Finite-State Technology

Teil III:
Automaten

Wiederholung

- Formale Grammatiken sind entweder axiomatische Systeme mit Ableitungsregeln oder Automaten. Beide beschreiben formale Sprachen.
- Formale Sprachen und die sie beschreibenden Automaten:

Reguläre Sprachen - Endliche Automaten

Kontextfreie Sprachen - Kellerautomaten

Kontextsensitive Sprachen - Turingmaschinen

Heutiges Programm

- In den *FST-tools* von Xerox werden (ausschließlich) reguläre Sprachen in Netzwerke kompiliert. Dies ist eine andere Bezeichnung für endliche Automaten (*finite-state machine, finite-state automaton, transducer*). Die Automaten sind epsilonfrei, deterministisch und minimal (Beesley/Karttunen 2003:75).
- Was sind Automaten? Welche verschiedenen Typen gibt es?
- Was heißt epsilonfrei, deterministisch und minimal?

Von der Grammatik zur Implementation

1. Beschreibung natürlicher Sprachen mit komplexen regulären Ausdrücken und/oder regulären Grammatiken

↓

2. Umwandlung in einfache reguläre Ausdrücke

↓

3. Umwandlung in ε-NEA

↓

4. Umwandlung in ε-freien DEA

↓

5. Minimierung des DEA

↓

6. Programm zur Simulation des DEA

Von der Grammatik zur Implementation

- Reguläre Ausdrücke können also direkt in ε-NEA‘s umgewandelt werden.
- Um diese zu verstehen, muß man bei den einfachsten Automaten anfangen, den DEA‘s.

Definition eines DEA

- Ein DEA ist ein Quintupel $A = (Q, \Sigma, \delta, q_0, F)$, wobei

  Q = eine endliche Menge von Zuständen

  Σ = eine endliche Menge von Eingabesymbolen

  q_0 = der Anfangszustand

  F = eine Menge von Endzuständen (akzeptierende Zustände), wobei

  $F \subseteq Q$

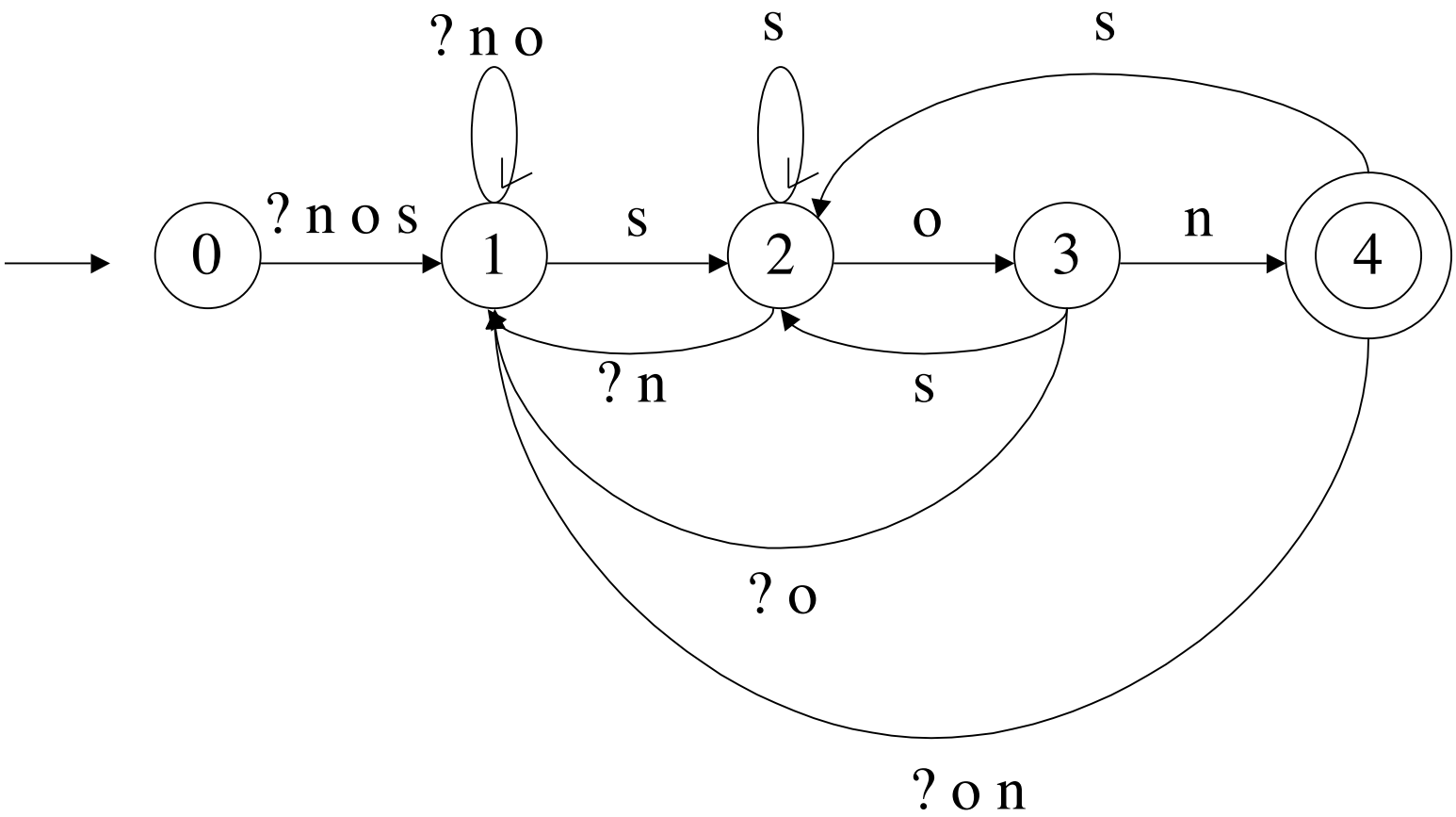
  δ = die Übergangsfunktion nimmt einen Zustand und ein Eingabesymbol als Argument und gibt einen Zustand zurück;

  also $Q \times \Sigma \times Q$

Determinismus

- Der Zustand eines deterministischen Automaten ist genau durch seinen vorherigen Zustand und das Eingabesymbol bestimmt.
- Der Automat kann immer nur in genau einem Zustand sein.
- Für jeden Zustand und jedes Eingabesymbol aus $\sum$ ist ein Folgezustand definiert.
- Beispiel: Die Sprache L(A) sind alle Zeichenreihen, die aus mindestens vier Symbolen bestehen und auf “son” enden.
- *xfst*: read regex [?+ s o n];
- (Im folgenden Diagramm hat “?” die Bedeutung *unknown*.)

Notation 1: Übergangsdiagramme



Notation 2: Übergangstabellen für δ

Eingabe →	?	s	o	n
Zustand↓				
→ S0	S1	S1	S1	S1
S1	S1	S2	S1	S1
S2	S1	S2	S3	S1
S3	S1	S2	S4	S1
*S4	S1	S2	S1	S1

Die Felder in der Tabelle geben an, wie man von einem Zustand (Spalte ganz links) und einem Eingabesymbol (oberste Reihe) zu einem Folgezustand gelangt.

Definition des Beispiel-DEA

- DEA A = ({S0, S1, S2, S3, S4}, {s, o, n, ?}, S0, {S4}, δ)
- Noch einmal explizit:

  {S0, S1, S2, S3, S4} = Q

  {s, o, n, ?} = Σ

  S0 = q_0

  {S4} = F

  δ ist die Übergangsfunktion, die durch die Übergangstabelle beschrieben wird.

Übergangsdiagramme in *xfst*

- Beispiel: read regex [?+ s o n];
- Was zeigt der Befehl *print net* genau an?
- Was ist der Unterschied zwischen *any* und *unknown*?
- Was zeigt der Befehl *print sigma*?

- Beispiel: read regex [s o n];
- Welches Netzwerk wird hier ausgegeben? Ist dies ein DEA?

Erweiterte Übergangsfunktion

- Bisher können wir einzelne Zeichen verarbeiten, also von einem Zustand und einem Zeichen zu einem Folgezustand gehen.
- Wie können wir die Übergangsfunktion definieren, damit ganze Zeichenreihen verarbeitet werden?
- Dies geschieht induktiv:

Basis: $\delta\hat{}(q, \varepsilon) = q$

Induktionsschritt: Angenommen, w besteht aus einer Zeichenreihe xa, wobei x eine beliebig lange Zeichenreihe und a das letzte einzelne Symbol von w ist; dann gilt:

$\delta\hat{}(q, w) = \delta(\delta\hat{}(q, x)a)$

Erweiterte Übergangsfunktion

- Wie liest man: δ^(q, w) = δ(δ^(q, x)a)?
- Man bricht eine Zeichenreihe w in 2 Teile auf: Das letzte Einzelsymbol und die davor stehende Zeichenreihe x.
- Dann arbeitet man sich solange nach links vor, bis x nur noch das leere Symbol ist.
- Dann nimmt man den Anfangszustand des Automaten und das leere Symbol als Eingabe und bestimmt den Folgezustand.
- Dann nimmt man diesen Zustand und das nächste Symbol und bildet wiederum den Folgezustand. Man arbeitet sich also nach rechts durch.
- Dies wiederholt man, bis die gesamte Zeichenreihe verarbeitet ist.
- Beispiel…

Beispiel: Analysiere “larsson”

- δ^(S0, larsson) = δ(δ^(δ^(δ^(δ^(δ^(δ^(δ^S0,ε)l)a)r)s)s)o)n)

- δ^(S0, ε) = S0
- δ^(S0, l) = δ(δ^(S0, ε)l) = δ(S0, l) = S1
- δ^(S0, la) = δ(δ^(S0, l)a) = δ(S1, a) = S1
- δ^(S0, lar) = δ(δ^(S0, la)r) = δ(S1, r) = S1
- δ^(S0, lars) = δ(δ^(S0, lar)s) = δ(S1, s) = S2
- δ^(S0, larss) = δ(δ^(S0, lars)s) = δ(S2, s) = S2
- δ^ (S0, larsso) = δ(δ^ (S0, larss)o) = δ(S2, o) = S3
- δ^ (S0, larsson) = δ(δ^ (S0, larsso)n) = δ(S3, n) = S4

Beispiel: Analysiere “larsson”

- $\delta\hat{}(S0, larsson) =$

  $\delta(\delta\hat{}(\delta\hat{}(\delta\hat{}(\delta\hat{}(\delta\hat{}(\delta\hat{}S0,\varepsilon)l)a)r)s)s)o)n) = \delta(S3, n) = S4$

- Die Zeichenreihe wird akzeptiert, da $\{S4\} \subseteq F$.

Die Sprache eines DEA

- Die Sprache, die ein DEA A beschreibt, ist die Menge aller Zeichenreihen w, die mit der Übergangsfunktion vom Startzustand in einen akzeptierenden Zustand führen.
- Formal: L(A) = {w | δ^(qo, w) ist in F}.
- Diese Sprache ist eine reguläre Sprache.

Definition eines NEA

- Ein NEA ist ein Quintupel $A = (Q, \Sigma, \delta, q_0, F)$, wobei

  Q = eine endliche Menge von Zuständen

  Σ = eine endliche Menge von Eingabesymbolen

  q_0 = der Anfangszustand

  F = eine Menge von Endzuständen (akzeptierende Zustände), wobei

  $F \subseteq Q$

  δ = die Übergangsfunktion nimmt einen Zustand und ein Eingabesymbol als Argument und gibt eine Teilmenge von Q zurück; also $Q \times \Sigma \times 2^Q$

Definition eines NEA

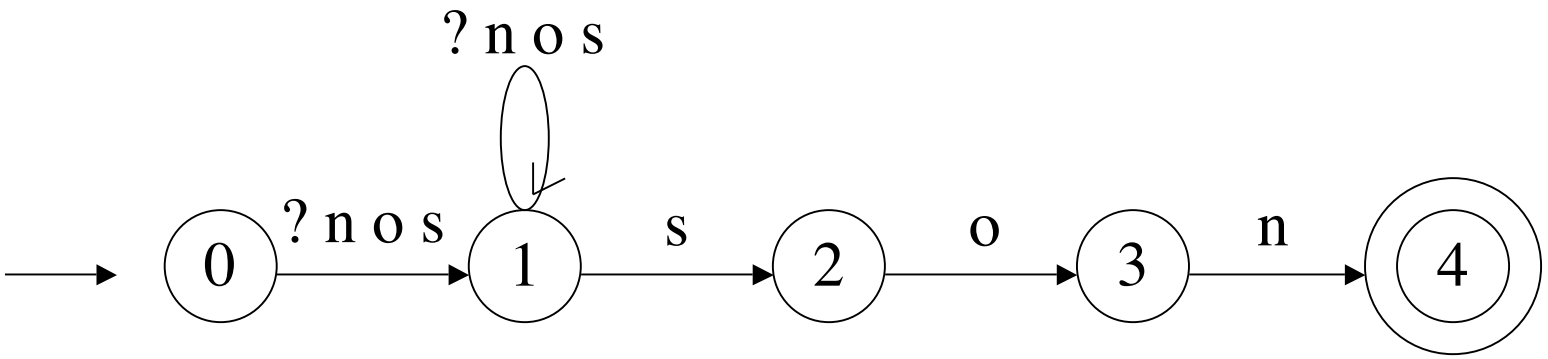
- NEA‘s können gleichzeitig in mehreren Zuständen sein.
- Ein Vorteil ist, daß dadurch die Beschreibung von Sprachen sehr vereinfacht wird, wie man an folgendem Beispiel sehen kann, das genau dieselbe Sprache wie der Beispiel-DEA beschreibt.

  (Die Sprache L(A) sind alle Zeichenreihen, die aus mindestens vier Symbolen bestehen und auf “son” enden.

  xfst: read regex [?+ s o n];

  (Im folgenden Diagramm hat “?” die Bedeutung *unknown*.))

Beispiel NEA: Übergangsdiagramm



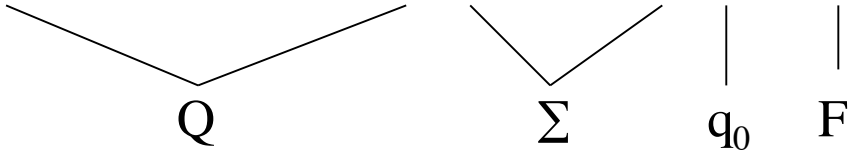
Beispiel NEA: Übergangstabelle für δ

Input →	?	s	o	n
State↓				
→ S0	{S1}	{S1}	{S1}	{S1}
S1	{S1}	{S1, S2}	{S1}	{S1}
S2	∅	∅	{S3}	∅
S3	∅	∅	∅	{S4}
*S4	∅	∅	∅	∅

Die Tabelle wird ähnlich gelesen, wie beim DEA.
Man beachte, daß in den einzelnen Feldern diesmal Mengen stehen.

Definition des Beispiel-NEA

- NEA A = ({S0, S1, S2, S3, S4}, {s, o, n, ?}, S0, {S4}, δ)



Erweiterte Übergangsfunktion für NEA's

- Wir brauchen wieder eine erweiterte Übergangsfunktion, um ganze Zeichenreichen verarbeiten zu können. Dies wird ähnlich definiert, wie bei den DEA's.
- Allerdings geht es jetzt nicht um einzelne Folgezustände, sondern um Mengen von Folgezuständen.

Erweiterte Übergangsfunktion für NEA's

- Basis: $\delta\hat{}(q, \varepsilon) = q$
- Induktionsschritt: Angenommen, w ist eine Zeichenreihe xa, wobei x eine beliebig lange Zeichenreihe und a das letzte Symbol von w ist. Außerdem:

  $\delta\hat{}(q, x) = \{p_1, p_2, ..., p_k\}$

  $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, ..., r_m\}$

- Dann gilt: $\delta\hat{}\,(q, w) = \{r_1, r_2, ..., r_m\}$

Erweiterte Übergangsfunktion für NEA's

- Erklärung: $\delta\hat{}(q, x) = \{p_1, p_2, ..., p_k\}$

  Dies bedeutet, daß man nach der Verarbeitung von x in eine Menge von Folgezuständen $\{p_1, p_2, ..., p_k\}$ kommen kann.

- Erklärung:

  $$\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, ..., r_m\}$$

  Dies bedeutet, daß man von diesen Folgezuständen $\{p_1, p_2, ..., p_k\}$ und dem Symbol a wiederum zu einer Menge von Folgezuständen $\{r_1, r_2, ..., r_m\}$ kommen kann.

Erweiterte Übergangsfunktion für NEA's

- Erklärung: $\delta\hat{}\ (q, w) = \{r_1, r_2, ..., r_m\}$

  Dann kommt man auch nach der Verarbeitung der ganzen Zeichenreihe w (=xa) in diese Folgezustände $\{r_1, r_2, ..., r_m\}$.

Beispiel: Analysiere “larsson”

- $\delta\hat{}(S0, \varepsilon) = \{S0\}$
- $\delta\hat{}(S0, l) = \delta(S0, l) = \{S1\}$
- $\delta\hat{}(S0, la) = \delta(S1, a) = \{S1\}$
- $\delta\hat{}(S0, lar) = \delta(S1, r) = \{S1\}$
- $\delta\hat{}(S0, lars) = \delta(S1, s) = \{S1, S2\}$
- $\delta\hat{}(S0, larss) = \delta(S1, s) \cup \delta(S2, s) = \{S1, S2\} \cup \varnothing = \{S1, S2\}$
- $\delta\hat{}(S0, larsso) = \delta(S1, o) \cup \delta(S2, o) = \{S1\} \cup \{S3\} = \{S1, S3\}$
- $\delta\hat{}(S0, larsson) = \delta(S1, n) \cup \delta(S3, n) = \{S1\} \cup \{S4\} = \{S1, S4\}$

- Die Zeichenreihe wird akzeptiert, wenn $\{S4\} \subseteq F$ zutrifft.

Die Sprache eines NEA

- Die Sprache, die ein NEA A beschreibt, ist die Menge aller Zeichenreihen, die vom Startzustand in einen akzeptierenden Zustand führen.
- Formal: $L(A) = \{w \mid \delta\hat{}(q_0, w) \cap F \neq \emptyset\}$
- Die Menge von Zuständen, die ich von q_0 und w aus erreiche, ist also nicht leer.

DEA's vs. NEA's: Kompilierung

- Wie unterscheidet sich die Anzahl der Zustände und Übergänge zwischen NEA's und DEA's?
- Nachteil DEA: DEA's haben mehr Übergänge als NEA's. Meist haben sie eine ähnliche Anzahl von Zuständen, im schlimmsten Fall aber 2^{Qn} Zustände, wobei Q_n die Anzahl der Zustände von NEA ist.
- DEA's verbrauchen also in der Regel mehr Rechenkraft beim Kompilieren des Netzwerkes als NEA's.

DEA’s vs. NEA’s: Ambiguität

- Wie werden ambige Eingaben verarbeitet?
- Vorteil DEA: DEA’s haben keine ambigen Eingaben.
- Wie wird Ambiguität bei NEA’s aufgelöst: *backup, look ahead, parallel processing*.
- Diese Operationen erfordern beim Verarbeiten von Eingaben mehr Arbeitsspeicher und/oder Rechenleistung als die Verarbeitung bei DEA’s.

- Sowohl DEA’s als auch NEA’s können von einem Programm simuliert werden. In *xfst* wird mit DEA’s gearbeitet.

Äquivalenz von DEA's und NEA's

- Jeder NEA läßt sich in einen DEA umwandeln.
- Prozedur (informell):

  a) Bilde die Potenzmenge von Q_n: $Q_d = 2^{Qn}$.

  b) Gib für jede Teilmenge der Potenzmenge an, zu welchen Zuständen sie bei allen Eingaben führen. Diese Teilmengen bilden die neuen Zustände des DEA.

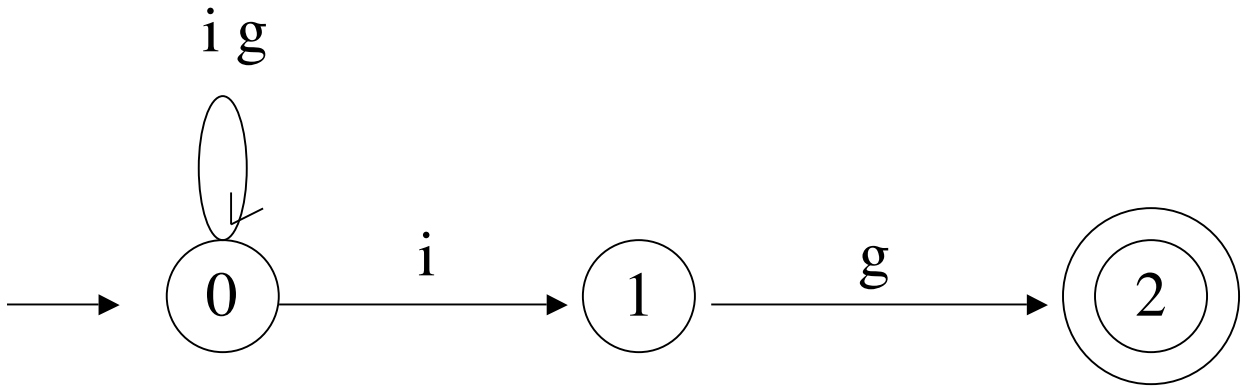
  c) Fakultativ: Benenne die Zustände um.

  d) Streiche die unerreichbaren Zustände.

- Beispiel...

Beispiel: NEA -> DEA

NEA mit Σ = {i, g}:



Sprache L(A) besteht aus alle Zeichenreihen aus beliebig vielen “i” und “g”, die auf “ig” enden und deren Mindestlänge 2 Symbole ist.

xfst: read regex [[i | g]* i g];

NEA -> DEA: Schritt 1

- Bilde die Potenzmenge 2^{Q_n} von Q_n.
- Diese Teilmengen bilden die Zustände des DEA.
- "→" steht für den Startzustand, " * " für den akzeptierenden Zustand.

NEA -> DEA: Schritt 1

	2^Q	i	g	i	g
	∅				
	→{0}				
	{1}				
	*{2}				
	{0,1}				
	*{0,2}				
	*{1,2}				
	*{0,1,2}				

NEA -> DEA: Schritt 2

- Bestimme, wie man von jeder dieser Teilmengen und den Eingabesymbolen zu Folgezuständen kommt. Beispiel:

  Von $\varnothing$ und i ist kein Folgezustand definiert. Also ist der Folgezustand wiederum $\varnothing$ (die leere Menge). Dasselbe gilt für die Eingabe g.

  Von {0} und i ist als Folgezustand die Menge {0, 1} definiert. Für 0 und g ist es die Menge {0}.

  Von {0, 1} kommt man mit i zu {0, 1}, da man von {0} und i zu {0, 1} kommt, von {1} und i zu $\varnothing$ und weil $\{0,1\} \cup \varnothing = \{0, 1\}$.

  ...

NEA -> DEA: Schritt 2

	2^Q	i	g	i	g
	∅	∅	∅		
	→{0}	{0, 1}	{0}		
	{1}	∅	{2}		
	*{2}	∅	∅		
	{0,1}	{0, 1}	{0, 2}		
	*{0,2}	{0, 1}	{0}		
	*{1,2}	∅	{2}		
	*{0,1,2}	{0, 1}	{0, 2}		

NEA -> DEA: Schritt 3

- Die Zustände können jetzt der Einfachheit halber umbenannt werden. Beispiel:

  $\varnothing$ = A
  {0} = B

  ...

NEA -> DEA: Schritt 3

	2^Q	i	g	i	g
A	∅	∅	∅	A	A
→ B	→{0}	{0, 1}	{0}	E	B
C	{1}	∅	{2}	A	D
*D	*{2}	∅	∅	A	A
E	{0,1}	{0, 1}	{0, 2}	E	F
*F	*{0,2}	{0, 1}	{0}	E	B
*G	*{1,2}	∅	{2}	A	D
*H	*{0,1,2}	{0, 1}	{0, 2}	E	F

NEA -> DEA: Schritt 4

- Die nicht erreichbaren Zustände können jetzt gestrichen werden. Man folgt also dem Startzustand und behält nur die Zustände, die man von ihm aus in einer Ableitung erreichen kann. Beispiel:

  A kann nie von B aus (über eine Ableitung) erreicht werden. Ebenso fällt D weg. ...

- Übrig bleiben 3 Zustände: B, E, F.

NEA -> DEA: Schritt 4

	2^Q	i	g	i	g
A	∅	∅	∅	A	A
→ **B**	→{0}	{0, 1}	{0}	**E**	**B**
C	{1}	∅	{2}	A	D
*D	*{2}	∅	∅	A	A
E	{0,1}	{0, 1}	{0, 2}	**E**	**F**
***F**	*{0,2}	{0, 1}	{0}	**E**	**B**
*G	*{1,2}	∅	{2}	A	D
*H	*{0,1,2}	{0, 1}	{0, 2}	E	F

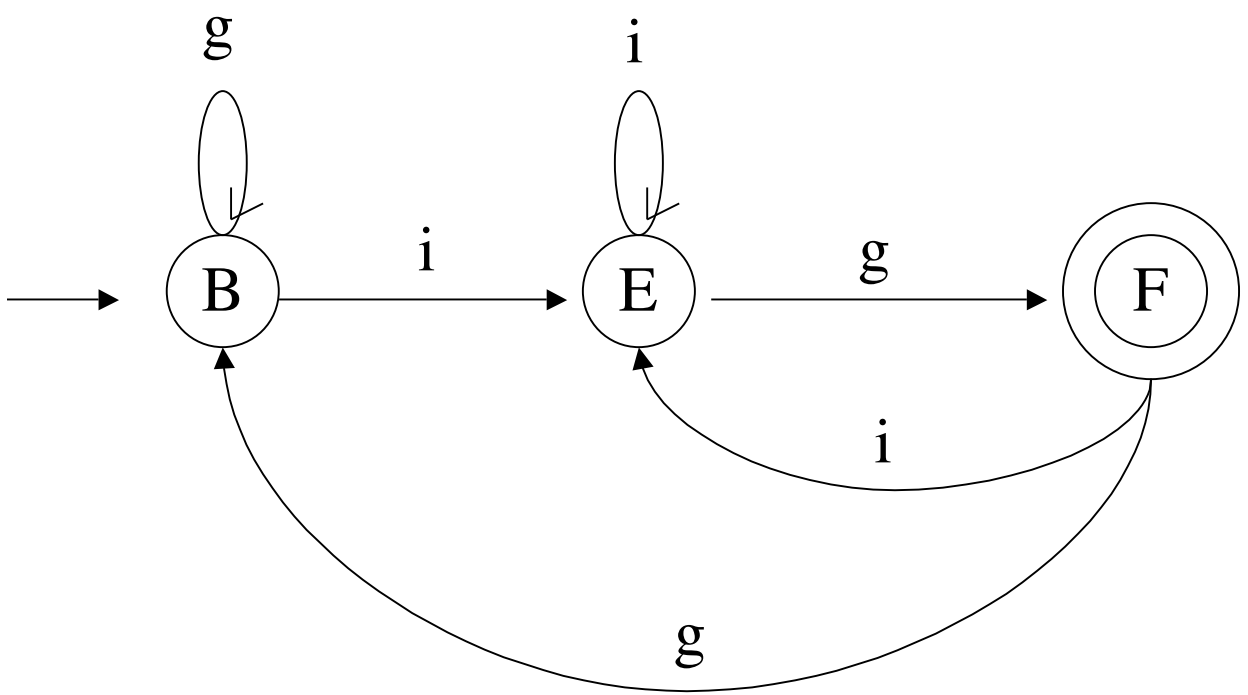
NEA -> DEA: Schritt 4

	2^Q	i	g	i	g
→ B	→{0}	{0, 1}	{0}	**E**	**B**
E	{0,1}	{0, 1}	{0, 2}	**E**	**F**
***F**	*{0,2}	{0, 1}	{0}	**E**	**B**

Dies ist die reduzierte Übergangstabelle für den DEA.
Muß nur noch gezeichnet werden…

Beispiel: NEA -> DEA

Neuer DEA:



Beispiel: NEA -> DEA

- Überprüfe das Ergebnis in *xfst* mit:

  a) read regex [[i | g]* i g];
  b) print net

Definition eines ε-NEA

- Ein ε-NEA ist ein Quintupel $A = (Q, \Sigma, \delta, q0, F)$, wobei

  Q = eine endliche Menge von Zuständen

  Σ = eine endliche Menge von Eingabesymbolen

  $q0$ = der Anfangszustand

  F = eine Menge von Endzuständen (akzeptierende Zustände), wobei $F \subseteq Q$

  δ = die Übergangsfunktion nimmt einen Zustand und ein Eingabesymbol aus $\Sigma \cup \{\varepsilon\}$ als Argument und gibt eine Teilmenge von Q zurück, also $Q \times (\Sigma \cup \{\varepsilon\}) \times 2^Q$

Definition eines ε-NEA

- Ein ε-NEA kann mit der leeren Zeichnenkette in einen nächsten Zustand überzugehen.
- Da ε nicht Teil von ∑ ist, kann der Automat also ohne Verarbeitung eines Zeichens in den nächsten Zustand übergehen.
- Allerdings muß dies in δ angegeben werden.
- Dies vereinfacht ebenfalls die Umwandlung von regulären Ausdrücken in Automaten.
- Die Definition ist ähnlich wie für den NEA.

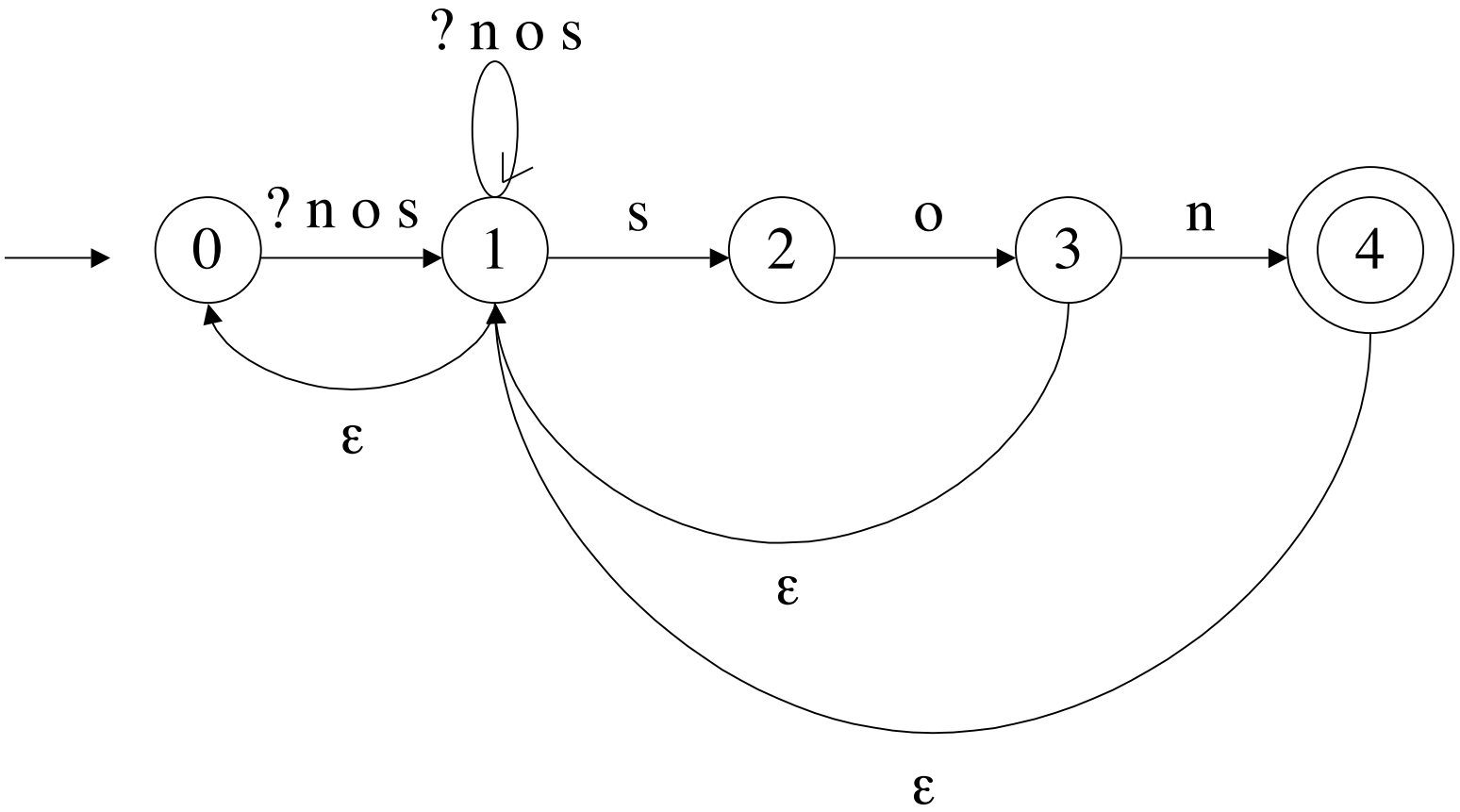
- Beispielsprache: Wie für den DEA und den NEA...

Beispiel ε-NEA: Übergangstabelle für δ

Input →	?	s	o	n	ε
State↓					
→ S0	{S1}	{S1}	{S1}	{S1}	∅
S1	{S1}	{S1, S2}	{S1}	{S1}	{S0}
S2	∅	{S3}	∅	∅	∅
S3	∅	∅	{S4}	∅	{S1}
*S4	∅	∅	∅	∅	{S1}

Frage: Wie sieht das Übergangsdiagramm aus?

Beispiel ε-NEA: Übergangsdiagramm



ε-Hülle

- Zum Verarbeiten von Eingaben braucht man die ε-Hülle.
- Definition von ε-Hülle:

  Basis: Zustand q ist in ε-Hülle(q) enthalten

  Induktionsschritt: Wenn der Zustand p in ε-Hülle(q) enthalten ist und es einen Übergang vom Zustand p zum Zustand r mit der Beschriftung ε gibt, dann ist r in ε-Hülle(q) enthalten

- Im vorigen Beispiel: ε-Hülle(S0) = {S0}; ε-Hülle (S1) = {S1, S2}

ε-Hülle

- Die ε-Hülle besagt, daß man von einem Zustand über ein Eingabesymbol zu einer Menge von Folgezuständen gelangt. In diese Menge von Folgezuständen sind dann noch all jene Zustände einzuschließen („einzuhüllen“), die man durch Verarbeitung von ε erhält.

Erweiterte Übergangsfunktion

- Basis: $\delta\hat{}(q, \varepsilon) = \varepsilon$-Hülle$(q)$.

- Induktionsschritt: Angenommen, w ist eine Zeichenreihe xa, wobei x eine beliebig lange Zeichenreihe und a das letzte Symbol von w ist. Außerdem:

  $\delta\hat{}(q, x) = \{p_1, p_2, ..., p_k\}$

  $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, ..., r_m\}$

- Dann gilt: $\delta\hat{}\,(q, w) = \bigcup_{j=1}^{m} \varepsilon$-Hülle$\{r_j\}$.

Beispiel: Analysiere “larsson”

- Basis:

  $\delta^{\wedge}(S0, \varepsilon) = \varepsilon\text{-Hülle}(S0) = \{S0\}$

Beispiel: Analysiere “larsson”

- $\delta(S0, l) = \{S1\}$
- $\delta\hat{}(S0, l) = \varepsilon\text{-Hülle}(S1) = \{S0, S1\}$
- $\delta(S0, a) \cup \delta(S1, a) = \{S1\} \cup \{S1\} = \{S1\}$
- $\delta\hat{}(S0, la) = \varepsilon\text{-Hülle}(S1) = \{S0, S1\}$
- $\delta(S0, r) \cup \delta(S1, r) = \{S1\} \cup \{S1\} = \{S1\}$
- $\delta\hat{}(S0, lar) = \varepsilon\text{-Hülle}(S1) = \{S0, S1\}$
- $\delta(S0, s) \cup \delta(S1, s) = \{S1\} \cup \{S1, S2\} = \{S1, S2\}$
- $\delta\hat{}(S0, lars) = \varepsilon\text{-Hülle}(S1) \cup \varepsilon\text{-Hülle}(S2) = \{S0, S1\} \cup \{S2\} = \{S0, S1, S2\}$
- $\delta(S0, s) \cup \delta(S1, s) \cup \delta(S2, s) = \{S1\} \cup \{S1, S2\} \cup \varnothing = \{S1, S2\}$
- $\delta\hat{}(S0, larss) = \varepsilon\text{-Hülle}(S1) \cup \varepsilon\text{-Hülle}(S2) = \{S0, S1\} \cup \{S2\} = \{S0, S1, S2\}$
- $\delta(S0, o) \cup \delta(S1, o) \cup \delta(S2, o) = \{S1\} \cup \{S1\} \cup \{S3\} = \{S1, S3\}$
- $\delta\hat{}(S0, larsso) = \varepsilon\text{-Hülle}(S1) \cup \varepsilon\text{-Hülle}(S3) = \{S0, S1\} \cup \{S1, S3\} = \{S0, S1, S3\}$
- $\delta(S0, n) \cup \delta(S1, n) \cup \delta(S3, n) = \{S1\} \cup \{S1\} \cup \{S4\} = \{S1, S4\}$
- $\delta\hat{}(S0, larsson) = \varepsilon\text{-Hülle}(S1) \cup \varepsilon\text{-Hülle}(S4) = \{S0, S1\} \cup \{S1, S4\} = \{S0, S1, S4\}$

Beispiel: Analysiere “larsson”

- Die Zeichenreihe wird akzeptiert, wenn $\{S4\} \subseteq F$ zutrifft.

Literatur

- Beesley/Karttunen (2003:II-III)
- Jurafsky/Martin 2000:2.2
- Partee et al. 1993:17.1
- Roche/Schabes 1997:I
- Hopcroft/Motwani/Ullman (2001)