Lektion 14

Zunächst werden wir ein Beispiel für die Anwendung von nichtdeterministischen Automaten und deren Transformation in deterministische Automaten vorstellen.

Danach werden den Zusammenhang zwischen regulären Ausdrücken und endlichen Automaten untersuchen. In dieser Lektion werden Sie sehen, dass zu jedem regulären Ausdruck ein ε-NEA konstruiert werden kann, der die Sprache des Ausdrucks akzeptiert. Um zu zeigen, dass das vorgestellt Konstruktionsverfahren für jeden regulären Ausdruck einen passenden Automaten generiert, wird ein spezielles Beweisverfahren verwendet, die sog. *strukturelle Induktion*.

8.3.3 Anwendungsbeispiel: Effiziente Suche in Texten

Beispiel 8.19 - Effiziente Suche nach Teilwörtern

In einem langen Text der Länge k über Alphabet $\Sigma = \{a_1, ..., a_m\}$ soll geprüft werden, ob eines von mehreren Suchwörtern $w_1, w_1, ..., w_n$ irgendwo darin vorkommt.

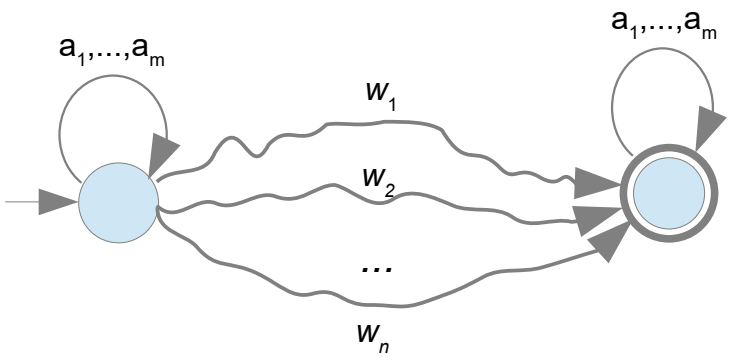
Ein Anwendungsfall wäre beispielsweise, dass ein Virenscanner eine Datei (= lange Byte-Folge) nach vielen verschiedenen Viren-Signaturen (= Teilfolgen) effizient durchsuchen muss.

Naive Lösung:

- Text jeweils Zeichen für Zeichen durchgehen. An jeder Position mit allen Suchwörtern vergleichen
- Rechenaufwand proportional $n \cdot k$, d.h. Prüfaufwand hängt von der Zahl der Suchwörter und der Länge der Zeichenkette ab.

Effiziente Lösung mittels nichtdeterministischer und deterministischer Automaten:

1. ε-NEA für Suche nach den Wörtern w_1 bis w_n nach folgendem Schema bilden (ε-Übergänge werden hier nicht gebraucht):



2. ε–NEA durch Teilmengenkonstruktion in einen DEA umwandeln. DEA kann effizient implementiert werden.

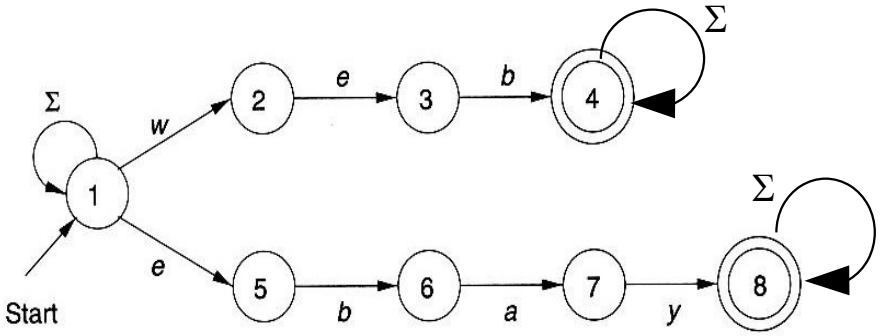
3. Text mit DEA analysieren.

Aufwand für Suche ist nur proportional zu Textlänge *k*, unabhängig von der Anzahl der Suchwörter.

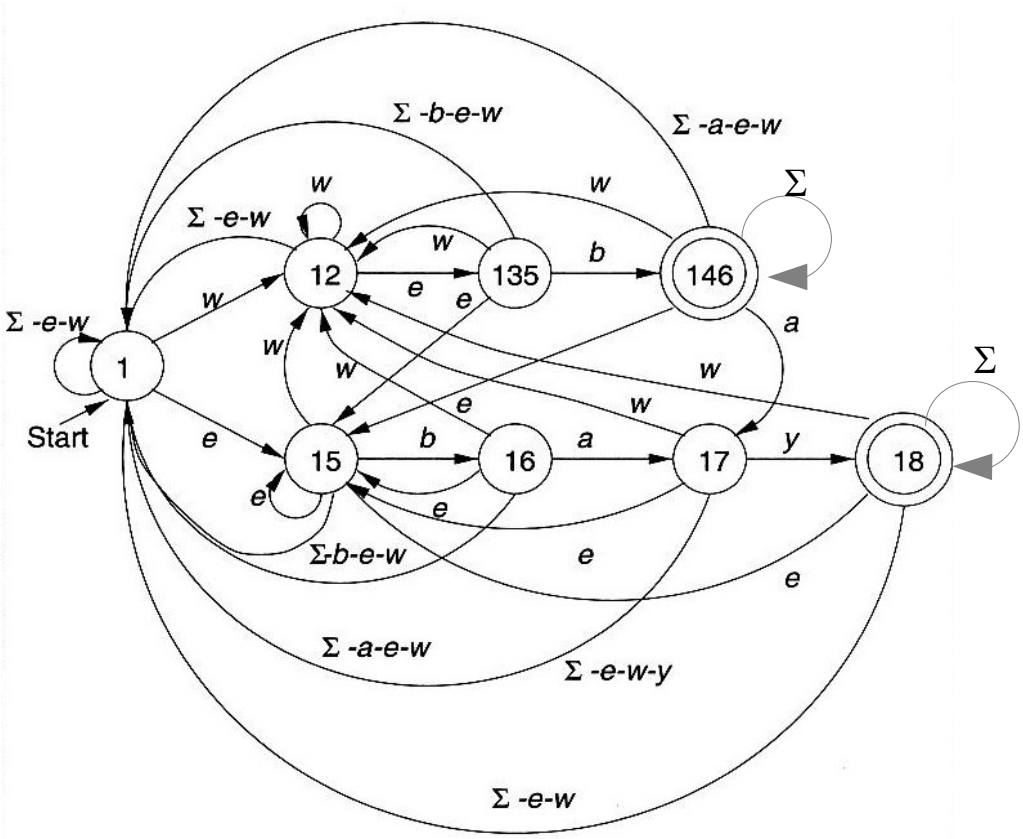
Beispiel 8.20 - effiziente Suche in Texten

Suche nach `web` oder `ebay` in Zeichenketten über Alphabet $\Sigma = \{a,\ldots,z\}$.

- ε-NEA für Suche:



- Ergebnis der Umwandlung in DEA:



[Quelle: Hopcroft/Motwani/Ullmann: Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie]

8.3.4 Äquivalenz von ε-NEAs und DEAs

Eigenschaft 8.21 - Gleiche Mächtigkeit von DEAs und ε-NEAs

Die Menge der von ε-NEAs akzeptierten Sprachen stimmt mit der Menge der von DEAs akzeptierten Sprachen überein, d.h. deterministische und nichtdeterministische endliche Automaten sind gleich mächtig.

Beweis

(1) Jeder DEA kann auch als ein äquivalenter ε-NEA betrachtet werden.
⇒ ε-NEAs können alle Sprachen akzeptieren, die DEAs akzeptieren.

(2) Zu jedem ε-NEA kann mit der Teilmengenkonstruktion ein äquivalenter DEA gebildet werden.
⇒ DEAs können alle Sprachen akzeptieren, die ε-NEAs akzeptieren.

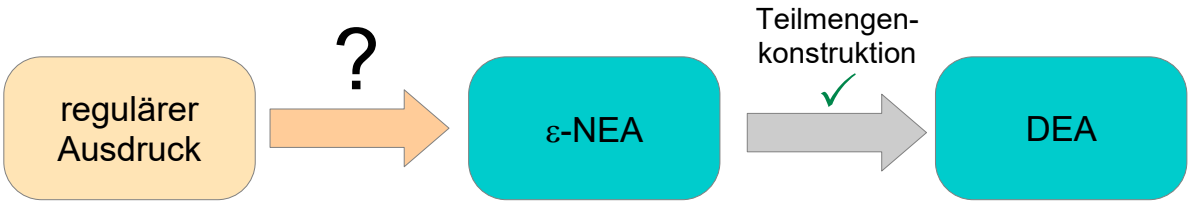
Kap. 9 Reguläre Sprachen

In diesem Kapitel wird der Zusammenhang zwischen regulären Ausdrücken, endlichen Automaten (DEA und ε-NEA) und Chomsky-Typ-3-Grammatiken untersucht. Ein Ergebnis wird sein, dass alle dieser Beschreibungsmethoden die gleiche Klasse von Sprachen beschreiben können, die sog. *regulären Sprachen*.

9.1 Reguläre Ausdrücke und endliche Automaten

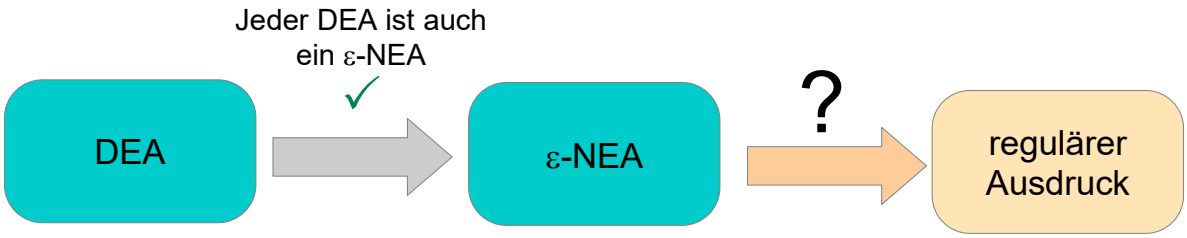
Reguläre Ausdrücke und endlichen Automaten können äquivalent ineinander umgewandelt werden.

- Zunächst wird vorgestellt, wie zu einem regulären Ausdruck ein nicht-deterministischer und dann auch ein deterministischer endlicher Automat gebildet werden kann.



Dies findet praktische Anwendung überall dort, wo geprüft werden muss, ob ein Wort zu einem regulären Ausdruck passt, d.h. zur Sprache des regulären Ausdrucks gehört, z.B. in Scanner-Generatoren oder auch in Funktionen der Java-Standardbibliothek, die mit regulären Ausdrücken arbeiten.

- Danach wird dann auch die umgekehrte Richtung betrachtet, d.h. wie zu einem ε-NEA oder DEA ein regulärer Ausdruck gebildet werden kann, der die Sprache des Automaten beschreibt.



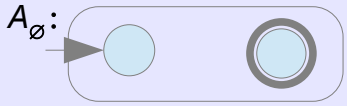
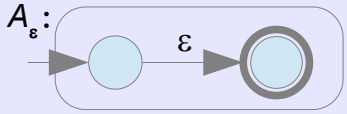
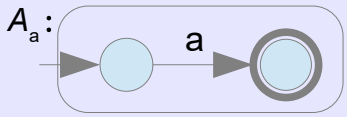
Dass dies immer geht, ist ein theoretisch interessantes Ergebnis, denn es ist nicht so leicht ersichtlich, dass Automaten, die völlig unstrukturiert, "spaghetti-artig" aufgebaut sein können, in die klar strukturierte Form von regulären Ausdrücken umgewandelt werden können.

9.1.1 Umwandlung regulärer Ausdrücke in ε-NEAs

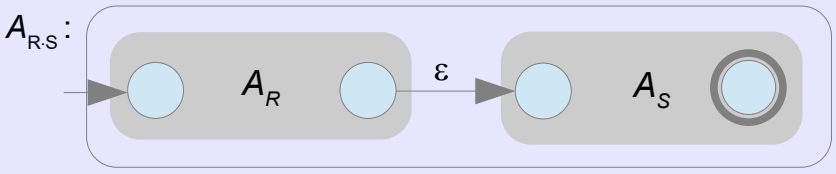
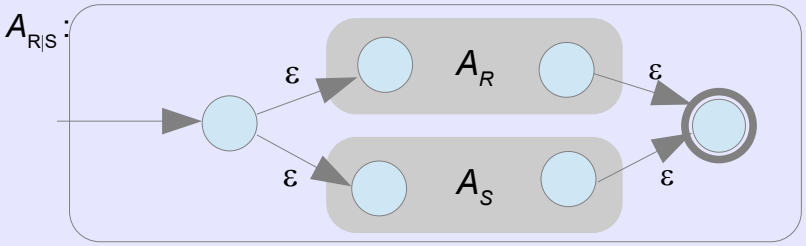
Mit dem folgenden Verfahren lässt sich jeder reguläre Ausdruck in einen äquivalenten ε-NEA umwandeln. Die Vorgehensweise orientiert sich an der syntaktischen Struktur des regulären Ausdrucks, analog wie auch die Semantik der Ausdrücke definiert wurde. Man beginnt mit den elementaren Teilausdrücken und kann dann nach und nach die Automaten für die zusammengesetzten Ausdrücke bilden.

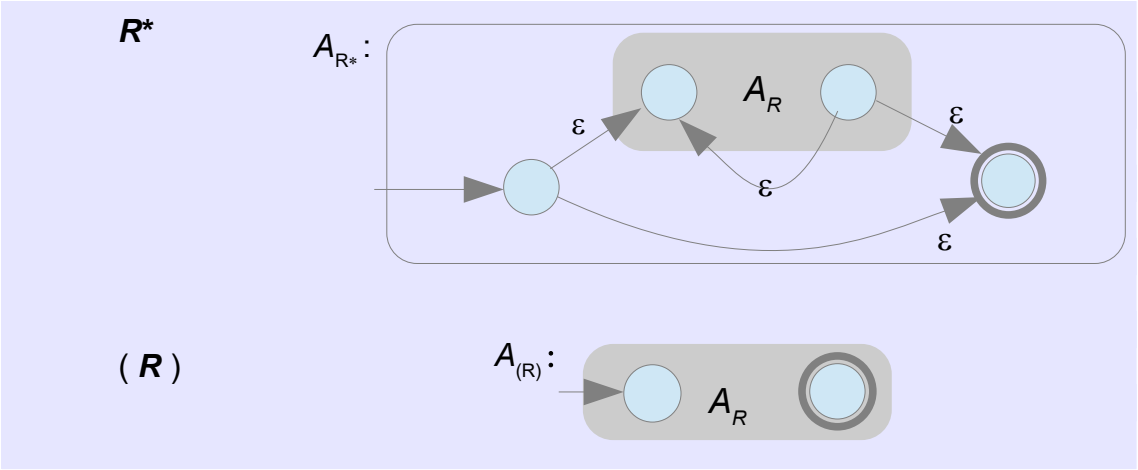
Verfahren 9.1 - Induktive Konstruktion eines ε-NEA zu regulärem Ausdruck

- Für elementare reguläre Ausdrücke kann ein Automat direkt angegeben werden:

regulärer Ausdruck	**ε-NEA**
∅	
ε	
a	

- Für zusammengesetzte reguläre Ausdrücke wird der Automat aus den Automaten für die Teilausdrücke gebildet (hier als graue Fläche dargestellt)

regulärer Ausdruck	**ε-NEA**
R*·*S	
R* | *S	



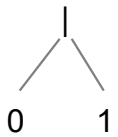
Anmerkungen

- Für die Konstruktion der Automaten gilt folgende Invariante:
  - Jeder konstruierte Automat hat genau einen Endzustand (und einen Startzustand).
  - Der Startzustand hat nur davon ausgehende Zustandsübergänge und keine eingehende Zustandsübergänge
  - Der Endzustand hat nur eingehende Zustandsübergänge, aber keine ausgehende Zustandsübergänge.
- In der Literatur finden Sie teilweise auch andere Regeln, wie ein Automat für einen regulären Ausdruck gebildet werden kann. Die hier vorgestellten Regeln haben den Vorteil, dass das Verfahren einfach zu implementieren (und auch einfach auf dem Papier nachzuvollziehen) ist, da immer nur Zustände oder Zustandsübergänge dazugenommen werden, aber nie etwas gelöscht oder vereinigt werden muss.

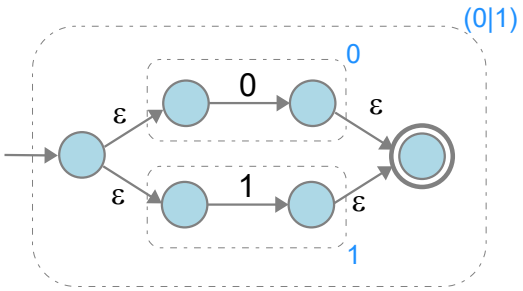
Beispiel 9.2 - ε-NEA für regulären Ausdruck

ε-NEA für regulären Ausdruck (0|1) bilden:

- Syntaktischer Aufbau des Ausdrucks:



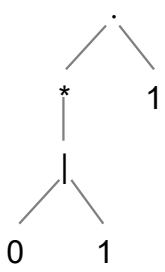
- gebildeter ε-NEA:



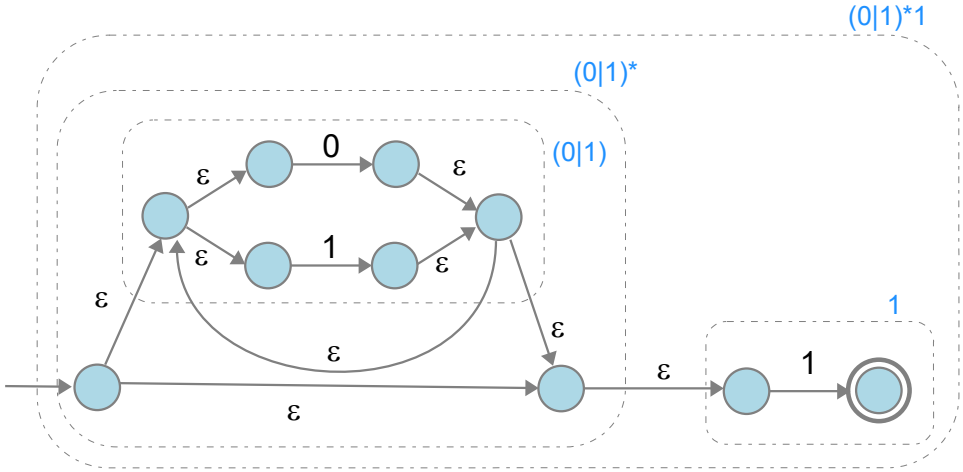
Beispiel 9.3 - ε-NEA für regulären Ausdruck

ε-NEA für regulären Ausdruck (0|1)*1 bilden:

- Syntaktische Struktur:



- Induktiv konstruierter ε-NEA:



Aufgabe 9.4 - ε-NEA für regulären Ausdruck

Konstruieren Sie einen ε-NEA zum regulären Ausdruck
(0|1)*1(0|1).

Eigenschaft 9.5 - Äquivalenz von regulärem Ausdruck und ε-NEA

Der mit Algorithmus 9.1 zu einem regulären Ausdruck *R* konstruierte ε-NEA A_R akzeptiert genau die Sprache, die durch den regulären Ausdruck dargestellt wird, d.h. $L(R) = L(A_R)$

Der Beweis für diese Eigenschaft folgt später. Es wird dafür das Beweisverfahren der Strukturellen Induktion verwendet.

9.1.2 Strukturelle Induktion

Wie kann man beweisen, dass eine Eigenschaft für jeden beliebigen regulären Ausdruck gilt, d.h. für eine unendliche Menge unterschiedlicher Ausdrücke?

Um Eigenschaften für unendliche Mengen nachzuweisen, verwendet man *Induktionsverfahren*. Sie kennen aus der Mathematik bereits das Beweisverfahren

der v*ollständigen Induktion,* mit dem man nachweist, dass eine Eigenschaft für jede natürliche Zahl gilt. Vollständige Induktion ist hier allerdings nicht direkt anwendbar, da es sich nicht um Zahlen, sondern um reguläre Ausdrücke, also strukturiert aufgebaute Objekte, handelt.

Ein Beweisverfahren, um eine Eigenschaft für eine unendliche Menge strukturierter Objekte zu zeigen, die nach gegebenen Bildungsregeln aufgebaut sind, ist die sog. *strukturelle Induktion*, die nachfolgend vorgestellt wird. Sie ist eine wichtige Denkweise für Informatiker, da in der Informatik sehr viel mit strukturierten Objekten gearbeitet wird, z.B. bei XML-Dokumenten, Programmcode, Bäumen oder baumartigen Datenstrukturen.

Eine Möglichkeit, um unendliche Mengen zu definieren, besteht darin anzugeben, wie die Elemente der Menge nach und nach mit einem endlichen Satz von Konstruktionsregeln aus einfacheren Elementen aufgebaut werden können, ausgehend von einer endlichen Menge von Grundelementen.

Verfahren 9.6 - Induktive Definition von Mengen

Gegeben ist folgendes:

- Eine endliche Menge $G = \{a_1, \ldots, a_n\}$ von **Grundelementen.**
- Eine endliche Menge $R = \{r_1, \ldots, r_k\}$ von **Konstruktionsregeln**, die beschreiben, wie aus bereits vorhandenen Elementen der Menge neue Elemente der Menge aufgebaut werden können.

Die damit **induktiv definierte Menge** besteht aus allen Elementen, die durch endlich viele Konstruktionsschritte ausgehend von den Grundelementen konstruiert werden können

Soll nun eine Eigenschaft P(x) für alle Elemente x einer induktiv definierten Menge gezeigt werden, kann das folgende Verfahren der *strukturellen Induktion* verwendet werden.

Beweisverfahren 9.7 - Strukturellen Induktion

Eine Menge M sei induktiv durch Menge der Grundelemente G und Konstruktionsregeln R definiert. Um zu zeigen, dass eine Eigenschaft P(x) für alle Elemente $x \in M$ gilt, muss folgendes nachgewiesen werden:

(1) **Induktionsbasis**: P(a_i) gilt für jedes Grundelemente $a_i \in G$.

(2) **Induktionsschritte**: Für jede Konstruktionsregel $r_j \in R$ gilt:

Wenn

ein Element y mit Konstruktionsregel r_j aus den Elementen $x_1, \ldots, x_n$ gebildet werden kann

und

P(x_i) für alle x_i gilt (**Induktionshypothese**),

dann

gilt auch P(y) .

Beispiel 9.8 - Induktive Definition für arithmetische Ausdrücke

Arithmetische Ausdrücke aus Variablen, Zahlen, binären Operatoren und Klammern, wie z.B. `13 * (7 + var1 - 42) / var2`, seien durch folgende Regeln induktiv definiert:

Grundelemente (elementare arithmetische Ausdrücke):

(1) Jede *Zahl* ist ein arithmetischer Ausdruck.

(2) Jede *Variable* ist ein arithmetischer Ausdruck.

Konstruktionsregeln (zusammengesetzte arithmetische Ausdrücke):

(3) Sind E_1 und E_2 arithmetische Ausdrücke und ist *Op* ein binärer Operator (`+`, `-` , `*`, `/`), dann ist auch E_1 *Op* E_2 ein arithmetischer Ausdruck.

(4) Ist E ein arithmetischer Ausdruck, dann ist auch `(`E`)` ein arithmetischer Ausdruck, d.h. Klammern können um jeden arithmetischen Ausdruck gesetzt werden.

Beispiel 9.9 - Strukturelle Induktion für arithmetische Ausdrücke

Es soll nun folgende Eigenschaft für alle s.o. definierte arithmetische Ausdrücken gezeigt werden:

> Besteht ein arithmetischer Ausdruck aus n Zahlen und v Variablen, dann enthält er (n + v - 1) Operatoren.

Diese Eigenschaft kann mittels struktureller Induktion über den Aufbau der Ausdrücke bewiesen werden:

- **Induktionsanfang**: Betrachte elementare arithmetische Ausdrücke:

  (1) Eine *Zahl* ist ein elementarer Ausdruck:
  Ausdruck besteht aus n=1 Zahlen, v=0 Variablen, 0 Operatoren
  $n + v - 1 = 1 + 0 - 1 = 0$
  $\Rightarrow$ Eigenschaft ist hier erfüllt.

  (2) *Variable* als elementarer arithmetischer Ausdruck:
  Ausdruck besteht aus n=0 Zahlen, v=1 Variablen, 0 Operatoren
  $n + v - 1 = 0 + 1 - 1 = 0$
  $\Rightarrow$ Eigenschaft ist hier erfüllt.

- **Induktionsschritt(e)**: Betrachte zusammengesetzte arithmetische Ausdrücke.

  Als *Induktionshypothese* kann verwendet werden, dass die Eigenschaft für die Teilausdrücke E_1 und E_2 schon gilt, d.h.

- Besteht E_1 aus n_1 Zahlen und v_1 Variablen, dann enthält E_1 $n_1 + v_1$-1 Operatoren
- Besteht E_2 besteht aus n_2 Zahlen und v_2 Variablen, dann enthält E_2 $n_2 + v_2$-1 Operatoren

(3) Zusammengesetzter Ausdruck $\boldsymbol{E_1}$ **Op** $\boldsymbol{E_2}$:

Ausdruck E_1 *Op* E_2 enthält $n = n_1 + n_2$ Zahlen
Ausdruck E_1 *Op* E_2 enthält $v = v_1 + v_2$ Variablen
Ausdruck E_1 *Op* E_2 hat Operatoren aus E_1 + Operatoren aus E_2 + 1 Operator dazwischen, d.h. Anzahl der Operatoren ist

$$(n_1 + v_1 - 1) + 1 + (n_2 + v_2 - 1)$$
$$= (n_1 + n_2) + (v_1 + v_2) - 1 + 1 - 1$$
$$= n + v - 1$$

$\Rightarrow$ Eigenschaft gilt auch für E_1 *Op* E_2.

(4) Zusammengesetzter Ausdruck $\boldsymbol{(E_1)}$ mit Klammern:

Nach Induktionshypothese beseht E_1 aus n_1 Zahlen, v_1 Variablen und hat n_1+v_1 - 1 Operatoren.
Der geklammerte Ausdruck $\boldsymbol{(E_1)}$ enthält die gleiche Anzahl an Zahlen, Variablen und Operatoren.

$\Rightarrow$ Eigenschaft gilt für zusammengesetzten Ausdruck $\boldsymbol{(E_1)}$.

Induktionsschluss: Für jeden mit den angegebenen Regeln gebildeten arithmetischen Ausdruck E gilt: Wenn E aus n Zahlen und v Variablen besteht, dann enthält E insgesamt $n+v$-1 Operatoren.

Aufgabe 9.10 - Strukturelle Induktion

Eine Menge $M \subseteq \mathbb{N}\times\mathbb{N}$ von Paaren natürlicher Zahlen sei folgendermaßen induktiv definiert:

(1) $(1,1) \in M$

(2) $(3,9) \in M$

(3) Sind $(a,b) \in M$ und $(c,d) \in M$, dann ist auch $(a \cdot c, b \cdot d) \in M$

(4) Sind $(a,b) \in M$ und $(c,d) \in M$, dann ist auch $(a+c, b+d+2 \cdot a \cdot c) \in M$

Beweisen Sie durch Strukturelle Induktion, dass für alle Paare $(x,y) \in M$ gilt: $y = x^2$.

Anmerkung

- Die *vollständige Induktion* lässt sich als ein Spezialfall der strukturellen Induktion betrachten, wenn man folgende induktive Definition für die natürlichen Zahlen zugrunde legt:
  - Ein Grundelement: 0 ist eine natürliche Zahl

- Eine Konstruktionsregel: Ist *n* eine natürliche Zahl, dann ist auch der Nachfolger von *n* (d.h. n+1), eine natürliche Zahl.

- Umgekehrt können Beweise mittels *struktureller Induktion* auch auf Beweise mit vollständiger Induktion zurückgeführt werden, indem die Induktion über die Anzahl der Konstruktionsschritte geführt wird, die zur Erzeugung eines Elements nötig sind.

Beweis zu Eigenschaft 9.5 - Äquivalenz von regulärem Ausdruck und e-NEA

Mittels struktureller Induktion lässt sich nun auch beweisen, dass für jeden regulären Ausdruck der dazu konstruierte Automat die Sprache des regulären Ausdrucks akzeptiert.

[Sie müssen diesen Beweis in der Klausur nicht reproduzieren können. Sie sollten aber verstehen, wie strukturelle Induktion hier eingesetzt wird.]

Wir wollen zeigen, dass für jeden regulären Ausdruck *R* gilt:

$$L(R) = L(A_R),$$

wobei A_R der nach Algorithmus 9.1 zu *R* gebildete Automat ist.

Wir beweisen dies mittels struktureller Induktion über den Aufbau der regulären Ausdrücke.

- **Induktionsbasis** (elementare reguläre Ausdrücke):

  Fall $R = \varnothing$: Sprache des regulären Ausdrucks nach Definition 5.5 (Semantik regulärer Ausdrücke):

  $L(\varnothing) = \{\}$

  Sprache des Automaten:

  $L(A_\varnothing) = \{\}$ (nichts wird akzeptiert)

  somit $L(\varnothing) = \{\} = L(A_\varnothing)$

  Fall $R = \varepsilon$: Sprache des regulären Ausdrucks nach Definition 5.5:

  $L(\varepsilon) = \{ \varepsilon \}$

  Sprache des Automaten

  $L(A_\varepsilon) = \{ \varepsilon \}$ (nur leeres Wort wird akzeptiert)

  somit $L(\varepsilon) = L(A_\varepsilon)$

  Fall $R = \mathbf{a}$ (für $a \in \Sigma$): Sprache des regulären Ausdrucks nach Definition 5.5:

  $L(a) = \{ a \}$

  Sprache des Automaten

  $L(A_a) = \{ a \}$ (akzeptiert nur Wort a)

  somit $L(a) = L(A_a)$

- **Induktionsschritte** (für zusammengesetzte Ausdrücke). Als Induktionshypothese kann angenommen werden, dass die Eigenschaft für die Teilausdrücke S und T schon gilt, d.h. $L(S) = L(A_S)$ und $L(T) = L(A_T)$.

  Fall $R = \boldsymbol{S \cdot T}$:

  Sprache des regulären Ausdrucks nach Definition 5.5:

  $$L(S \cdot T) = L(S) \cdot L(T) = \{uv \mid u \in L(S), v \in L(T)\}$$

  Sprache des Automaten $A_{S \cdot T}$: Automat akzeptiert alle Wörter $w=uv$, so dass Anfangsstück u von A_S akzeptiert wird und Rest v von A_T, d.h.

  $$L(A_{S \cdot T}) = \{uv \mid u \in L(A_S), v \in L(A_T)\} = L(A_S) \cdot L(A_T)$$

  somit $L(S \cdot T) = L(S) \cdot L(T) = L(A_S) \cdot L(A_T) = L(A_{S \cdot T})$

  Fall $R = \boldsymbol{S \mid T}$:

  Sprache des regulären Ausdrucks nach Definition 5.5:

  $$L(S \mid T) = L(S) \cup L(T)$$

  Sprache des Automaten $A_{S|T}$: Automat akzeptiert sowohl alle Wörter, die A_S akzeptiert, als auch alle Wörter, die Teilautomat A_T akzeptiert, d.h.

  $$L(A_{S|T}) = L(A_S) \cup L(A_T)$$

  somit $L(S \mid T) = L(S) \cup L(T) = L(A_S) \cup L(A_T) = L(A_{S|T})$

  Fall $R = \boldsymbol{S^*}$:

  Sprache des regulären Ausdrucks nach Definition 5.5:

  $$L(S^*) = L(S)^*$$

  Automat A_{S^*} akzeptiert das leere Wort ε sowie alle Wörter $w = u_1 u_2 \ldots u_k$, für die gilt, dass jedes Teilwort u_i von Teilautomat A_S akzeptiert wird, d.h.

  $$L(A_{S^*}) = L(A_S)^*$$

  somit $L(S^*) = L(S)^* = L(A_S)^* = L(A_{S^*})$

  Fall $R = \boldsymbol{(S)}$:

  Sprache des regulären Ausdrucks nach Definition 5.5:

  $$L((S)) = L(S)$$

  Sprache des Automaten

  $$L(A_{(S)}) = L(A_S)$$ (Da Automat für (S) gleich ist wie für S)

  somit $L((S)) = L(A_{(S)})$

Induktionsschluss: Für jeden regulären Ausdruck R gilt $L(R) = L(A_R)$, d.h. der konstruierte Automat akzeptiert genau die Sprache des regulären Ausdrucks. □

Als simple Folgerung ergibt sich nun, dass zu einem regulären Ausdruck nicht nur ein ε-NEA, sondern auch ein DEA gebildet werden kann (der dann auch effizient implementiert werden kann).

Eigenschaft 9.11 - Konvertierung regulärer Ausdrücke in DEAs

Zu jedem *regulären Ausdruck R* gibt es einen *DEA* A_R, der die Sprache des Ausdrucks akzeptiert, d.h. $L(R) = L(A_R)$.

Beweis

Zu jedem regulären Ausdruck kann mit Verfahren 9.1 ein ε-NEA konstruiert werden, der die Sprache des regulären Ausdrucks *R* akzeptiert.

Dieser Automat kann mit der Teilmengenkonstruktion 8.13 oder 8.15 in einen äquivalenten DEA umgewandelt werden, der die gleiche Sprache akzeptiert.

Zusammenfassung zu Lektion 14

Diese Fragen sollten Sie nun beantworten können

- Wie können ε-NEAs praktisch eingesetzt werden?
- Wie kann zu einem regulären Ausdruck eine äquivalenter ε-NEA bzw. DEA konstruiert werden?
- Was ist strukturelle Induktion? Wozu wird sie eingesetzt?