

Einführung in die Computerlinguistik

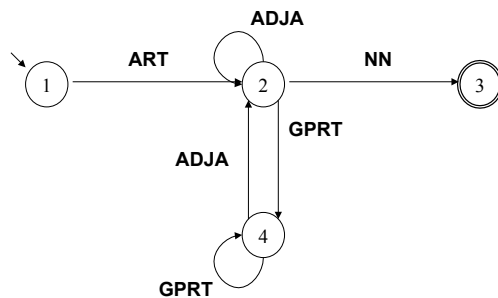
Morphologie und Automaten II

WS 2010/2011

Manfred Pinkal

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

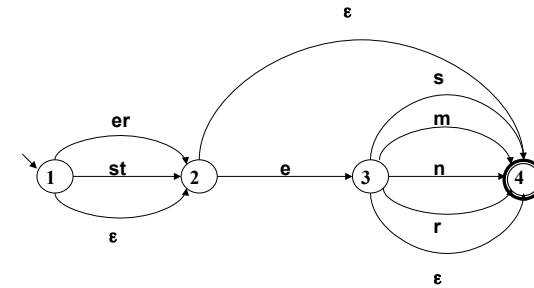
Ein deterministisches Diagramm



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

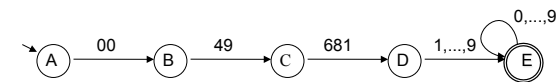
Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Adjektivendungen: Zustandsdiagramm



Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Saarbrücker Telefonnummern (international)



Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

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 beschriftet 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.

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Definition NEA

Formal wird ein Zustandsdiagramm definiert als ein Quintupel (Folge von 5 Elementen)

$A = \langle K, \Sigma, \Delta, s, F \rangle$, 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)

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

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**.

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Definition: Deterministischer endlicher Automat

Ein deterministischer endlicher Automat ist ein Quintupel

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

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Beispiel: Der DEA für Wortartmuster [1]

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

- $K = \{1, 2, 3, 4\}$
- $\Sigma = \{\text{ART}, \text{ADJA}, \text{NN}, \text{GPRT}\}$
- $s = 1$
- $F = \{3\}$
- δ definiert durch:
 - $\delta(1, \text{ART}) = 2$
 - $\delta(2, \text{ADJA}) = 2$
 - $\delta(2, \text{NN}) = 3$
 - $\delta(2, \text{GPRT}) = 4$
 -

Beispiel [2]: Übergangstabelle für δ

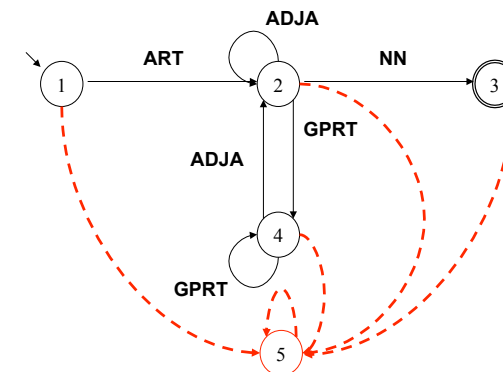
δ :	ART	ADJA	NN	GPRT
1	2			
2		2	3	4
3				
4		2		4

Beispiel [3]: 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!**

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UoS Computerlinguistik

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 $L(A') = L(A)$.

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UoS Computerlinguistik

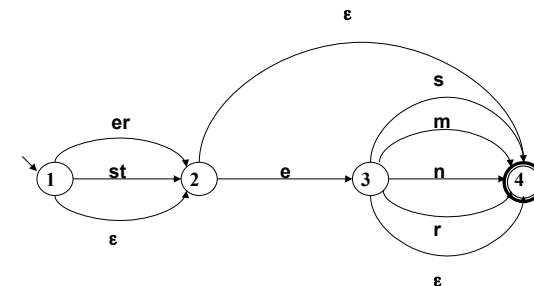
Die NEA-DEA-Überführung

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

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UoS Computerlinguistik

Adjektivendungen: Zustandsdiagramm



Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UoS Computerlinguistik

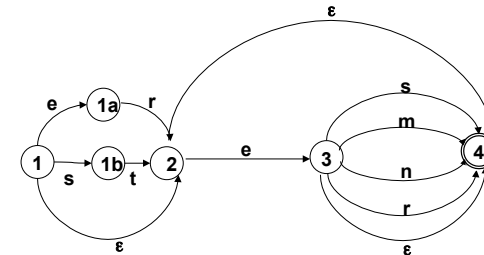
Schritt 1: Beseitigung von Mehrsymbolkanten

Gegeben sei der NEA $A = \langle K, \Sigma, \Delta, s, F \rangle$.

- Für alle Kanten $\langle q, w, q' \rangle$ mit $w = a_1 \dots a_n$, $n > 1$:
Entferne $\langle q, w, q' \rangle$ aus Δ .
- Erweitere K um neue Zustände q_1, \dots, q_{n-1} .
- Erweitere Δ um neue Kanten
 $\langle q, a_1, q_1 \rangle, \langle q_1, a_2, q_2 \rangle, \dots, \langle q_{n-1}, a_n, q' \rangle$

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Beispiel-Automat nach Schritt 1:



Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

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

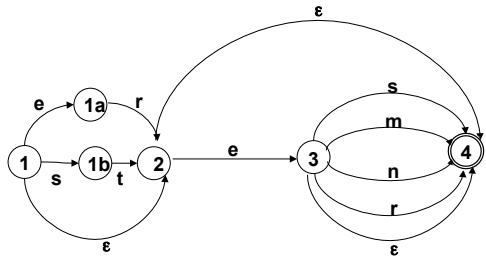
Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Schritt 2: Beseitigung von ϵ -kanten

- Wir definieren zunächst als Hilfsbegriffe den „ ϵ -Vorbereich“ $V_\epsilon(p)$ und den „ ϵ -Nachbereich“ $N_\epsilon(p)$ von Zuständen:
 - $V_\epsilon(p) = \{q \mid p \text{ ist von } q \text{ aus ohne Abarbeiten eines Symbols erreichbar}\}$
 - $N_\epsilon(p) = \{q \mid q \text{ ist von } p \text{ aus ohne Abarbeiten eines Symbols erreichbar}\}$
- Anmerkung: $V_\epsilon(p)$ und $N_\epsilon(p)$ enthalten insbesondere p selbst.
- Für jede nicht-leere Kante $\langle p, a, q \rangle \in \Delta$: Erweitere Δ um alle $\langle p', a, q' \rangle$ mit $p' \in V_\epsilon(p)$, $q' \in N_\epsilon(q)$.
- Entferne alle leeren Kanten aus Δ .

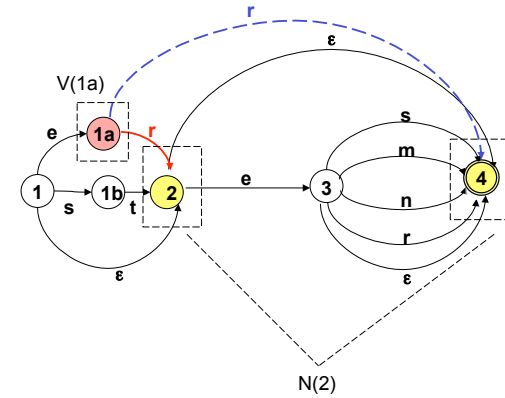
Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Beispiel-Automat nach Schritt 1:



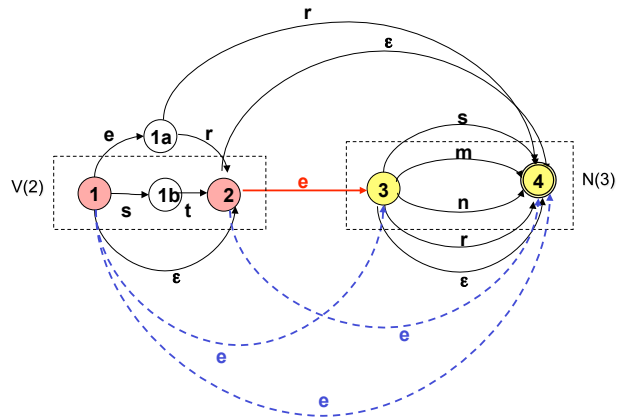
Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Schritt 2: Beseitigung von ε-Kanten



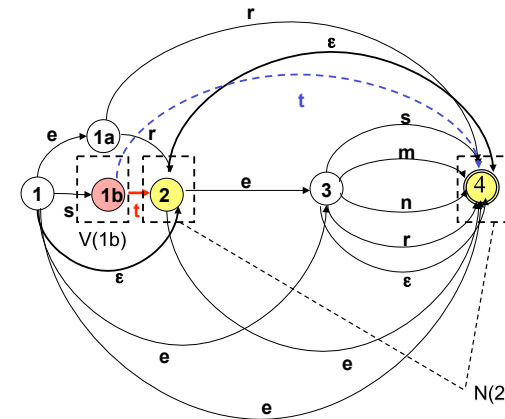
Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Schritt 2: Beseitigung von ε-kanten



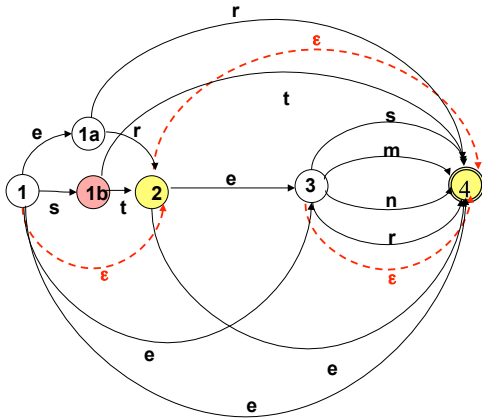
Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Schritt 2: Beseitigung von ε-kanten



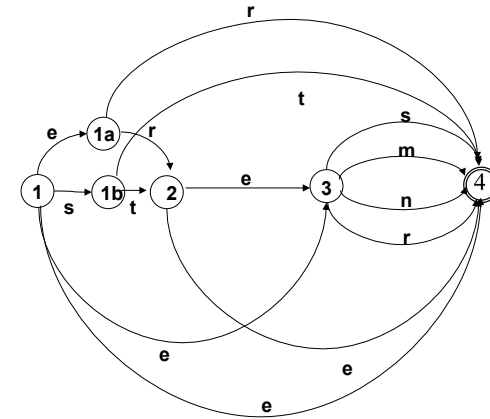
Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Schritt 2: Beseitigung von ϵ -kanten



Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

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



Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Schritt 2: Beseitigung von ϵ -kanten

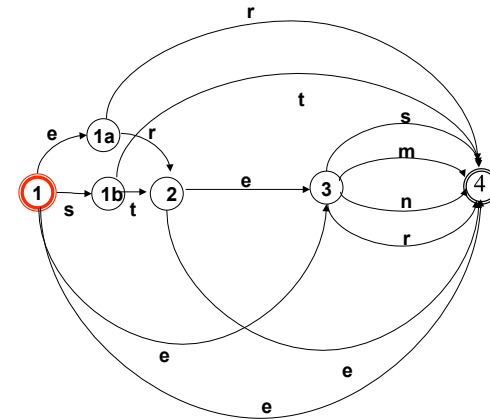
- Wir definieren zunächst als Hilfsbegriffe den „ ϵ -Vorbereich“ $V_\epsilon(p)$ und den „ ϵ -Nachbereich“ $N_\epsilon(p)$ von Zuständen:
 - $V_\epsilon(p) = \{q \mid p \text{ ist von } q \text{ aus ohne Abarbeiten eines Symbols erreichbar}\}$
 - $N_\epsilon(p) = \{q \mid q \text{ ist von } p \text{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

Anmerkung: $V_\epsilon(p)$ und $N_\epsilon(p)$ enthalten insbesondere p selbst.

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

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

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

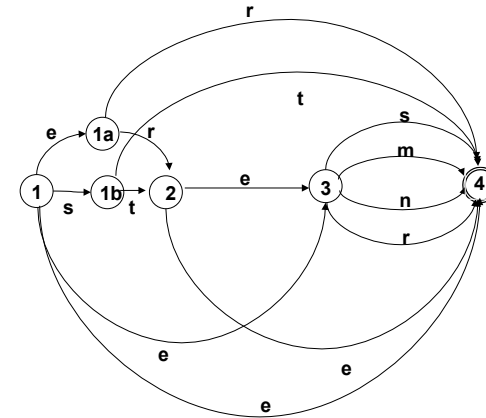


Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

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.

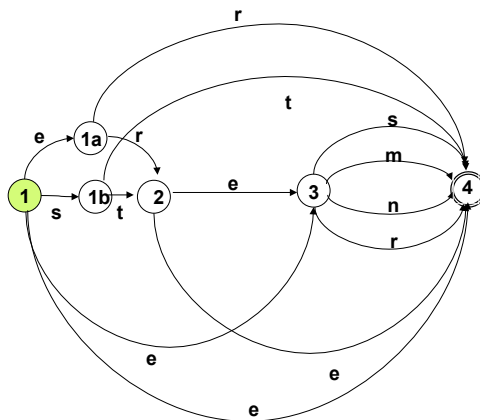
Pfadsuche als Breitensuche



Eingabewort:

Agenda: 1 -- klein eres

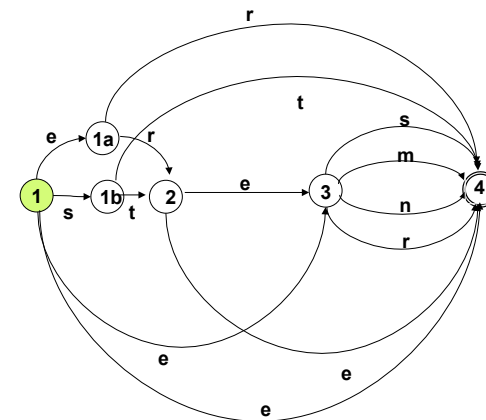
Pfadsuche als Breitensuche



Eingabewort: klein eres

Agenda: _____

Pfadsuche als Breitensuche

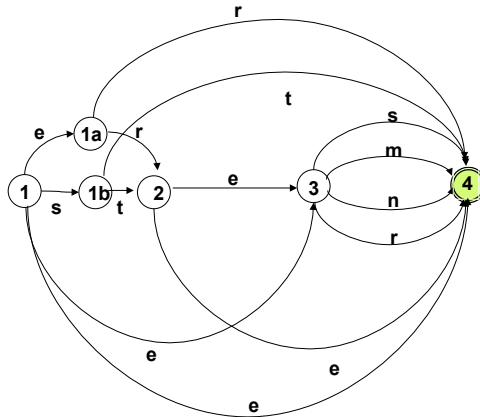


Eingabewort: klein eres

Agenda: 4 -- klein eres

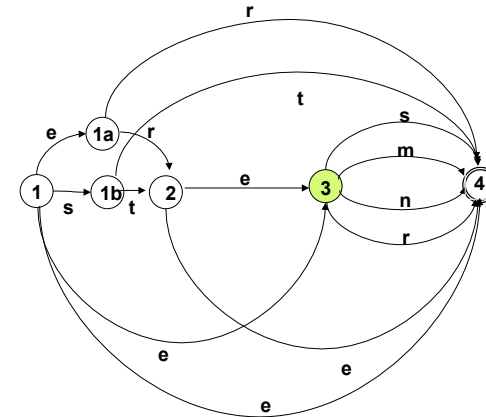
1a -- klein eres
3 -- klein eres

Pfadsuche als Breitensuche



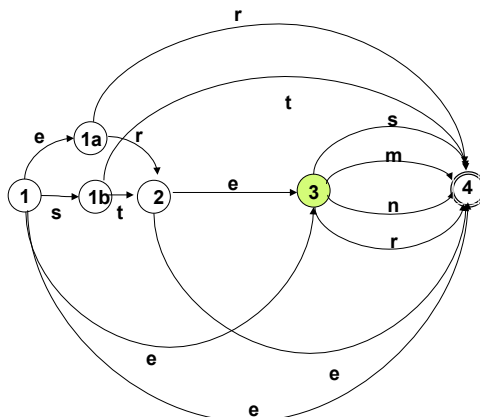
Eingabewort: klein eres Agenda: 3 -- klein eres

Pfadsuche als Breitensuche



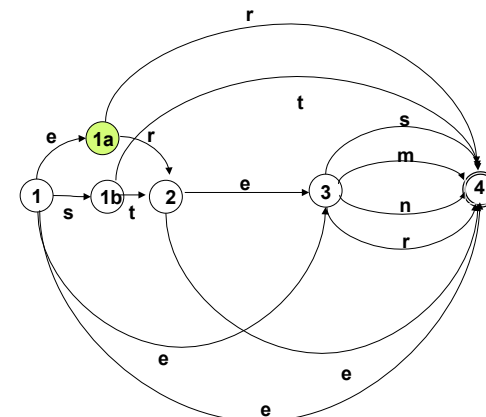
Eingabewort: klein eres Agenda: 1a -- klein eres

Pfadsuche als Breitensuche



Eingabewort: klein eres Agenda: 1a -- klein eres

Pfadsuche als Breitensuche

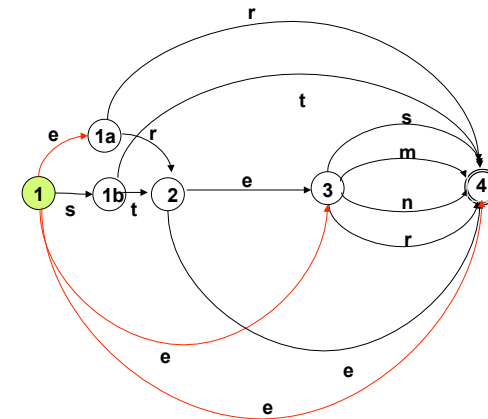


Eingabewort: klein eres Agenda: 4 -- klein eres
 2 -- klein eres
 4 -- klein eres

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.

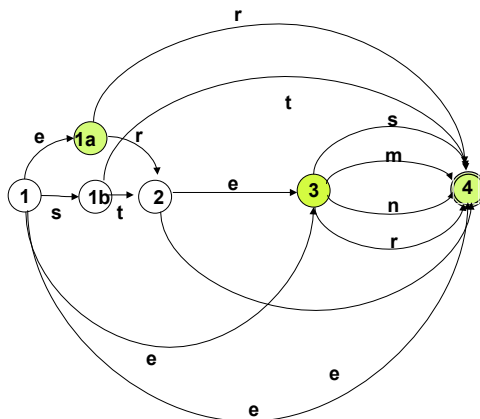
Pfadsuche als Breitensuche



Eingabewort: klein eres

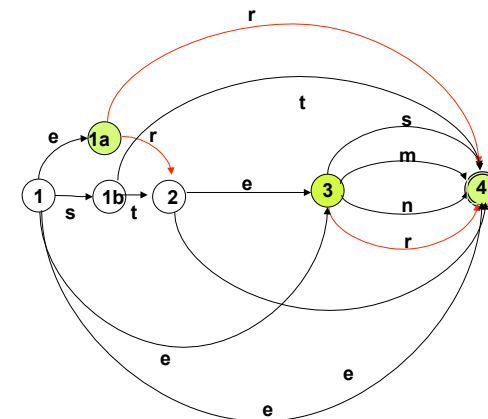
1a -- klein eres
 3 -- klein eres
 4 -- klein eres

Pfadsuche als Breitensuche



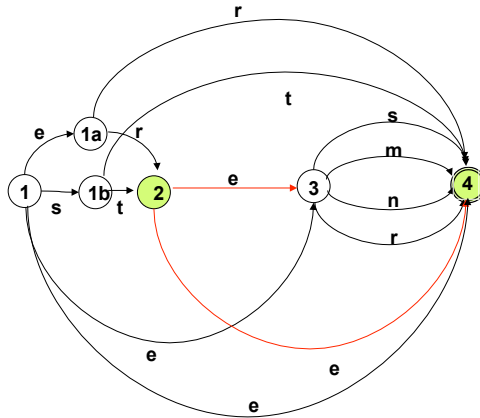
Eingabewort: klein eres

Pfadsuche als Breitensuche



Eingabewort: klein eres

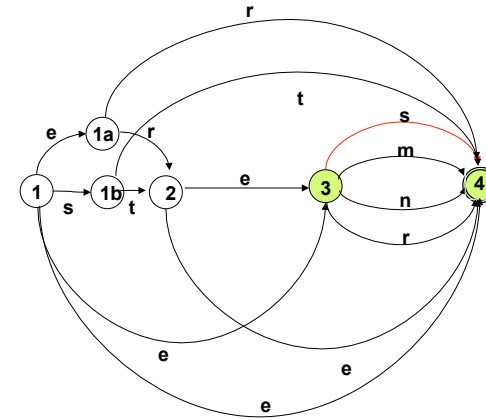
Pfadsuche als Breitensuche



Eingabewort: klein eres

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

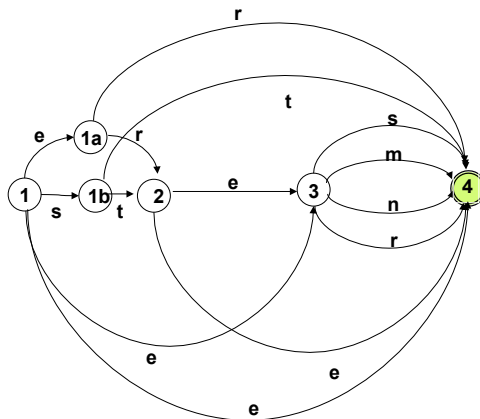
Pfadsuche als Breitensuche



Eingabewort: klein eres

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

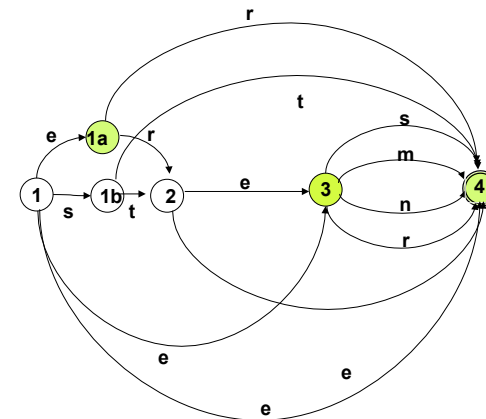
Pfadsuche als Breitensuche



Eingabewort: klein eres_

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Pfadsuche als Breitensuche



Eingabewort: klein e

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

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.

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UoS Computerlinguistik

Praktisches Vorgehen

Der Potenzautomat A' zu $A = \langle K, \Sigma, \Delta, s, F \rangle$ 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.

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UoS Computerlinguistik

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)
- $s' = \{s\}$
- $\delta(p', a) = \{q' \mid \text{es gibt } p \in p' \text{ und } \langle p, a, q \rangle \in \Delta\}$ für jedes $p' \subseteq K, a \in \Sigma$
- $q' \in F'$ gdw. $q' \cap F \neq \emptyset$

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UoS Computerlinguistik

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

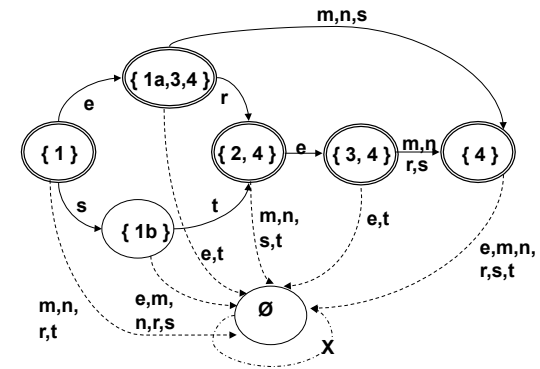
Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UoS Computerlinguistik

DEA für Adjektiv-Endungen, Übergangstabelle

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

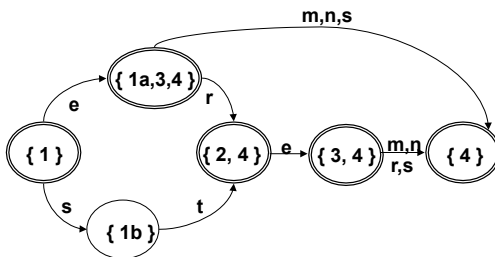
Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Das Diagramm



Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Das Diagramm, vereinfacht

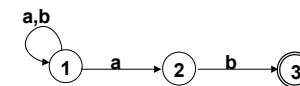


Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Potenzautomatenkonstruktion, ein weiteres Beispiel

NEA $A = \langle \{1,2,3\}, \{a,b\}, \Delta, 1, \{3\} \rangle$

Δ gegeben durch:



DEA

$A' = \langle \emptyset(\{1,2,3\}), \{a,b\}, \delta, \{1\}, F' \rangle$

$F' = \{\{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Potenzautomatenkonstruktion, Beispiel 2: Die Übergangstabelle

q	$\delta(q, a)$	$\delta(q, b)$
{1}	{1,2}	{1}
{2}	\emptyset	{3}
{3}	\emptyset	\emptyset
{1,2}	{1,2}	{1,3}
{1,3}	{1,2}	{1}
{2,3}	\emptyset	{3}
{1,2,3}	{1,2}	{1,3}
\emptyset	\emptyset	\emptyset

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Potenzautomatenkonstruktion, Beispiel 2: Die Übergangstabelle

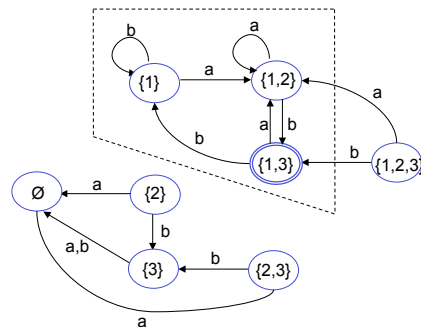
q	$\delta(q, a)$	$\delta(q, b)$
{1}	{1,2}	{1}
{2}	\emptyset	{3}
{3}	\emptyset	\emptyset
{1,2}	{1,2}	{1,3}
{1,3}	{1,2}	{1}
{2,3}	\emptyset	{3}
{1,2,3}	{1,2}	{1,3}
\emptyset	\emptyset	\emptyset

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Potenzautomatenkonstruktion, Beispiel 2: Das Zustandsdiagramm

Nur ein Teil der Zustände ist vom Startzustand aus erreichbar.

Die übrigen Zustände sind funktionslos.



Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Morphologiesysteme

- Flexionsmorphologie: Lemmatisierung/Stemming
 - *veranstalt+et, Veranstaltung+en*
- Ableitungs-/Derivationsmorphologie
 - *Veranstalt+ung, un+glaubwürdig*
- Komposita-Zerlegung
 - *Fach+veranstaltung, glaub+würdig*

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Anforderungen an Morphologiesysteme

- Korrektheit
- Vollständigkeit / Abdeckung (engl. „coverage“)
- Effizienz

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Korrektheit

- Flexionsmorphologie: Typischerweise unproblematisch, Korrekt, wenn die Flexionsklassen im Lexikon korrekt angegeben sind.
- Kompositazerlegung:
 - **Übergenerierung** ist ein massives Problem
 - ... wenn sie nicht durch Zusatzmechanismen behoben wird (Blockierungslisten, statistische Gewichtung)
- Derivationsmorphologie:
 - Tendenziell Übergenerierung (Semiproduktivität)
 - Tendenziell semantisch irreführende Identifikation von Stämmen

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Übergenerierung: Beispiele aus der Praxis

- Ein klassisches Beispiel aus der maschinellen Übersetzung (Systran, um 1980)
 - Barbarei
 - > nightclub nightclub egg
 - Bar|bar|ei
- Ein Beispiel aus der Rechtschreibkonversion (Corrigo, um 2000)
 - Hufeisenniere
 - > Hufeisenniere
 - Huf|ei|senn|niere

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Wortbildungsmorphologie: Ableitung/ Derivationsmorphologie

Derivationsmorphologie ist in verschiedener Hinsicht unsystematisch:

- viele Ableitungspräfixe und -suffixe sind semiproduktiv:
- viele Ableitungen sind semantisch "nicht transparent": Sie haben eine konventionelle, lexikalisierte Bedeutung, die mit der Bedeutung des Stammworts nicht in systematischer Beziehung steht.

Beispiele:

- die **Lesung** bezeichnet den Akt des Vorlesens,
- die **Singung** ist unmöglich
- die **Vorlesung** gibt es, bezeichnet aber nicht den Akt des Vorlesens,
- die **Schreibung** nicht den Akt des Schreibens

Vorlesung "Einführung in die CL" 2010/2011 © M. Pinkal UdS Computerlinguistik

Abdeckung

- Aktuelle Morphologiesysteme haben eine gute bis sehr gute Abdeckung (s. z.B. Word-Rechtschreibung)
- Abdeckung und Korrektheit allein sind für sich genommen keine guten Bewertungskriterien:
 - Man kann Korrektheit billig auf Kosten der Abdeckung erreichen und umgekehrt.
 - Ziel: Zuverlässigkeit bei gleichzeitig großer Abdeckung

Effizienz

- Grundsätzlich: Morphologische Analyse benötigt lineare Zeit in Abhängigkeit von der Länge der Eingabe.
- Gute Morphologiesysteme liegen im Bereich von < 1 ms pro Wortform (auf normalem PC)
- Durch Vorverarbeitung, Zwischenspeichern von Analysen, Indexierung etc. lässt sich für größere Dokumente die Zeit pro Textwort auf den unteren μ s-Bereich drücken.
- Das ist
 - exzellent für Online- und Offline-Rechtschreibkorrektur
 - akzeptabel für begrenzte Datensammlungen (große Textkorpora, Firmen-Intranet etc.)
 - zu langsam fürs Web-Suchmaschinen.