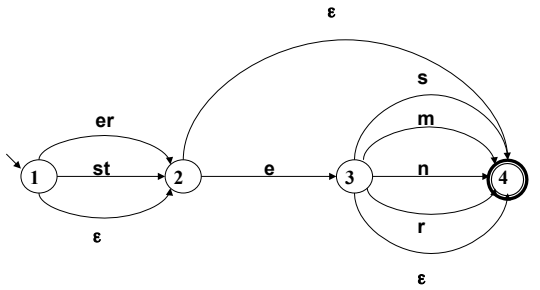
Einführung in die Computerlinguistik

Morphologie und Automaten II

WS 2012/2013

Manfred Pinkal

Adjektivendungen: Zustandsdiagramm



Systematische Pfadsuche

- Der NEA erlaubt es, bei derselben Eingabe unterschiedliche Kanten zu beschreiten. Um sicher zu sein, dass kein möglicher Weg durch den Automaten übersehen wurde, müssen wir ein Verfahren (einen Algorithmus) spezifizieren, der vollständige Suche gewährleistet.
- Die Idee: Wenn wir für den aktuellen Zustand und die aktuelle Position in der Eingabekette die möglichen Übergänge identifiziert haben, rücken wir nicht unmittelbar vor, sondern legen die alternativ erreichbaren Konfigurationen – Zustand und Position in w – in einer Agenda ab.
- Wir nehmen die erreichbaren Konfigurationen (Informationen über erreichbare Zustand-Positionspaare) nach und nach von der Agenda und testen dadurch alle Möglichkeiten systematisch aus.

Ein Verfahren für die Pfadsuche

- Für Eingabewort w = x1 ... xn: Initialisiere die Agenda mit <s, x1 ... xn>.
- Solange die Agenda Einträge enthält:
  1. Nimm den obersten Eintrag von der Agenda, setze den aktuellen Zustand und die aktuelle Eingabeposition auf die dort gespeicherten Werte.
  2. Wenn der aktuelle Zustand Endzustand ist und die Positionsmarkierung am Ende des Eingabewortes steht, akzeptiere w;
     andernfalls weiter mit 3.
  3. Lege alle möglichen Zielkonfigurationen, die sich vom aktuellen Zustand aus erreichen lassen, als neue Aufgaben auf der Agenda ab. Weiter mit 1.
- Wenn die Agenda leer ist, weise w zurück.

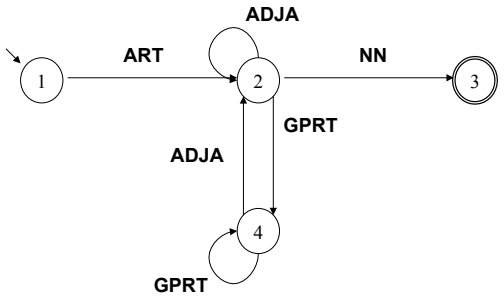
Anmerkungen zum Pfadsuche-Algorithmus

- Wir haben die Agenda als Stapel („Stack“) konzipiert: Wir legen neue Aufgaben oben auf der Agenda ab, und nehmen einzelne Aufgaben ebenfalls von oben von der Agenda ("Last In – First Out").
- Die Last In – First Out-Regel führt dazu, dass ein einmal gewählter Pfad so weit wie möglich weiter verfolgt wird. Wenn die Suche in eine Sackgasse gerät, werden aktueller Zustand und Eingabeposition auf in der Agenda zuvor gespeicherte Werte zurückgesetzt. Der Pfadsuche-Algorithmus ist ein Beispiel für „Tiefensuche mit Backtracking“.

Anmerkungen zum Pfadsuche-Algorithmus

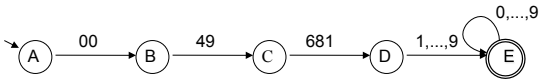
- Der Algorithmus ist nicht vollständig: Wenn der Automat Leerwort-Zyklen enthält (bei deren Durchlaufen keine Eingabe abgearbeitet wird), kann er in eine Endlosschleife geraten.
- Der Algorithmus ist ineffizient: Bei einem maximalen Verzweigungsfaktor n des Automaten benötigt der Algorithmus zur vollständigen Abarbeitung eines Eingabewortes w im schlechtesten Fall $n^{|w|}$ Schritte. Der Zeitbedarf wächst exponentiell mit der Wortlänge.
- Man kann die Situation verbessern, indem man
  - die Information im Diagramm geschickt formuliert und anordnet
  - den Algorithmus optimiert (z.B. durch Testen, ob eine neu generierte Konfiguration schon einmal vorgekommen ist)
  - eine alternative Repräsentation der linguistischen Information wählt, die effizientere Verarbeitung erlaubt

Ein deterministisches Diagramm



Beobachtung: Bestimmte Diagramme erfordern keine Suche, weil Übergänge bei gegebenem Zustand und Eingabesymbol eindeutig festgelegt sind.

Ein anderes deterministisches Diagramm



Deterministische endliche Automaten

- Die beiden Diagramme unterscheiden sich von dem Adjektiv-Diagramm in einem wesentlichen Punkt: Für jeden Zustand/Knoten und jede Eingabe gibt es höchstens eine Kante, die beschritten werden kann. Sie sind deterministisch.
- Die Definition des „deterministischen endlichen Automaten“ (DEA oder DFA, für „deterministic finite-state automaton“) führt einige weitere, weniger wesentliche, aber nützliche Beschränkungen gegenüber dem NEA ein.

Deterministische und nicht-deterministische Automaten

- NEA erlaubt beliebige Worte (incl. ε) als Kanteninschrift
- NEA erlaubt für einen Ausgangszustand und eine Eingabe mehrere oder gar keinen Zielzustand
- D.h.: NEA hat eine Übergangsrelation.

- DEA hat nur Einzelsymbole als Kanten-Inschriften, insbesondere sind Leerwort-Kanten nicht zulässig.
- DEA hat zu jedem Zustand und zu jedem Symbol genau eine wegführende Kante
- D.h.: DEA hat eine Übergangsfunktion.

Definition NEA

Ein NEA ist ein Quintupel

A = <K, Σ, Δ, s, F>, wobei

- K nicht-leere endliche Menge von Knoten (Zuständen)
- Σ nicht-leeres Alphabet
- $s \in K$ Startzustand
- $F \subseteq K$ Menge von Endzuständen
- $\Delta : K \times \Sigma^* \times K$ Menge von beschrifteten Kanten (Übergangsrelation)

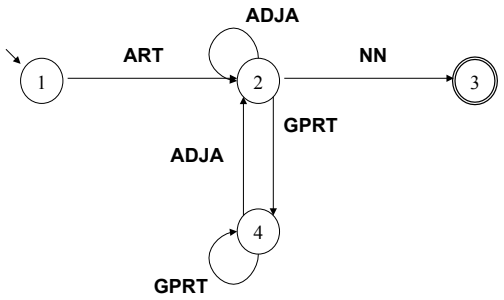
Definition: Deterministischer Endlicher Automat

Ein DEA ist ein Quintupel

A = <K, Σ, δ, s, F>, wobei

- K nicht-leere endliche Menge von Knoten (Zuständen)
- Σ nicht-leeres Alphabet
- $s \in K$ Startzustand
- $F \subseteq K$ Menge von Endzuständen
- $\delta : K \times \Sigma \rightarrow K$ Übergangsfunktion

Beispiel: DEA für Wortartmuster



Beispiel: DEA für Wortartmuster

DEA A = <K, Σ, δ, s, F> mit

- K= {1,2,3,4}
- Σ = {ART, ADJA, NN, GPRT}
- s = 1
- F = {3}
- δ definiert durch: δ(1,ART) = 2
  δ(2, ADJA) = 2
  δ(2,NN) = 3
  δ (2,GPRT) = 4
  ... ...

Beispiel: Übergangstabelle für δ

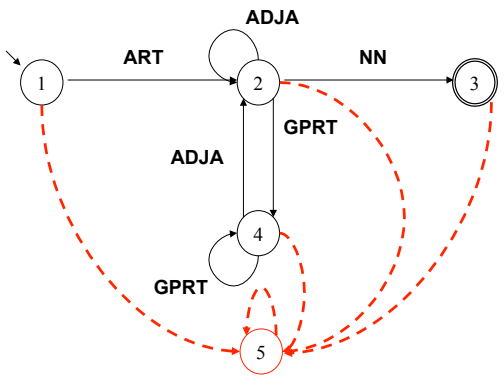
δ:	ART	ADJA	NN	GPRT
1	2			
2		2	3	4
3				
4		2		4

Beispiel: Übergangstabelle für δ, komplettiert

δ:	ART	ADJA	NN	GPRT
1	2	5	5	5
2	5	2	3	4
3	5	5	5	5
4	5	2	5	4
5	5	5	5	5

- Der Zustand eines DEA, aus dem es keine Möglichkeit gibt, in einen Endzustand zu gelangen, heißt „Senke" oder engl. „trap state": Falle.

Das Zustandsdiagramm für Wortartmuster: Übergangsfunktion δ komplettiert



Deterministische und nicht-deterministische Automaten [1]

- DEAs erlauben den Test von Eingabeketten in linearer Zeit: Jedes Wort der Länge n wird in genau n Schritten abgearbeitet.
- DEAs haben allerdings ein eingeschränkteres Beschreibungs-Inventar als NEAs.
- Frage: Ist deshalb die Ausdrucksstärke des DEA-Formalismus eingeschränkter als die von NEAs? Das heißt, gibt es Sprachen, die durch einen NEA, aber nicht durch einen DEA beschrieben werden?
- Die Antwort lautet: Nein!

Deterministische und nicht-deterministische Automaten [2]

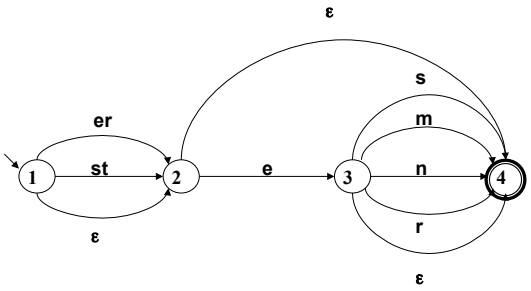
- Jede Sprache, die von einem NEA akzeptiert wird, kann auch durch einen DEA beschrieben werden (und, trivialerweise, auch umgekehrt: ein DEA ist ein spezieller NEA). NEAs und DEAs besitzen die gleiche Ausdrucksstärke, die Formalismen sind beschreibungsäquivalent.
- Das ist beweisbar. Noch wichtiger: Der Beweis ist konstruktiv, d.h.:
- Er basiert auf einem Konstruktionsverfahren, das es erlaubt, zu jedem NEA A einen DEA A' zu konstruieren, so dass L(A') = L(A).

Die NEA-DEA-Überführung

Der Algorithmus zur NEA-DEA-Überführung besteht aus drei Schritten:

1. Beseitigung von Mehrsymbol-Kanten
2. Beseitigung von ε-Kanten
3. Die „Potenz-Automaten"-Konstruktion

Adjektivendungen: Zustandsdiagramm



Schritt 1: Beseitigung von Mehrsymbolkanten

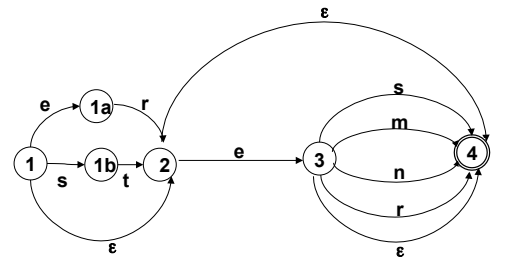
Gegeben sei der NEA A = <K, Σ, Δ, s, F>.

- Für alle Kanten <q,w,q'> mit w = $a_1 \ldots a_n$, n >1:

  Entferne <q,w,q'> aus Δ.
- Erweitere K um neue Zustände $q_1, \ldots, q_{n-1}$.
- Erweitere Δ um neue Kanten

  $<q, a_1, q_1>, <q_1, a_2, q_2>, \ldots, <q_{n-1}, a_n, q'>$

Beispiel-Automat nach Schritt 1:



Die NEA-DEA-Überführung

Der Algorithmus zur NEA-DEA-Überführung besteht aus drei Schritten:

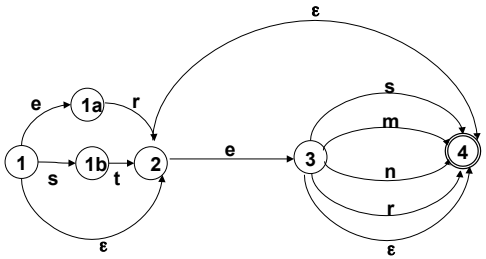
1. Beseitigung von Mehrsymbol-Kanten
2. Beseitigung von ε-Kanten
3. Die „Potenz-Automaten"-Konstruktion

Schritt 2: Beseitigung von ε-kanten

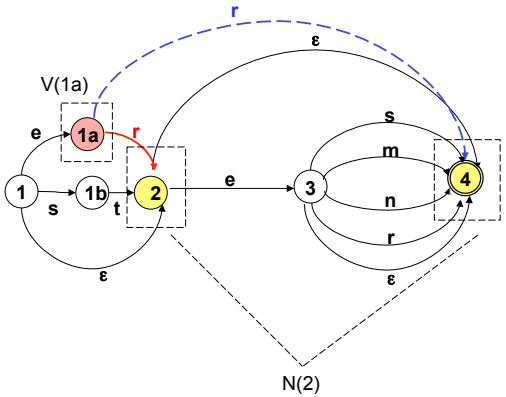
- Wir definieren zunächst als Hilfsbegriffe den „ε-Vorbereich" $V_\varepsilon(p)$ und den „ε-Nachbereich" $N_\varepsilon(p)$ von Zuständen:
  - $V_\varepsilon(p) = \{q|$ p ist von q aus ohne Abarbeiten eines Symbols erreichbar$\}$
  - $N_\varepsilon(p) = \{q|$ q ist von p aus ohne Abarbeiten eines Symbols erreichbar$\}$

  Anmerkung: $V_\varepsilon(p)$ und $N_\varepsilon(p)$ enthalten insbesondere p selbst.
- Für jede nicht-leere Kante $<p, a, q> \in \Delta$: Erweitere Δ um alle $<p', a, q'>$ mit $p' \in V_\varepsilon(p)$, $q' \in N_\varepsilon(q)$.
- Entferne alle leeren Kanten aus Δ.

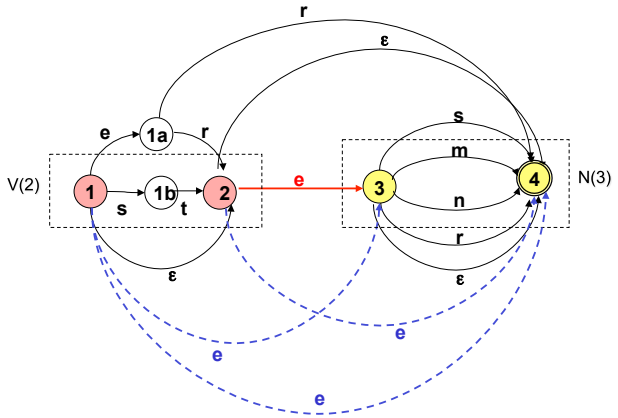
Beispiel-Automat nach Schritt 1:



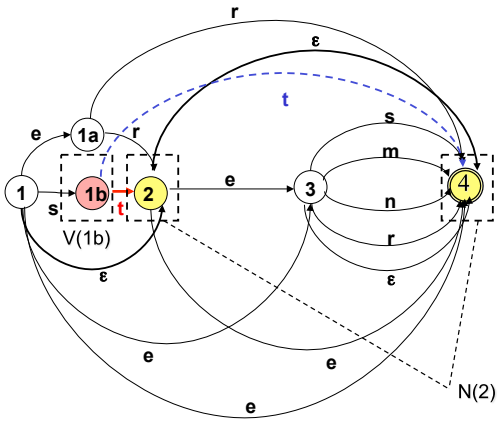
Schritt 2: Beseitigung von ε-Kanten



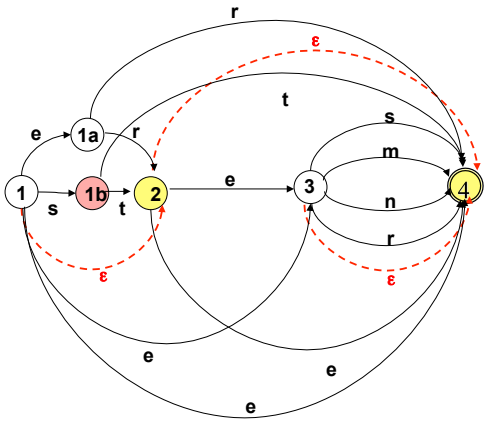
Schritt 2: Beseitigung von ε-kanten



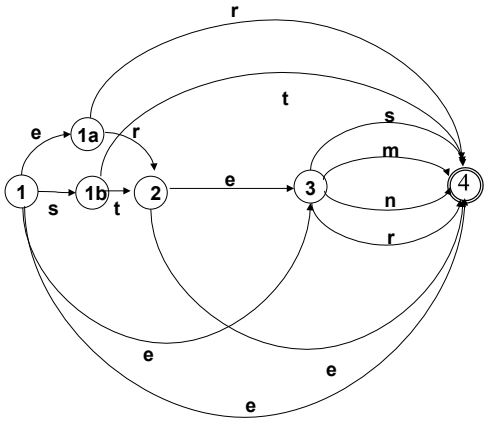
Schritt 2: Beseitigung von ε-kanten



Schritt 2: Beseitigung von ε-kanten



Schritt 2: Beseitigung von ε-kanten: Resultat ist „buchstabierender Automat"

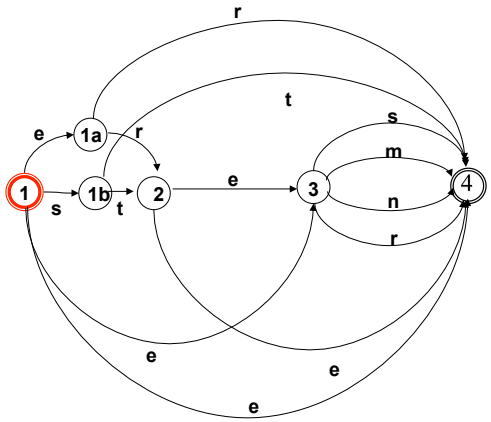


Schritt 2: Beseitigung von ε-kanten

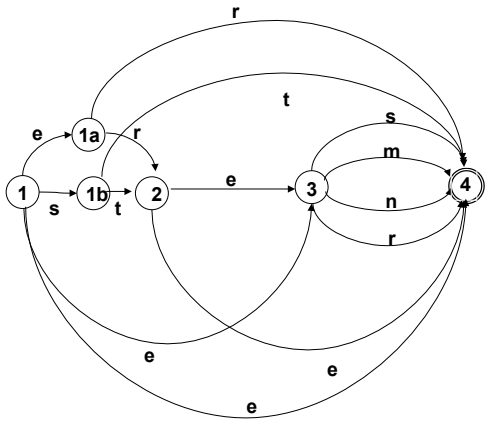
- Wir definieren zunächst als Hilfsbegriffe den „ε-Vorbereich" $V_\varepsilon(p)$ und den „ε-Nachbereich" $N_\varepsilon(p)$ von Zuständen:
  - $V_\varepsilon(p) = \{q|$ p ist von q aus ohne Abarbeiten eines Symbols erreichbar$\}$
  - $N_\varepsilon(p) = \{q|$ q ist von p aus ohne Abarbeiten eines Symbols erreichbar$\}$

  Anmerkung: $V_\varepsilon(p)$ und $N_\varepsilon(p)$ enthalten insbesondere p selbst.
- Für jede nicht-leere Kante $<p, a, q> \in \Delta$: Erweitere Δ um alle $<p', a, q'>$ mit $p' \in V_\varepsilon(p)$, $q' \in N_\varepsilon(q)$.
- Entferne alle leeren Kanten aus Δ.
- Wenn sich ein Endzustand im ε-Nachbereich des Startzustandes s befindet, füge s zu den Endzuständen hinzu.

Schritt 2: Beseitigung von ε-kanten: Resultat ist „buchstabierender Automat“



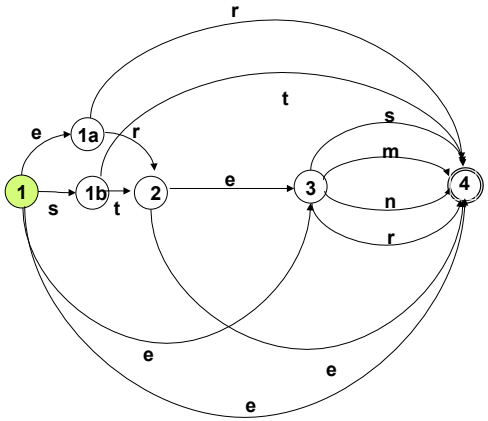
Pfadsuche als Breitensuche



Eingabewort:

Agenda: 1 -- klein eres

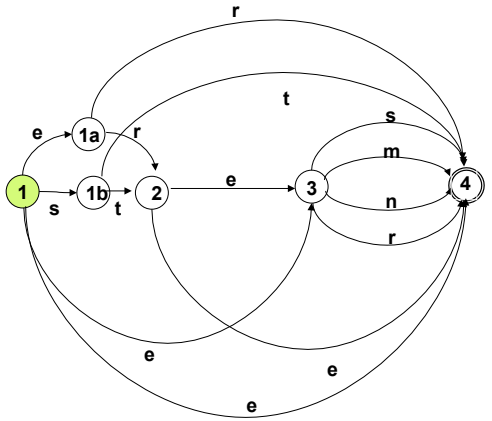
Pfadsuche als Breitensuche



Eingabewort: klein eres

Agenda:

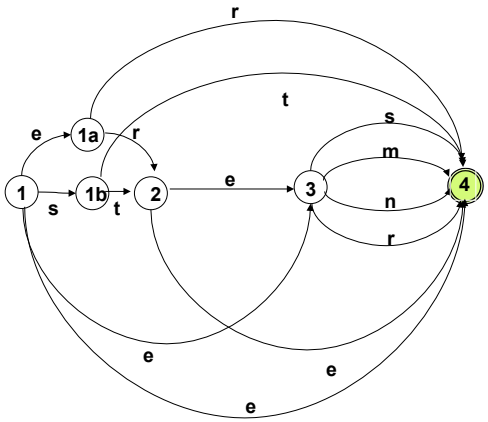
Pfadsuche als Breitensuche



Eingabewort: klein eres

Agenda: 1a -- klein eres
3 -- klein eres
4 -- klein eres

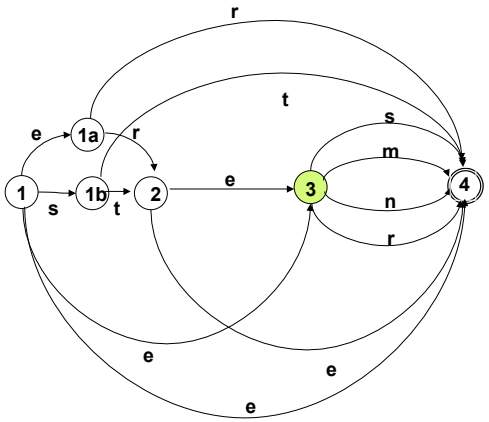
Pfadsuche als Breitensuche



Eingabewort: klein eres

Agenda: 1a -- klein eres
3 -- klein eres

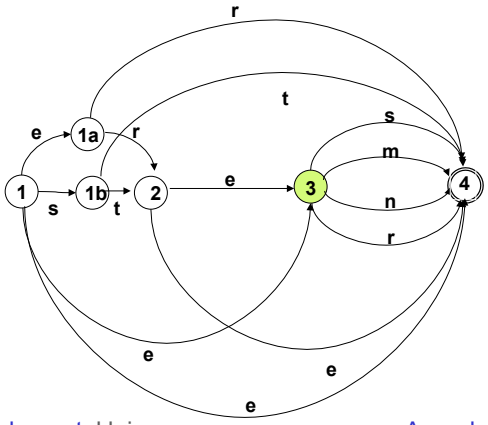
Pfadsuche als Breitensuche



Eingabewort: klein eres

Agenda: 1a -- klein eres

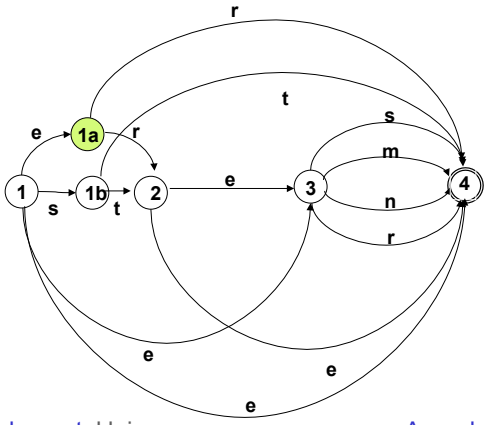
Pfadsuche als Breitensuche



Eingabewort: klein eres

Agenda: 4 -- klein eres
1a -- klein eres

Pfadsuche als Breitensuche



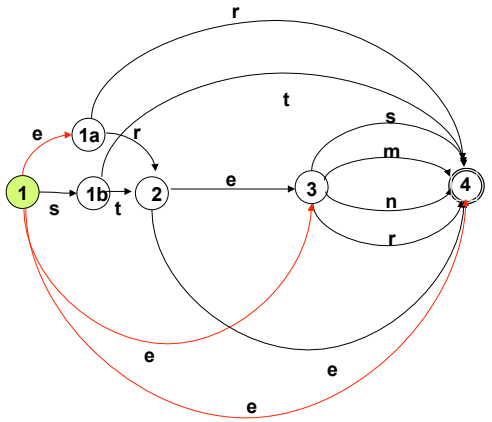
Eingabewort: klein eres

Agenda: 2 -- klein eres
4 -- klein eres
4 -- klein eres

Schritt 3: Potenzautomaten-Konstruktion, Vorüberlegung

- Wir haben einen Algorithmus zur Pfadsuche am Beispiel des unbearbeiteten Adjektivendungs-Diagramms kennengelernt: „Tiefensuche mit Backtracking“. Durch die Organisation der Agenda als Stapel/Stack („last in – first out“) wird eine Alternative so weit wie möglich verfolgt; bei endgültigem Scheitern wird das System zurückgesetzt.
- Durch die Organisation der Agenda als Warteschlange (queue), bei der die Aufgaben in der Reihenfolge ihrer Generierung abgearbeitet werden („first in – first out“), erhalten wir Breitensuche. Die alternativen Pfade werden (quasi) parallel verfolgt.

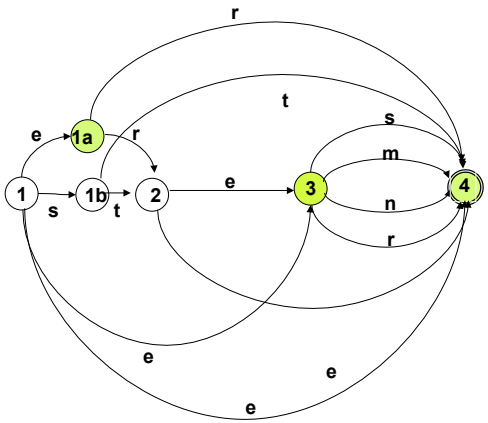
Pfadsuche als Breitensuche



Eingabewort: klein eres

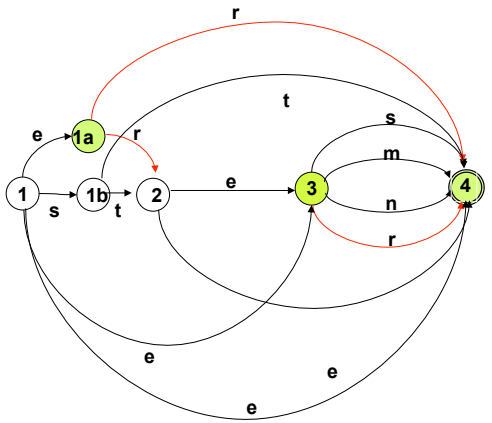
Agenda:
1a -- klein eres
3 -- klein eres
4 -- klein eres

Pfadsuche als Breitensuche



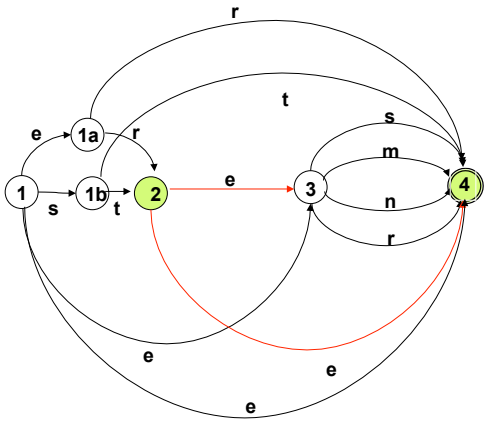
Eingabewort: klein eres

Pfadsuche als Breitensuche



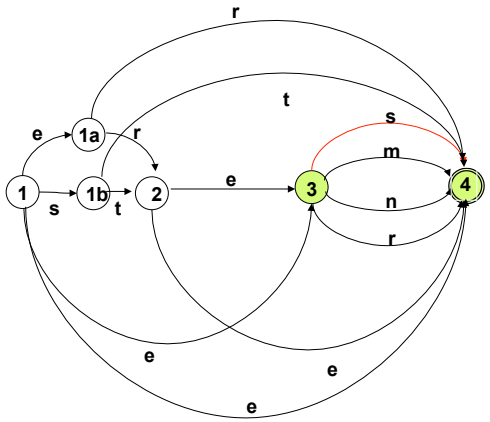
Eingabewort: klein eres

Pfadsuche als Breitensuche



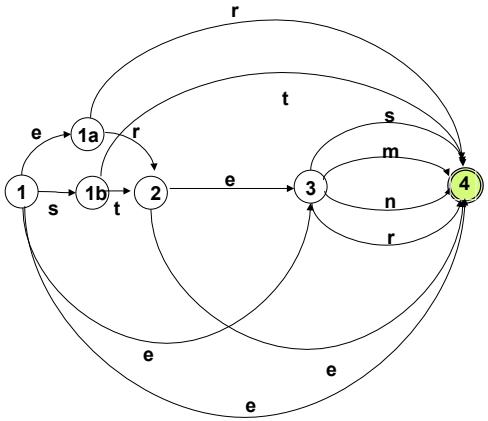
Eingabewort: klein eres

Pfadsuche als Breitensuche



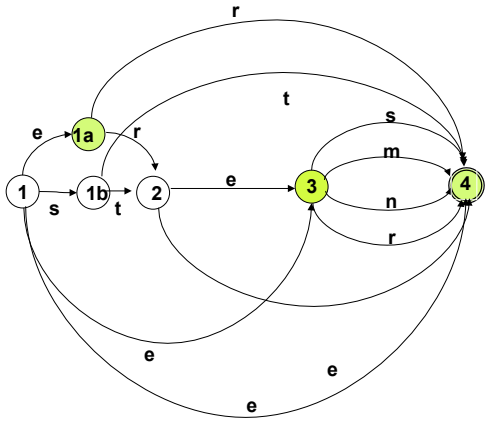
Eingabewort: klein eres

Pfadsuche als Breitensuche



Eingabewort: klein eres_

Pfadsuche als Breitensuche



Eingabewort: klein e

Schritt 3: Potenzautomaten-Konstruktion, Vorüberlegung [2]

- Wir können „getaktete“ Breitensuche in einem buchstabierenden NEA so beschreiben:
  - Wir ermitteln alle Zustände, die durch die Abarbeitung des ersten Eingabesymbols vom Startzustand aus erreicht werden können.
  - Wir ermitteln alle Zustände, die durch die Abarbeitung des zweiten Eingabesymbols von einem Zustand dieser Zustandsmenge erreicht werden können, usf.
  - Wenn die Zustandsmenge, die wir auf diese Weise nach Abarbeiten des kompletten Wortes w enthalten, einen Endzustand des NEA enthält, wird w akzeptiert.

Schritt 3: Potenzautomaten-Konstruktion, Vorüberlegung [3]

- Wir können diese "getaktete Suche" selbst mit einem endlichen Automaten beschreiben:
  - Zustände des neuen Automaten lassen sich als Mengen von Zuständen des NEA beschreiben. Am Beispiel: Nach Abarbeiten des ersten Symbols „e“ befindet er sich in dem Zustand, dass es die Zustandsmenge des NEA {1a, 2, 4} als mögliche aktuelle Zustände erkannt hat.
  - Wenn die Eingabekette abgearbeitet ist, und der Automat sich in einem Zustand befindet, der einen Endzustand des NEA enthält, ist die Eingabe akzeptiert.
  - Die „möglichen Zustände“ des NEA, die sich durch ein bestimmtes Eingabe-Symbol erreichen lassen, sind eindeutig definiert. Der neue Automat ist also ein DEA.

Schritt 3: Potenzautomaten-Konstruktion: Die Definition

Der Potenzautomat zum buchstabierenden NEA
$A = \langle K, \Sigma, \Delta, s, F\rangle$ ist der DEA A':

$A' = \langle K', \Sigma, \delta, s', F'\rangle$ mit:

- $K' = \wp(K)$ (die Potenzmenge der Zustandsmenge des NEA)
- $s' = \{s\}$
- $\delta(p', a) = \{q \mid$ es gibt $p \in p'$ und $\langle p,a,q\rangle \in D\}$ für jedes $p' \subseteq K$, $a \in S$
- $q' \in F'$ gdw. $q' \cap F \neq \emptyset$

Beispiel: DEA für Adjektiv-Endungen

- Grundlage: der buchstabierende Automat
  $A = \langle\{1,1a,1b,2,3,4\}, \{e,m,n,r,s,t\}, \Delta, 1, \{1,4\}\rangle$,
  Δ wie im Diagramm

- Potenzautomat ist $A' = \langle K', \Sigma, \delta, s', F'\rangle$
  mit $K' = \wp(K)$
  $s' = \{s\}$
  $F' = \{q' \in K' \mid 1 \in q' \text{ oder } 4 \in q'\}$
  δ s. Übergangstabelle nächste Folie

Praktisches Vorgehen

Der Potenzautomat A' zu A = <K, Σ, Δ, s, F> hat $2^{|K|}$ Zustände. In der Regel sind viele dieser Zustände unerreichbar (vom Startzustand {s} aus) und deshalb funktionslos.

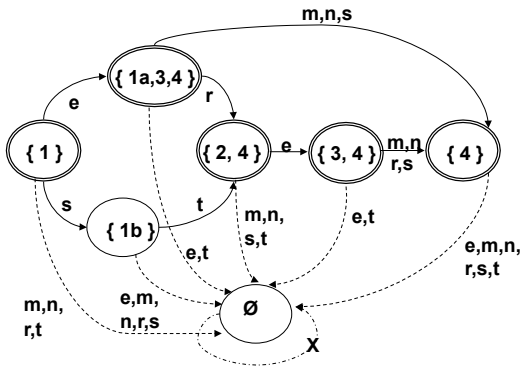
Praktisches Konstruktionsverfahren:

Beginne mit {s}, berechne die Übergangsfunktion für {s}, für alle direkt von s erreichbaren Zustände usw., bis keine neuen erreichbaren Zustände hinzukommen.

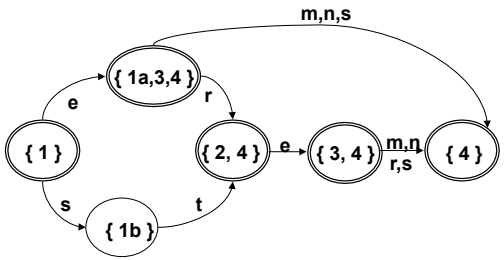
DEA für Adjektiv-Endungen, Übergangstabelle

δ:	e	m	n	r	s	t
{1}	{1a,3,4}	∅	∅	∅	{1b}	∅
{1a,3,4}	∅	{4}	{4}	{2,4}	{4}	∅
{1b}	∅	∅	∅	∅	∅	{2,4}
{2,4}	{3,4}	∅	∅	∅	∅	∅
{3,4}	∅	{4}	{4}	{4}	{4}	∅
{4}	∅	∅	∅	∅	∅	∅
∅	∅	∅	∅	∅	∅	∅

Das Diagramm



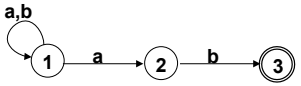
Das Diagramm, vereinfacht



Potenzautomatenkonstruktion, ein weiteres Beispiel

NEA A= < {1,2,3}, {a,b}, Δ,1, {3} >

Δ gegeben durch:



DEA

A‘ = < $\wp$({1,2,3}), {a,b}, δ, {1}, F‘>)

F´= {{3}, {1,3}, {2,3}, {1,2,3}}

Potenzautomatenkonstruktion, Beispiel 2: Die Übergangstabelle

q	δ(q, a)	δ(q, b)
{1}	{1,2}	{1}
{2}	Ø	{3}
{3}	Ø	Ø
{1,2}	{1,2}	{1,3}
{1,3}	{1,2}	{1}
{2,3}	Ø	{3}
{1,2,3}	{1,2}	{1,3}
Ø	Ø	Ø

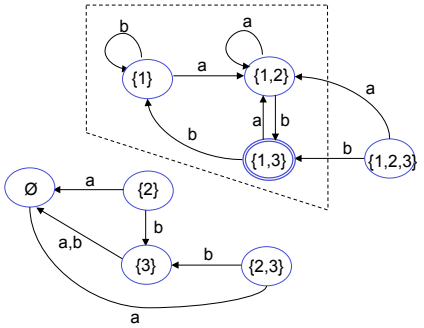
Potenzautomatenkonstruktion, Beispiel 2: Die Übergangstabelle

q	δ(q, a)	δ(q, b)
{1}	{1,2}	{1}
{2}	Ø	{3}
{3}	Ø	Ø
{1,2}	{1,2}	{1,3}
{1,3}	{1,2}	{1}
{2,3}	Ø	{3}
{1,2,3}	{1,2}	{1,3}
Ø	Ø	Ø

Potenzautomatenkonstruktion, Beispiel 2: Das Zustandsdiagramm

Nur ein Teil der Zustände ist vom Startzustand aus erreichbar.

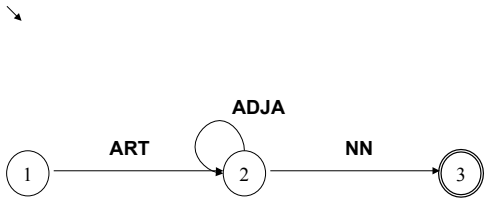
Die übrigen Zustände sind funktionslos.



Ausblick: Reguläre Ausdrücke

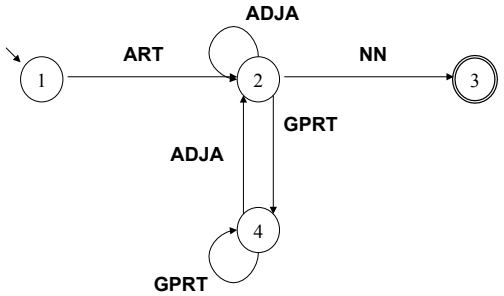
- NEA und DEA sind Formalismen mit äquivalenter Kernfunktionalität, aber unterschiedlichen praktischen Vorzügen.
- Es gibt einen dritten Formalismus, der mit NEA und DEA äquivalent ist: Reguläre Ausdrücke.
- Ein regulärer Ausdruck ist eine Zeichenkette, der eine Sprache beschreibt. Die Notation ist zum Teil weniger intuitiv als die Zustandsdiagramme, sie erlaubt aber die Spezifikation von Sprachen/ Wortmengen in Form eines kompakten Ausdrucks.

Regulärer Ausdruck: Beispiel



ART o ADJA* o NN

Ein deterministisches Diagramm



ART o (ADJA | (GPRT o GPRT*o ADJA))* o NN

Reguläre Ausdrücke: Formale Interpretation

- Reguläre Ausdrücke beschreiben Sprachen/ Mengen von Zeichenketten über einem gegebenen Alphabet.
- Die vom regulären Ausdruck Φ über dem Alphabet Σ beschriebene Sprache, $L(\Phi)$, wird in der folgenden Weise rekursiv definiert:
  - $L(a) = \{a\}$ für $a \in \Sigma$
  - $L(\alpha \mid \beta) = L(\alpha) \cup L(\beta)$
  - $L(\alpha \circ \beta) = L(\alpha\ \beta) = \{ww' \mid w \in L(\alpha) \text{ und } w' \in L(\beta)\}$
  - $L(\alpha^*) = \{w_1 \ldots w_n \mid w_1, \ldots, w_n \in L(\alpha), n \geq 0\}$
  - $L(\varnothing) = \varnothing$