Übungen zur Vorlesung Grundbegriffe der Theoretischen Informatik

technische universität dortmund

Thomas Schwentick

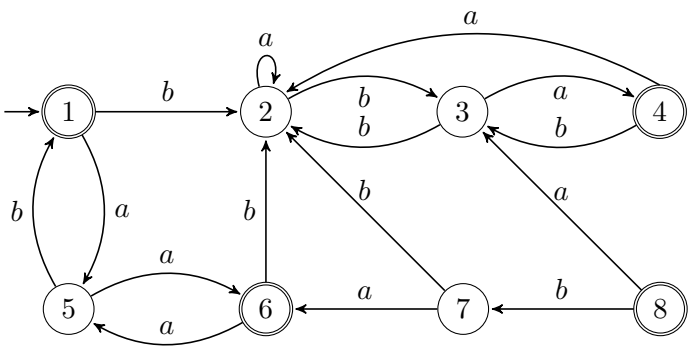
Gaetano Geck, Lutz Oettershagen,
Christopher Spinrath, Marco Wilhelm

Lehrstuhl 1
Logik in der Informatik

SoSe 2018 | **Übungsblatt 3** | **26.04.2018**

Aufgabe 3.1 [Minimierung endlicher Automaten] **4 Punkte**

Bestimmen Sie zu nachfolgend angegebenem DFA einen äquivalenten minimalen DFA unter Verwendung des in der Vorlesung vorgestellten Algorithmus. Geben Sie dabei für jedes markierte Zustandspaar in einer Tabelle, wie in der Vorlesung, an, in welchem Durchlauf es markiert wurde. In einem Durchlauf sollen dabei genau die Zustandspaare markiert werden, die aufgrund von Markierungen im vorherigen Durchlauf (zum ersten Mal) markiert werden.
Zeichnen Sie anschließend den resultierenden Minimalautomaten.



Lösung:

Bezeichne $\mathcal{A}$ den in der Aufgabenstellung gegebenen DFA.
1. Schritt: Zunächst werden diejenigen Zustände entfernt, die vom Startzustand des Automaten $\mathcal{A}$ aus nicht zu erreichen sind. Es handelt sich dabei um die Zustände 7 und 8.

2. Schritt: Anschließend wird mithilfe des Markierungsalgorithmus die Menge $N(\mathcal{A})$ der Zustandspaare berechnet, für die es ein Wort gibt, sodass sich das Akzeptanzverhalten des Automaten ausgehend von den beiden Zuständen bei Lesen dieses Wortes unterscheidet. Dies geschieht iterativ in mehreren Runden und resultiert in den folgenden Markierungen.

Initialisierung:

	1	2	3	4	5	6
1	—	$\times^0$	$\times^0$		$\times^0$	
2	—	—		$\times^0$		$\times^0$
3	—	—	—	$\times^0$		$\times^0$
4	—	—	—	—	$\times^0$	
5	—	—	—	—	—	$\times^0$

1. Iteration:

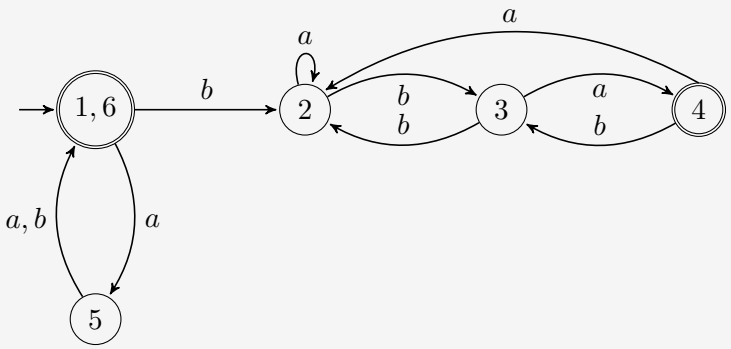
	1	2	3	4	5	6
1	—	$\times^0$	$\times^0$		$\times^0$	
2	—	—	$\times^1$	$\times^0$	$\times^1$	$\times^0$
3	—	—	—	$\times^0$	$\times^1$	$\times^0$
4	—	—	—	—	$\times^0$	
5	—	—	—	—	—	$\times^0$

2. Iteration:

	1	2	3	4	5	6
1	—	$\times^0$	$\times^0$	$\times^2$	$\times^0$	
2	—	—	$\times^1$	$\times^0$	$\times^1$	$\times^0$
3	—	—	—	$\times^0$	$\times^1$	$\times^0$
4	—	—	—	—	$\times^0$	$\times^2$
5	—	—	—	—	—	$\times^0$

In der Initialisierungsphase des Markierungsalgorithmus wird die vorher leere Menge M um die Zustandspaare (p, q) ergänzt, für die einer der beteiligten Zustände akzeptierend ist, der andere jedoch nicht: $M = \{(p, q), (q, p) \mid p \in F \Leftrightarrow q \notin F\}$. In jeder darauffolgenden Iteration werden all jene Zustandspaare (p, q) markiert, für die es ein Symbol σ gibt, sodass $\big(\delta(p, \sigma), \delta(q, \sigma)\big)$ markiert ist. Dies ist beispielsweise in der 1. Iteration für das Zustandspaar $(2, 3)$ der Fall, da $\big(\delta(2, a), \delta(3, a)\big) = (2, 4)$ in der vorhergehenden Iteration bereits markiert wurde.
Der Iterationsprozess endet, wenn in einer Iteration keine Markierungen mehr vorgenommen werden. Dies ist für den betrachteten Automaten in der 3. Iteration der Fall (die Markierungsmatrix sieht dort genauso aus wie in der 2. Iteration).
Nach Lemma 4.5 sind nach Terminierung des Markierungsalgorithmus alle nicht markierten Paare zueinander äquivalent, hier also lediglich das Paar $(1, 6)$, sodass die Zustände 1 und 6 äquivalent sind.

3. Schritt: Verschmelzen der Zustände 1 und 6 ergibt den folgenden Automaten:



Aufgabe 3.2 [Nerode-Äquivalenzklassen] 4 Punkte

Sei $L = \{w \in \{a, b\}^* \mid |w| > 1$ und der vorletzte Buchstabe in w ist ein $b\}$.

a) Geben Sie für jede Äquivalenzklasse der Nerode-Relation $\sim_L$ einen Repräsentanten an. Geben Sie außerdem für je zwei verschiedene dieser Repräsentanten x_i und x_j ein Wort z_{ij} an, das bezeugt, dass x_i und x_j verschiedene Äquivalenzklassen repräsentieren. Es soll also gelten $x_i z_{ij} \in L \not\Leftrightarrow x_j z_{ij} \in L$ für alle Repräsentanten x_i, x_j mit $x_i \neq x_j$. **(2,5 Punkte)**

b) Geben Sie einen minimalen DFA $\mathcal{A}$ an, so dass $L(\mathcal{A}) = L$ gilt. Begründen Sie sowohl, dass $\mathcal{A}$ die Sprache L entscheidet, als auch, dass $\mathcal{A}$ minimal ist. **(1,5 Punkte)**

Hinweis

Die Nerode-Relation $\sim_L$ hat genau vier Äquivalenzklassen. Sie können diesen Fakt in dieser Aufgabe ohne Begründung benutzen.

Lösung:

a) Die Wörter $x_1 = aa,\ x_2 = ab,\ x_3 = ba,\ x_4 = bb$ repräsentieren paarweise verschiedene Äquivalenzklassen der Nerode-Relation $\sim_L$. In der folgenden Tabelle ist zu jedem Paar x_i, x_j, $1 \leq i < j \leq 4$ ein Wort angegeben, dass diese Behauptung bezeugt:

	aa	ab	ba	bb
aa	—	b	ε	ε
ab	—	—	b	ε
ba	—	—	—	a

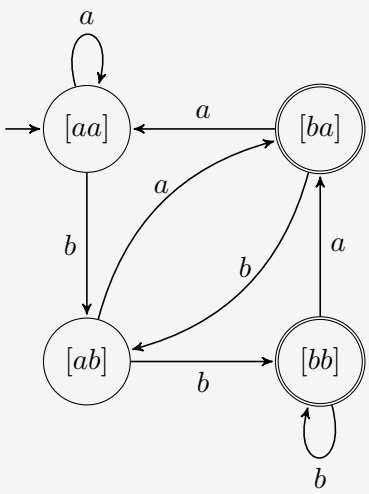
Beispielsweise repräsentieren ab und ba verschiedene Äquivalenzklassen, weil für $z_{23} = b$ gilt, dass $abz_{23} = abb \in L$ ist und $baz_{23} = bab \notin L$ gilt. Es gilt also insbesondere $abb \in L \not\Leftrightarrow bab \in L$.

Alternativ kann man die Repräsentanten systematisch bestimmen, indem man, angefangen mit ε, die Wörter über Σ lexikographisch aufzählt und jeweils prüft, ob das Wort äquivalent zu einem bereits gefundenen Repräsentanten ist. Zum Beispiel gilt $a \sim_L \varepsilon$ und $b \not\sim_L \varepsilon$. Das Wort a wird daher nicht als Repräsentant ausgewählt, b dagegen ist ein neuer Repräsentant. Insgesamt erhalten wir die Repräsentanten $x'_1 = \varepsilon,\ x'_2 = b,\ x'_3 = ba,\ x'_4 = bb$, die paarweise verschiedene Äquivalenzklassen der Nerode-Relation $\sim_L$ repräsentieren.

Im Vergleich zur ersten Lösung gilt $aa \sim_L \varepsilon$ und $ab \sim_L b$. Die Tabelle, die die Wörter z'_{ij} angibt, ist daher, bis auf Umbennung der Repräsentanten, identisch mit der vorherigen Tabelle:

	ε	b	ba	bb
ε	—	b	ε	ε
b	—	—	b	ε
ba	—	—	—	a

b) Nach Vorlesung ist der Äquivalenzklassenautomat $\mathcal{A}_L$ ein minimaler DFA. Es gilt insbesondere $L(\mathcal{A}_L) = L$. Mit Aufgabenteil a) und dem Hinweis können wir folgern, dass $\mathcal{A}_L$ genau jene vier Zustände hat, die den Äquivalenzklassen der Wörter aa, ab, ba und bb entsprechen. Der Äquivalenzklassenautomat kann wie in dem Beweis von Satz 4.1 konstruiert werden und ist im Folgenden angegeben. Hierbei ist zu beachten, dass $[aa] = [\varepsilon]$ sowie $[ab] = [b]$ bzw. $aa \sim_L \varepsilon$ und $ab \sim_L b$ gilt (vergleiche mit der Alternativlösung zu a)).

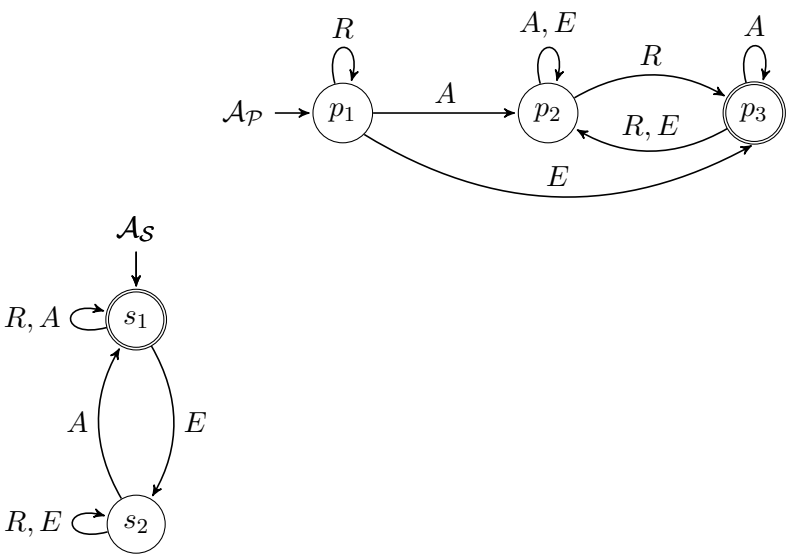


Aufgabe 3.3 [Verifikation mit endlichen Automaten] **7 Punkte**

In dieser Aufgabe wollen wir mit Hilfe der Automatentheorie verifizieren, ob ein gegebenes Programm $\mathcal{P}$ eine Spezifikation $\mathcal{S}$ erfüllt. Dazu nehmen wir stark vereinfachend an, dass ein Programm zu jedem Zeitpunkt genau eine der drei folgenden Aktionen ausführen kann:

- Es kann einen Rechenschritt durchführen;
- Es kann eine Ausgabe machen; oder
- Es kann eine Eingabe gelesen werden.

Ein *Programmablauf* ist eine Folge von Aktionen. Wir nehmen vereinfachend an, dass jedes Programm terminiert; jeder Programmablauf ist also eine *endliche* Folge. Ein Programm erfüllt eine Spezifikation genau dann, wenn jeder Programmablauf des Programms die Bedingungen der Spezifikation erfüllt.
Da Programmabläufe endliche Folgen sind, können wir sie durch Wörter über dem Alphabet $\{R, A, E\}$ modellieren, wobei R für einen Rechenschritt, A für eine Ausgabe und E für eine Eingabe stehen soll. Die Programmabläufe eines Programms können dann durch einen DFA modelliert werden: Der Automat akzeptiert genau die Wörter über $\{R, A, E\}$, die Programmabläufe des Programms darstellen. Ebenso lässt sich eine Spezifikation durch einen DFA modellieren: Es werden genau die Wörter akzeptiert, die Programmabläufe darstellen, die die Spezifikation erfüllen.
Der im Folgenden angegebene Automat $\mathcal{A}_\mathcal{P}$ entscheidet die Sprache aller Wörter, die Programmabläufe eines Beispielprogramms $\mathcal{P}$ darstellen. Weiterhin wird die Spezifikation $\mathcal{S}$, die fordert, dass nach jeder gelesenen Eingabe irgendwann eine Ausgabe erfolgen muss, gerade durch den Automaten $\mathcal{A}_\mathcal{S}$ modelliert.



a) Konstruieren Sie einen Produktautomaten $\mathcal{A}_\mathcal{S} \times \mathcal{A}_\mathcal{P}$. Wählen Sie dabei die Menge der akzeptierenden Zustände so, dass der Produktautomat für die Frage, ob das Programm $\mathcal{P}$ die Spezifikation $\mathcal{S}$ erfüllt, hilfreich ist. **(3 Punkte)**

b) Begründen sie mit Hilfe des in Aufgabenteil a) konstruierten Produktautomaten, ob das Programm $\mathcal{P}$ die Spezifikation $\mathcal{S}$ erfüllt. **(1 Punkt)**

c) Geben Sie einen Algorithmus in Pseudo-Code an, der als Eingabe zwei DFAs $\mathcal{A}_{\mathcal{P}'}$ und $\mathcal{A}_{\mathcal{S}'}$ über dem Alphabet $\{R, A, E\}$ erhält und entscheidet, ob das Programm $\mathcal{P}'$ die Spezifikation $\mathcal{S}'$ erfüllt. Bestimmen Sie außerdem die asymptotische Laufzeitschranke (in $\mathcal{O}$-Notation) Ihres Algorithmus.

Sie können die Algorithmen und Abschlusseigenschaften der Vorlesung als Bausteine in ihrem Algorithmus verwenden.

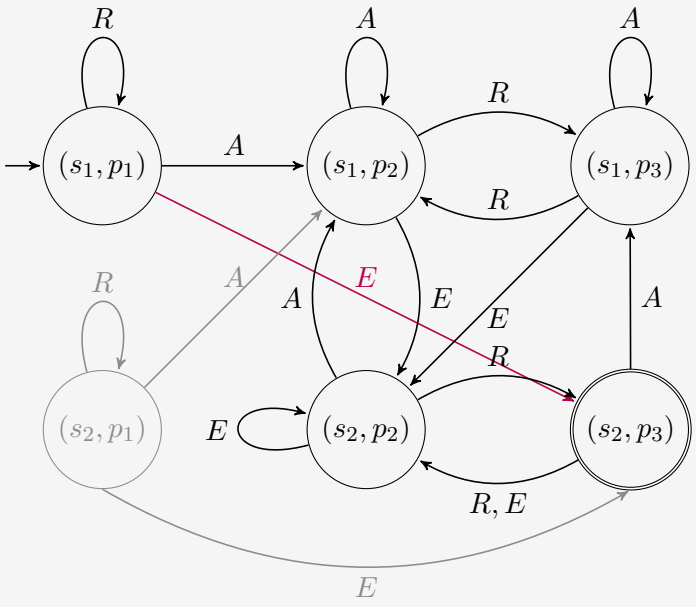
> **Hinweis**
>
> Zur Angabe der Algorithmen können Sie sich an den Algorithmen in Abschnitt 5.3 der Vorlesung orientieren.

(2 Punkte)

d) Wie kann Ihr Algorithmus abgewandelt werden, wenn die Spezifikation durch einen regulären Ausdruck statt einem DFA angegeben wird? Hat dies Auswirkungen auf die Laufzeit? Wenn ja, geben Sie eine neue Laufzeitschranke an. **(1 Punkt)**

Lösung:

a) Der Produktautomat $\mathcal{A}_\mathcal{S} \times \mathcal{A}_\mathcal{P}$ ist im folgenden angegeben.



Der Zustand (s_2, p_1) muss nicht konstruiert werden, weil er nicht vom Startzustand erreichbar ist. Er ist aber hier der Vollständigkeit halber mit angegeben.

Als Menge der akzeptierenden Zustände wurde $(Q_\mathcal{S} \setminus F_\mathcal{S}) \times F_\mathcal{P}$ gewählt, wobei $Q_\mathcal{S}$ die Menge der Zustände von $\mathcal{A}_\mathcal{S}$ ist und $F_\mathcal{S}$, $F_\mathcal{P}$ die Mengen der akzeptierenden Zustände von $\mathcal{A}_\mathcal{S}$ bzw. $\mathcal{A}_\mathcal{P}$ sind. Die akzeptierenden Zustände wurden also so gewählt, dass der Produktautomat jene Wörter akzeptiert, die gegen die Spezifikation verstoßen, aber Programmabläufe von $\mathcal{P}$ darstellen.

Alternativ können jene Zustände (s_i, p_j) als akzeptierend gewählt werden, sodass s_i akzeptierend ist *oder* p_j nicht akzeptierend ist. In diesem Fall akzeptiert der Produktautomat alle Wörter, die, wenn sie Programmabläufe von $\mathcal{P}$ darstellen, auch die Spezifikation erfüllen. In den weiteren Aufgabenteilen muss dann geprüft werden, ob es ein Wort gibt, dass der Produktautomat *nicht* akzeptiert bzw. ob der Produktautomat *alle* Wörter akzeptiert.

b) **Behauptung:** Das Programm $\mathcal{P}$ erfüllt die Spezifikation *nicht*. Das Wort $w = E$ wird von dem Produktautomaten $\mathcal{A}_\mathcal{S} \times \mathcal{A}_\mathcal{P}$ akzeptiert, wie man an der markierten Transition im Produktautomaten ablesen kann. Mit unserer Wahl der akzeptierenden Zustände (und der Semantik des Produktautomaten) könnnen wir schließen, dass $\mathcal{A}_\mathcal{P}$ das Wort w akzeptiert und $\mathcal{A}_\mathcal{S}$ das Wort w verwirft. Also stellt das Wort w einen Programmablauf von $\mathcal{P}$ dar, der die Spezifikation *nicht* erfüllt. Also erfüllt $\mathcal{P}$ die Spezifikation $\mathcal{S}$ nicht.

c) Wir verallgemeinern unser Vorgehen aus den Aufgabenteilen a) und b). Seien $\mathcal{A}_{\mathcal{S}'} = (Q_{\mathcal{S}'}, \{R, A, E\}, s_{\mathcal{S}'}, F_{\mathcal{S}'})$ und $\mathcal{A}_{\mathcal{P}'} = (Q_{\mathcal{P}'}, \{R, A, E\}, s_{\mathcal{P}'}, F_{\mathcal{P}'})$ die zwei gegebenen Automaten. Der Algorithmus arbeitet nun wie folgt:

1. Konstruiere den Produktautomaten $\mathcal{B} = \mathcal{A}_{\mathcal{S}'} \times \mathcal{A}_{\mathcal{P}'}$ und wähle $(Q_{\mathcal{S}'} \setminus F_{\mathcal{S}'}) \times F_{\mathcal{P}'}$ als Menge der akzeptierenden Zustände von $\mathcal{B}$.
2. Prüfe, ob $L(\mathcal{B}) \neq \emptyset$ gilt und gib NEIN aus, falls $L(\mathcal{B}) \neq \emptyset$ gilt, sonst JA.

Die Korrektheit lässt sich analog zu den Aufgabenteilen a) und b) begründen.

Beide Eingabe-Automaten sind DFAs. Nach Vorlesung hat der Produktautomat $|Q_{\mathcal{S}'} \times Q_{\mathcal{P}'}| = |Q_{\mathcal{S}'}| \cdot |Q_{\mathcal{P}'}|$ viele Zustände, also müssen für $\mathcal{B}$ höchstens $3(|Q_{\mathcal{S}'}| \cdot |Q_{\mathcal{P}'}|)$ viele Transitionen konstruiert werden (der Faktor 3 spiegelt die Anzahl der möglichen Transitionen pro Zustand wieder und entspricht der Größe des Alphabets $\{R, A, E\}$).

Der Leerheitstest kann nach Vorlesung in Zeit $\mathcal{O}(|\delta_{\mathcal{B}}|)$ durchgeführt werden. Die Anzahl $|\delta_{\mathcal{B}}|$ entspricht gerade der Anzahl der Transitionen von $\mathcal{B}$, also maximal $3(|Q_{\mathcal{S}'}| \cdot |Q_{\mathcal{P}'}|)$. Insgesamt ergibt sich also ein Aufwand von $\mathcal{O}(|Q_{\mathcal{S}'}| \cdot |Q_{\mathcal{P}'}|)$.

d) Ist die Spezifikation durch einen regulären Ausdruck $\alpha_{\mathcal{S}'}$ gegeben, so lässt sich wie folgt vorgehen.

1. Konstruiere ε-NFA $\mathcal{B}_{\mathcal{S}'}$ mit $L(\mathcal{B}_{\mathcal{S}'}) = L(\alpha_{\mathcal{S}'})$.
2. Konstruiere DFA $\mathcal{A}_{\mathcal{S}'}$ mit $L(\mathcal{A}'_{\mathcal{S}}) = L(\mathcal{B}_{\mathcal{S}'})$.
3. Wende den Algorithmus aus Aufgabenteil c) auf $\mathcal{A}_{\mathcal{S}'}$ (und den gegebenen DFA für das Programm) an, und übernehme die Ausgabe.

Nach Vorlesung wissen wir, dass $\mathcal{B}_{\mathcal{S}'}$ Größe $\mathcal{O}(|\alpha_{\mathcal{S}'}|)$ hat. Für den zweiten Schritt erhalten wir mit der Potenzmengenkonstruktion einen DFA $\mathcal{A}_{\mathcal{S}'}$ der Größe $2^{\mathcal{O}(|\alpha_{\mathcal{S}'}|)}$. Kombiniert mit der Laufzeit unseres Algorithmus erhalten wir eine asymptotische Laufzeit in $\mathcal{O}((2^{|\alpha_{\mathcal{S}'}|} \cdot |Q_{\mathcal{P}'}|))$.

Zusatzaufgabe [Kleine NFAs] **2 Punkte**

Wir wissen, dass es für jede reguläre Sprache einen minimalen DFA gibt, der diese Sprache entscheidet. Außerdem wurde in der Vorlesung ein Verfahren vorgestellt, solch einen minimalen DFA systematisch zu konstruieren. Aber gilt dies auch für NFAs?

a) Seien p, q zwei verschiedene Primzahlen mit $p, q < 1336 < pq$. Wir betrachten die Sprache $L = \{a^n \mid n \not\equiv_{pq} 1336\}$, also die Sprache aller Wörter über dem unären Alphabet $\{a\}$ deren Länge modulo pq *nicht* gleich 1336 ist. Gibt es einen NFA mit $p + q + 1$ Zuständen, der L entscheidet? **(2 Punkte)**

b) Sei $K = \{a^n \mid n \neq 1336\}$. Gibt es einen NFA mit weniger als 999 Zuständen, der K entscheidet?

Hinweis

Abgaben für Teilaufgabe b) richten Sie bitte per E-Mail an Christopher Spinrath. Die ersten drei korrekten und nachvollziehbaren Lösungen werden von Thomas Zeume mit einer Süßigkeit aus dem Lehrstuhlvorrat belohnt. Die Abgabefrist ist das Ende des Semesters.

Lösung:

a) Ein Wort a^n ist genau dann nicht in L, wenn $n \not\equiv_{pq} 1336$ gilt. Da p und q verschiedene Primzahlen sind, sind sie insbesondere teilerfremd. Es gilt daher $n \not\equiv_{pq} 1336$ genau dann, wenn $n \not\equiv_p 1336$ oder $n \not\equiv_q 1336$ gilt. Ein NFA, der L entscheidet, kann raten, ob $n \not\equiv_p 1336$ oder aber $n \not\equiv_q 1336$ gilt und seine Behauptung deterministisch verifizieren.

Ob $n \not\equiv_p 1336$ gilt, kann ein DFA mit p Zuständen feststellen (der Transitionsgraph ist ein Kreis der Länge p). Analog kann für den zweiten Fall vorgegangen werden. Der resultierende NFA besteht dann aus einem Startzustand, von dem aus nicht-deterministisch in den DFA mit p Zuständen oder dem DFA mit q Zuständen gewechselt wird.