, bezeichen wir das Ereignis, dass ein x aus X mit Y kollidiert als **Kollision(X, Y, h)**.

$$\begin{aligned} Kollision(X,Y,h) &= \exists x \in X : Kollision(x,Y,h) \\ \text{Prob}[Kollision(X,Y,h)] &\le \sum_{x \in X} \text{Prob}[Kollision(x,Y,h)] \\ &\le \|X\| \cdot {}^{\|Y\|}\!/_{\|B\|} \end{aligned}$$

Damit kann nun einfach der Fall beschrieben werden, dass Kollisionen innerhalb einer Menge X von Schlüssel auftreten - **Kollision(X, h)**.

$$\begin{aligned} Kollision(X,h) &= \exists x \in X : Kollision(x,X-\{x\},h) \\ \text{Prob}[Kollision(X,h)] &\le \sum_{x \in X} \text{Prob}[Kollision(x,X-\{x\},h)] \\ &\le \|X\| \cdot {}^{\|X\|-1}\!/_{\|B\|} \end{aligned}$$

Def 6. Kollektion von Hashfunktionen

Die geordnete Menge H =(h_1, ..., h_l) heißt Kollektion von *l* Hashfunktionen von A nach B. Die Funktionen h_i müssen natürlich eben solche Hashfunktionen sein. Die Kollektion H definiert nun eine Abbildung von Schlüsseln in den Adressraum $\{1,...,l\} \times B$ wie folgt:

Sei $x_1,...,x_k$ eine Folge von Schlüsseln. Jedem x_i weise nun die kleinste Adresse der Form (j, $h_j(x_i)$) mit $j \in \{1,...,l\}$ zu, die noch nicht belegt ist. Ist keine solche Adresse mehr frei, so bezeichnen wir das Ereignis als Kollision des Schlüssels x_i mit all seinen Vorgängern x_1, ..., x_{i-1}.

Wir können nun die oben untersuchten Fälle für Kollisionen und deren Wahrscheinlichkeiten auch auf Kollektionen von Hashfunktionen erweitern:

$$\begin{aligned} Kollision(x,Y,H) &= \exists y_1,...,y_l \in Y : x \neq \{y_1,...,y_l\} \wedge \forall j \in \{1,...,l\} : h_j(x) = h_j(y_1) \\ \text{Prob}[Kollision(x,Y,H)] &= \text{Prob}[\forall j \in \{1,...,l\} : Kollision(x,Y,h_j)] \\ &= \prod_{j=1}^{l} \text{Prob}[Kollision(x,Y,h_j)] \\ &\le \beta^l \end{aligned}$$

$$\begin{aligned} Kollision(X,Y,H) &= \exists x \in X : Kollision(x,Y,H) \\ \text{Prob}[Kollision(X,Y,H)] &\le \sum_{x \in X} \text{Prob}[Kollision(x,Y,H)] \\ &\le \|X\| \cdot \beta \end{aligned}$$

$$
\begin{aligned}
Kollision(X,H) &= \exists x \in X : Kollision(x, X-\{x\}, H) \\
\text{Prob}[Kollision(X,H)] &\le \sum_{x\in X} \text{Prob}[Kollision(x, X-\{x\}, H)] \\
&\le \|X\| \cdot \left(\|X\| - 1/\|B\|\right)^{l} \\
&< \|X\| \cdot \beta^{l} \qquad (15)
\end{aligned}
$$

Lemma 2 (Coding Lemma)
Mit (15) können wir nun für bestimmte Kollektionen und Schlüsselmengen die Kollisionswahrscheinlichkeiten berechnen. Bezeichne ***Lin[l;m;k]*** die Menge aller Kollektionen H =(h_1, ..., h_l) mit $h_i \in$Lin[m;k] i = 1,...,*l*. Sei wie gewohnt A die Schlüsselmenge, $A \subseteq \Sigma^m - \{0^m\}$. Dabei enthalte A höchstens 2^{k-1} Schlüssel. Weiter sei $H \in$ *Lin[k;m;k]*. Dann gilt:

$$
\begin{aligned}
\text{Prob}[Kollision(A,H)] &< \|A\| \cdot {\beta_H}^{l} \\
&< 2^{k-1} \cdot \left(\frac{2^{k-1}}{2^k}\right)^k \\
&< 2^{-1}
\end{aligned}
$$

Das heißt, dass höchstens die Hälfte alle Kollektionen $H \in$ *Lin[k;m;k]* eine Kollision auf A verursachen können.

Lemma 3
Eine Kollektion von *l* Hashfunktionen, kann natürlich nur so viele Schlüssel der Schlüsselmenge A kollisionsfrei in den erweiterten Adressraum ($\{l\}\times$B) abbilden, wie dieser Elemente hat. Das heißt:
Jede Kollektion H von *l* Hashfunktionen mit Adressraum B verursacht mit Sicherheit eine Kollision auf A, wenn $\|A\| > l \cdot \|B\|$.

Lemma 4
Im folgenden seien:

$a \ge 1$, $d \ge 0$, und ε eine Fehlerschranke mit $0 < \varepsilon \le 1$	fest vorgegeben
$r \ge 1 + 4d/\varepsilon + 8m/\varepsilon^2$ und $k = 1 + \lceil r \log a \rceil$	beliebig

Dann gilt:

a) Jede Menge $X \subseteq \Sigma^m$, mit: X^r kann von einer Kollektion $H \in$ Lin[d+k; m·r; k] kollisionsfrei abgebildet werden, enthält weniger als $(1+\varepsilon)\cdot a$ Elemente.

Beweis:
O.B.d.A. ist $a < 2^m$, da anderenfalls die Menge X trivialerweise weniger als $(1+\varepsilon)\cdot a$ Elemente enthält.

$$
\begin{aligned}
\text{NR:} \quad (1+\varepsilon)^r &\ge 1 + r\varepsilon + r(r-1)\varepsilon^2/2 \\
&= 1 + (\varepsilon + 4d + 8m/\varepsilon) + r(2d\varepsilon + 4m) \\
&> 1 + 4d + 8 + 4rm \\
&> 4(2 + d + r\log a) \qquad \text{wegen } a < 2^m \qquad (*)
\end{aligned}
$$

$$\begin{aligned}
\left\|X^r\right\| &\le (d+k)\cdot 2^k && \text{wegen Lemma 3}\\
&= (d+1+\lceil r\log a\rceil)\cdot 2^{1+\lceil r\log a\rceil}\\
&< (1+d+(r\log a+1))\cdot 2\cdot 2^{(r\log a+1)}\\
&= 4(2+d+r\log a)\cdot a^r\\
&< (1+\varepsilon)^r\cdot a^r && \text{mit (*)}
\end{aligned}$$

$$\Rightarrow \ \|X\| < (1+\varepsilon)\cdot a \qquad \square \qquad (16)$$

b) Für jede Menge $X \subseteq \Sigma^m - \{0^m\}$, mit $\|X\| \le a$ ist die Wahrscheinlichkeit einer Kollision auf der Menge X^r durch eine zufällig aus Lin[d+k; m·r; k] gewählte Kollektion H kleiner als 2^{-d-1}.

Beweis:

$$\begin{aligned}
\text{Prob}\left[Kollision(X^r,H)\right] &< \left\|X^r\right\|\cdot\beta^l && \text{mit (15)}\\
&= \left\|X^r\right\|\cdot\left(\frac{\left\|X^r\right\|}{2^k}\right)^{d+k}\\
&\le a^r\cdot\left(\frac{a^r}{2^{1+\lceil r\log a\rceil}}\right)^{d+1+\lceil r\log a\rceil}\\
&< a^r\cdot\left(\frac{a^r}{2\cdot 2^{r\log a}}\right)^{d+1+\lceil r\log a\rceil}\\
&= a^r\cdot a^{r(d+1+\lceil r\log a\rceil)}\cdot 2^{-(d+1+\lceil r\log a\rceil)}\cdot 2^{-r\log a(d+1+\lceil r\log a\rceil)}\\
&= a^r\cdot a^{r(d+1+\lceil r\log a\rceil)}\cdot 2^{-(d+1)}\cdot a^{-r}\cdot a^{-r(d+1+\lceil r\log a\rceil)}\\
&= 2^{-d-1}
\end{aligned}$$

$$\Rightarrow \ \text{Prob}\left[Kollision\left(X^r,H\right)\right] < 2^{-d-1} \qquad \square \qquad (17)$$

Man beachte, dass die Wahl von r in dieser Abschätzung unerheblich ist, solange $r \ge 1$.

Die Idee für das Artur-Merlin-Protokoll ist nun folgende: Die bekannte Menge $N(G_1,G_2)$ wird durch r-fache Kreuzproduktbildung hinreichend vergrößert. Merlin muss diesmal nicht ein, sondern eine bestimmte Anzahl von Elementen aus dieser Menge finden. Für geeignet gewählte Größen folgt aus Lemma 4 die sichere Kollision im einen und eine geringe Kollisionswahrscheinlichkeit im anderen Fall.

Wähle also:

$d = \varepsilon = 1$

$a = n!$ (n ist die Anzahl der Knoten in G_1 und G_2)

$m = p(n)$ (das die Länge der Elemente (H,φ) aus $N(G_1,G_2)$ angebende Polynom)

$r = 8m/\varepsilon^2 + 4d/\varepsilon + 1 = 8m + 5$

$k = 1+\lceil r\log a\rceil$

$\Rightarrow$ #Hashfunktionen = k+1

Wenden wir auf diesen Daten nun Lemma 4 an:
a) besagt, dass eine Menge X mit einer k+1-elementigen Hashkollektion nur kollisionsfrei abgebildet werden kann, wenn $\|X\| < (1+1)\cdot n!$ ist.
Aus b) folgt, dass für $\|X\| \leq n!$ die Kollisionswahrscheinlichkeit höchstens 2^{-1-1} beträgt.

Daraus ergibt sich folgendes one-sided-AM-Protokoll:

- Eingabe G_1 und G_2 auf einer n-elementigen Knotenmenge
- **Arthur** wählt zufällig eine Kollektion H aus *Lin[k+1; mr; k]* (m und k wie oben beschrieben) und sendet diese (polynomiell in n kodiert) an Merlin
- **Merlin** rät sich (sofern existent) k+2 verschiedene Elemente x, x_1, ..., x_{k+1} aus $X^r = N(G_1,G_2)^r$ und sendet diese und die Zeugen (Beweise für $x_i \in X^r$, d.h. die entsprechenden Isomorphismen) an Arthur.
- **Arthur** überprüft ob $x_i \in X^r$ und akzeptiert, falls $\{x, x_1, ..., x_{k+1}\}$ eine Kollision in H verursacht, d.h. $h_i(x) = h_i(x_i)$ für i = 1,...k+1. Anderenfalls verwirft er.

Ist $G_1 \ncong G_2$, dann ist $\|X\| = 2n!$ und es ist nicht möglich, k+2 Elemente aus X^r immer kollisionsfrei durch H auf B abzubilden. Merlin wird also solche kollidierenden Elemente mit Sicherheit finden.
Für $G_1 \cong G_2$, ist ist $\|X\| = n!$ und eine Kollision eher unwahrscheinlich (Prob < 1/4).
$\Rightarrow \overline{\mathbf{GI}} \in \mathbf{AM_1}$

□