Einführung in die Computerlinguistik:

Morphologie und Automaten I

WS 2009/2010

Manfred Pinkal

Morphologie

- Morphologie ist der Teilbereich der Linguistik, der sich mit der internen Struktur von Wörtern befasst.
- Die wesentlichen Aufgaben der Morphologie in der Computerlinguistik sind
  - die Reduktion komplexer Wörter bzw. Wortformen auf ihre Bestandteile
  - die Identifikation von grammatischer Information, die in der Wortform kodiert ist (z.B. Kasus, Numerus, Tempus)

Teilbereiche der Morphologie

- Flexion:
  Deklination, Konjugation von Substantiven, Verben, Adjektiven, Pronomina
  frag+te+st
  ge+frag+t
- Derivation (Ableitung)
  Er+kenn+ung
- Komposition (Zusammensetzung)
  Sprach+erkennung+s+technik

Elemente der morphologischen Struktur

- Stämme, Präfixe, Suffixe (in Flexion und Derivation)
  - Flexionsmorphologie:
    frag+te+st
    ge+frag+t
  - Derivationsmorphologie:
    Er+kenn+ung

Elemente der morphologischen Struktur

- Stämme, Präfixe, Suffixe (in Flexion und Derivation)
  - Flexionsmorphologie:
    frag+te+st
    ge+frag+t
  - Derivationsmorphologie:
    Er+kenn+ung

Elemente der morphologischen Struktur

- Stämme, Präfixe, Suffixe (in Flexion und Derivation)
  - Flexionsmorphologie:
    frag+te+st
    ge+frag+t
  - Derivationsmorphologie:
    Er+kenn+ung

Elemente der morphologischen Struktur

- Grund- , Bestimmungswörter, Fugenelemente (in der Komposition)
  Sprach+erkennung+s+technik
  Sprach+erkennung+s+technik
- Bestimmungswort: *Sprach* + Grundwort: *erkennung*
- Bestimmungswort: *Spracherkennung* +Fugenelement: *s* + Grundwort: technik
- Fugenelemente sind keine Flexionssuffixe. Sie sind aufgrund der phonologischen Struktur nur partiell vorhersagbar und müssen deshalb im Lexikon gelistet werden.

Elemente der morphologischen Struktur

- Stämme, Präfixe, Suffixe (in Flexion und Derivation)
- Grund- , Bestimmungswörter, Fugenelemente (in der Komposition)
- Modifikation von Stämmen (Umlaut, Ablaut)
  - *Mutter, Mütter*
  - *schwimmen, schwamm, geschwommen*
- Morpho-phonologische Prozesse

Morpho-phonologische Prozesse

- Systematische Modifikation, Einfügung und Tilgung von Lauten/Phonemen, die aus der phonetisch/ phonologischen Struktur vorhersagbar sind:

  Einfügung: *bad+st → badest*

  Tilgung: *ras+st → rast*

- Behandlung durch allgemeine Tilgungs-/Enfügungs-/ Modifikationsregeln in Morphologiesystemen

Morphologische Verarbeitung in der Computerlinguistik

- Systeme zur morphologischen Analyse werden in vielen computerlinguistischen Anwendungen eingesetzt (z.B. Rechtschreib- und Grammatikkorrektur; Informationszugriff; maschinelle Übersetzung).
- Komponenten:
  - Lemmatisierung: Flexionsmorphologische Analyse: Ermittlung des Stammes/Lemmas („stemming") und ggf. der in den Flexionsformen enthaltenen grammatischen Information
  - Wortformengenerierung: Erzeugung der (in Genus, Numerus, Kasus etc.) passenden Wortform zu einem Stamm
  - Derivativ- und Komposita-Zerlegung: Reduktion komplexer Wörter (Ableitungen und Zusammensetzungen) auf ihre Bestandteile

Ein Flexionsbeispiel aus dem Türkischen

- *Evlerinizdeyiz*
- *Ev+ler+iniz+de+yiz*
- *Haus+pl+poss-2.pers-pl+in+wir-sind*
- *"Wir sind in euren Häusern"*

Ein Kompositionsbeispiel aus dem Deutschen

- Forstspezialrückeschlepper
- Forst+spezial+rücke+schlepper

Bei diesen Fahrzeugen handele es sich nämlich um Forstspezialrückeschlepper, deren Fahren als Beispiel in der Lohngr. W 7 Fallgr. 1 angeführt sei. (...) Ein Forstspezialschlepper, dessen Bestimmung das "Rücken" sei, sei nach den Regeln des allgemeinen Sprachgebrauchs ein Forstspezialrückeschlepper. Dabei spiele es auch keine Rolle, in welcher Reihenfolge die Bestandteile dieses Wortes verwendet seien. Mit Forstspezialrückeschlepper gleichbedeutend wäre auch "Spezialforstrückeschlepper", "Forstrückespezialschlepper" oder "Rückeforstspezialschlepper".

Aus einer Urteilsbegründung des Bundesarbeitsgerichtes

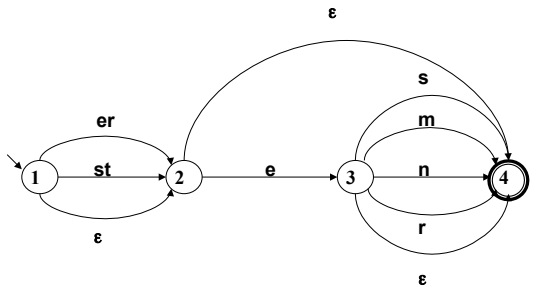
Morphologische Verarbeitung in der Computerlinguistik

- Methodisches Werkzeug für alle Aufgaben morphologischer Verarbeitung sind „Endliche Automaten“.
- Wir betrachten die Verwendung endlicher Automaten an einer vergleichsweise einfachen Teilaufgabe der Lemma-Ermittlung.

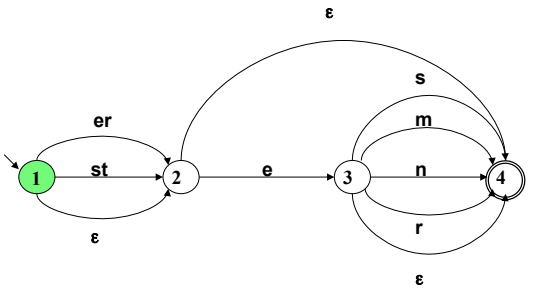
Adjektivflexion: Paradigma
(nur sog. „starke Flexion“)

klein+er	klein+e	klein+es	klein+e
klein+es/en	klein+er	klein+es/en	klein+er
klein+em	klein+er	klein+em	klein+en
klein+en	klein+e	klein+es	klein+e
klein+er+er	klein+er+e	klein+er+es	klein+er+e
klein+er+es/en	klein+er+er	klein+er+es/en	klein+er+er
klein+er+em	klein+er+er	klein+er+em	klein+er+en
klein+er+en	klein+er+e	klein+er+es	klein+er+e
klein+st+ er	klein+st+e	klein+st+ es	klein+st+e
klein+st+es/en	klein+st+er	klein+st+es/en	klein+st+er
klein+st+em	klein+st+er	klein+st+em	klein+st+en
klein+st+en	klein+st+e	klein+st+es	klein+st+e

Adjektivendungen: Kompakte Darstellung durch Zustandsdiagramm

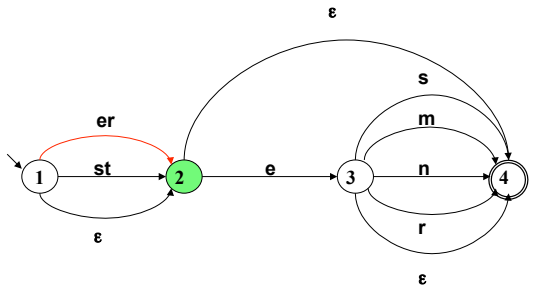


Funktion des Zustandsdiagramms [1]



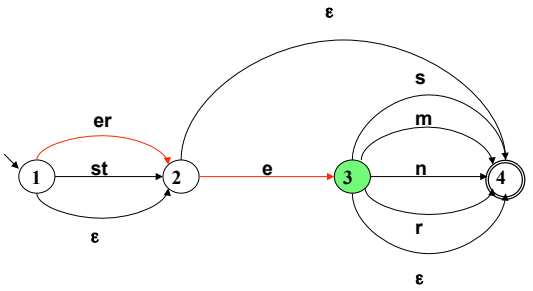
klein eres

Funktion des Zustandsdiagramms [2]



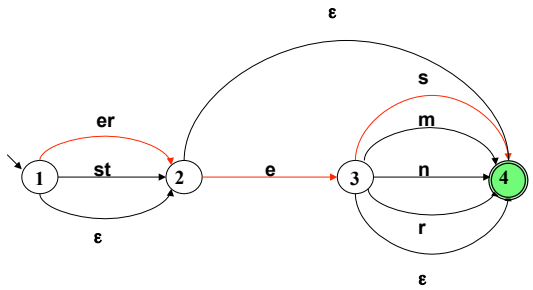
klein er<u>es</u>

Funktion des Zustandsdiagramms [3]



klein er<u>es</u>

Funktion des Zustandsdiagramms [4]



klein eres_

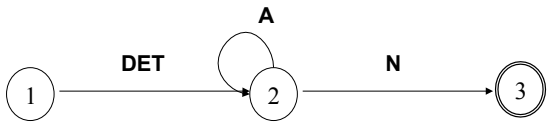
Zustandsdiagramme: Ein zweites Beispiel [1]

- Wortarten kombinieren sich in bestimmter Weise zu Satzteilen
- Um zu testen, ob eine Wortfolge in einem Dokument eine erlaubte Abfolge von Wortarten darstellt, können Zustandsdiagramme benutzt werden.
- Das folgende Zustandsdiagramm akzeptiert bestimmte einfache Nominalausdrücke, wie

  der Wagen

  eine interessante Vorlesung

  das neue schöne rote Dach



Zustandsdiagramme: Ein zweites Beispiel [2]

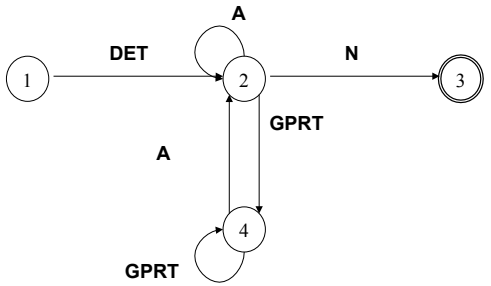
- Das „Alphabet“ des Zustandsdiagramms sind Wortart-Bezeichnungen („Wortart-Tags“ oder auch „POS-Tags“, POS für „part of speech“, engl. für „Wortart“), in unserem Beispiel DET, A, N
  - DET (Artikel)
  - A (adjektivisches Attribut)
  - N (Gattungssubstantiv, Gattungsnomen)
- Erkannte „Worte“ sind erlaubte Abfolgen von Wortartsymbolen, z.B. „DET N“, „DET A N“, „DET A A A N“
- Im Gegensatz zum Adjektivendungsdiagramm akzeptiert das Nominalausdrucksdiagramm beliebig lange Worte und beschreibt eine unendliche Sprache. Grund: Es enthält eine Schleife, es ist zyklisch.

Zustandsdiagramme: Ein zweites Beispiel [3]

Das abgebildete Diagramm akzeptiert auch Adjektive, die mit (einer oder mehreren) Gradpartikeln (GPRT) versehen sind, wie z.B.

eine ziemlich interessante Vorlesung

das recht neue sehr sehr schöne rote Dach



Definitionen: Alphabet und Wort

- Ein Alphabet Σ ist eine endliche, nicht-leere Menge von Symbolen.
- Ein Wort w über dem Alphabet Σ ist eine endliche Kette von Symbolen aus Σ .
- Die Wortlänge |w| eines Wortes w ist die Anzahl der verketteten Symbole von w.
- Das leere Wort ε ist das Wort mit Wortlänge 0 ($|\varepsilon|=0$).

Definitionen: Sprache

- Ein Sprache über dem Alphabet Σ ist eine Menge von Worten über Σ .

  Zwei besondere Sprachen:
- Die leere Wortmenge $\varnothing$ heißt die „leere Sprache“.
- Die maximale Sprache, die die Menge aller Worte über dem Alphabet Σ umfasst, ist Σ^* (der „Stern“ von Σ).

  Anmerkung:

  Für jedes Alphabet Σ gilt: $\varepsilon \in \Sigma^*$.

Beispiele

Beispiel 1:

Σ = {e, m, n, r, s, t}

e, er, rrrrr, mnstmnst, ... $\in \Sigma^*$

L = {ε, e, er, em, en, es, ere, erer, erem, eren, eres, st, ste, stem, sten, ster, stes}

Beispiel 2:

Σ = {DET, A, N}

L = {DET N, DET A N, DET A A N...}

Alternative Formulierung:

L = {DET A^n N| n $\in \mathbb{N}$}

Bemerkungen:

- Mit $\mathbb{N}$ bezeichnen wir hier die Menge der natürlichen Zahlen inklusive 0.
- a^n ist die Kette, die durch n-faches Hintereinander-schreiben des Symbols a entsteht (für n=0 ist $a^n = \varepsilon$)

Beispiele

Beispiel 3:

Σ = {0,1,2,3,4,5,6,7,8,9}

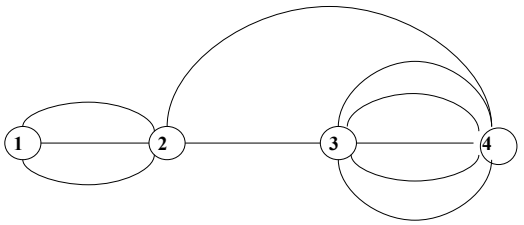
L = {$x_1 \ldots x_n$ y| n $\in \mathbb{N}$, $x_i \in \Sigma$ für $1 \leq i \leq n$, y $\in$ {0,5}, n $\in \mathbb{N}$}

(die Menge der durch 5 teilbaren natürlichen Zahlen, wenn wir Ziffernfolgen mit 0-Präfixen ebenfalls zulassen)

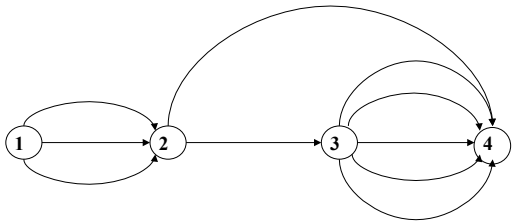
Definition: Zustandsdiagramm [1]

Ein Zustandsdiagramm ist ein gerichteter Graph mit Kanteninschriften.

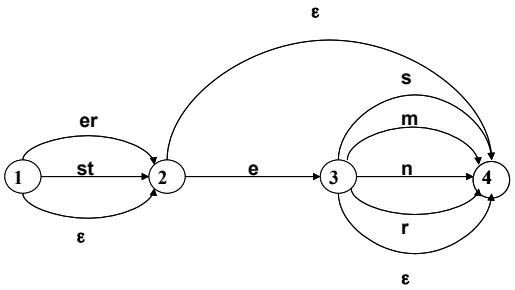
Ein Graph



Ein gerichteter Graph



Ein gerichteter Graph mit Kanteninschriften



Zustandsdiagramm

Ein Zustandsdiagramm besteht aus

- Knoten (Zuständen) (im Beispiel: 1,2,3,4)
- davon ein Startknoten (1)
- einem oder mehreren Zielknoten (4)
- Kanten zwischen den Knoten, die
  - gerichtet und
  - beschriftet sind
- Die Kanteninschriften bestehen aus Ketten von Symbolen über einem Alphabet (im Beispiel: e,r,m,n,s,t).
- Auch das leere Wort (ε) ist als Kantenbeschriftung zugelassen; ε ist kein Symbol des Alphabets, sondern bezeichnet den Grenzfall der „aus 0 Symbolen bestehenden" leeren Kette.

Definition: Zustandsdiagramm [2]

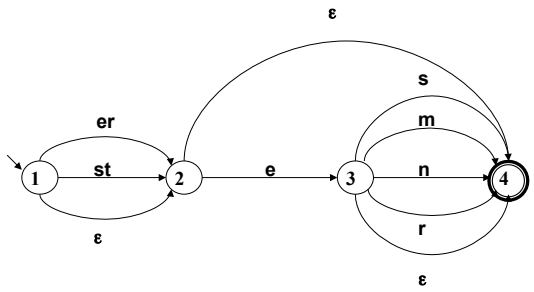
Formal wird ein Zustandsdiagramm definiert als ein Quintupel (Folge von 5 Elementen)

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)

Anmerkung: Das Zustandsdiagramm heißt auch „nicht-deterministischer endlicher Automat" (NEA), engl.: „non-deterministic finite-state automaton" (NFA) (Erklärung später)

Adjektivendungen: Zustandsdiagramm



Beispiel: Das Adjektivendungs-Diagramm

NEA A = <K, Σ, Δ, s, F> mit

- K= {1,2,3,4} (Zustände)
- Σ = {e, m, n, r, s, t} (Alphabet)
- s = 1 (Startzustand)
- F = {4} (einziger Endzustand)
- Δ = {<1,*er*,2>, <1,*st*,2>, <1,ε, 2>, <2,*e*,3>, <2,ε,4>, ...} (Übergangsrelation)

Die Interpretation des Zustandsdiagramms

- Das Zustandsdiagramm beschreibt alle Symbolkombinationen oder „Worte", die sich dadurch ergeben, dass man das Diagramm vom Startknoten zu einem Zielknoten durchläuft und die Inschriften der Kanten, die man dabei beschreitet, aufliest und aneinanderhängt. Man nennt die Menge der Worte, die sich so erzeugen lassen, die vom Diagramm beschriebene „Sprache".
- Üblicherweise werden Diagramme verwendet, um Eingabeketten zu testen. Man spricht von endlichen Automaten, und sagt, dass ein Automat ein Wort akzeptiert.

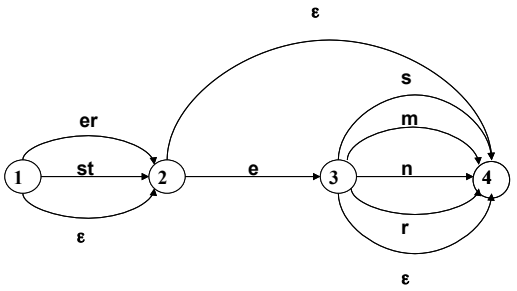
Durch NEA akzeptiertes Wort/definierte Sprache

- Ein Wort $w \in \Sigma^*$ wird durch den NEA A =<K,Σ,Δ,s,F> akzeptiert gdw. es eine Folge von Kanten (einen Pfad durch den NEA) $<s,u_1, k_1>, <k_1, u_2, k_2>, \ldots, <k_{n-1}, u_n, k_n>$ gibt, sodass $k_n \in F$ und $u_1 u_2 \ldots u_n = w$ (die Konkatenation, das Aneinanderhängen der Inschriften der durchlaufenen Kanten ergibt das Wort w).
- Die vom NEA A =<K,Σ,Δ,s,F> definierte (akzeptierte) Sprache L(A) ist die Menge der von A akzeptierten Worte.

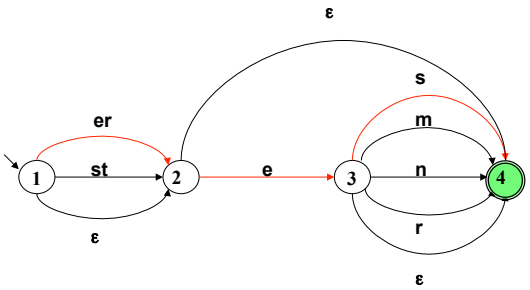
Eine methodische Bemerkung

- Die Definition des Zustandsdiagramms/NEA spezifiziert eine formale Notation, die für sich genommen keine Bedeutung hat.
- Durch die Definitionen der letzten Folie (akzeptiertes Wort/definierte Sprache) wird diese Datenstruktur interpretiert: Wir verwenden Zustandsdiagramme, um die Zugehörigkeit von Symbolketten zu Sprachen zu definieren und zu testen.
- Hinzu kommen muss ein handhabbares Verfahren, ein Algorithmus, um den Zugehörigkeitstest tatsächlich durchzuführen.

Ein Suchproblem

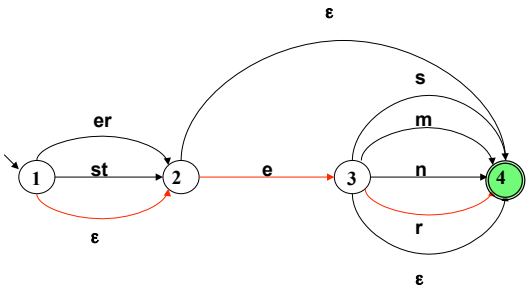


Ein Weg durchs Diagramm



klein eres|

Ein alternativer Weg durchs Diagramm



klein er|es

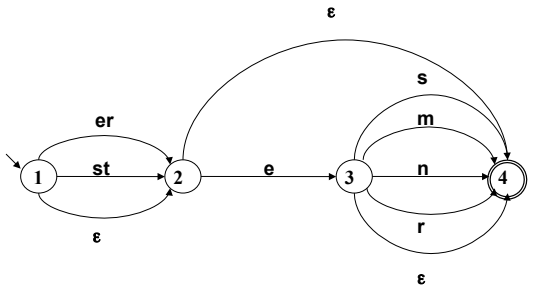
Ein Suchproblem

- Das Diagramm erlaubt typischerweise an einem Knoten/ einem Zustand mehrere Übergänge bei derselben Eingabe (deshalb „nicht-deterministisch“).
- Die zufällige Wahl einer Kante kann sich erst viel später als falsch herausstellen. Sie gibt keine Gewähr, dass tatsächlich alle möglichen Wege durch das Diagramm getestet wurden.
- Wir benötigen ein Verfahren, das uns die vollständige Suche garantiert.

Ein Verfahren für die Pfadsuche: Die Idee

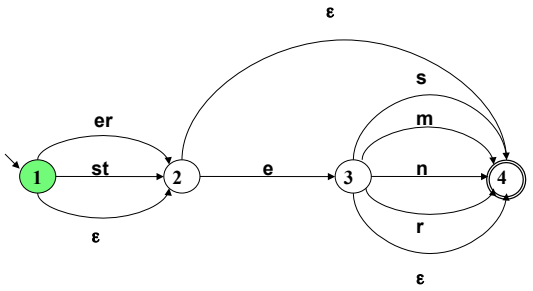
- Angenommen, wir befinden uns an einem Knoten k des Diagramms (dem „aktuellen Zustand“) und an einer Position im Eingabewort w (der „aktuellen Position“) – beides zusammen nennen wir die aktuelle Konfiguration.
- Wir testen, welche Kanten wir beschreiten können.
- Wir rücken nicht direkt im Automaten vor, sondern legen die alternativ möglichen Resultatkonfigurationen – Zustand und Position im Wort – in einer Liste noch zu erledigender Teilaufgaben, der Agenda, ab.
- Dann nehmen wir einen Eintrag von der Agenda. – Welchen? Es gibt unterschiedliche Möglichkeiten, die unterschiedlichen Suchverfahren entsprechen.
- Wir betrachten die Agenda zunächst als Stapel („Stack“), legen neue Aufgaben oben auf die Agenda und nehmen einzelne Aufgaben ebenfalls von oben von der Agenda ("Last In – First Out").
- Wir führen die Aufgabe aus (d.h., rücken im Eingabewort auf die bezeichnete Stelle vor und gehen in den bezeichneten Zustand), testen mögliche nächste Schritte, legen die Resultate auf die Agenda usw. – bis wir die Eingabe erfolgreich abgearbeitet haben (akzeptieren!) oder die Agenda keine Aufgaben mehr enthält ist (zurückweisen!).

Pfadsuche: Startkonfiguration: Startzustand und Eingabewort auf der Agenda



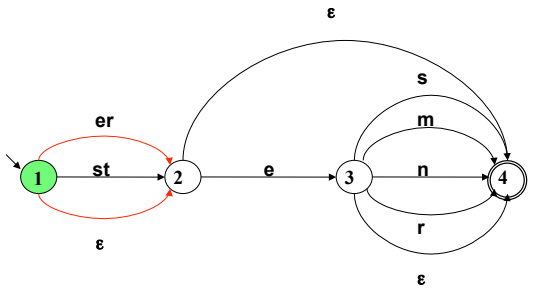
Eingabewort: Agenda: 1 -- klein eres

Pfadsuche: Nimm Aufgabe von der Agenda



Eingabewort: klein eres Agenda:

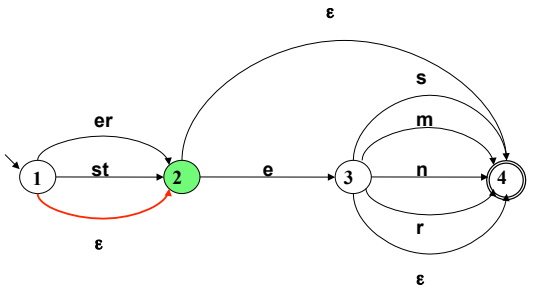
Pfadsuche: Generiere neue Aufgaben



Eingabewort: klein eres

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

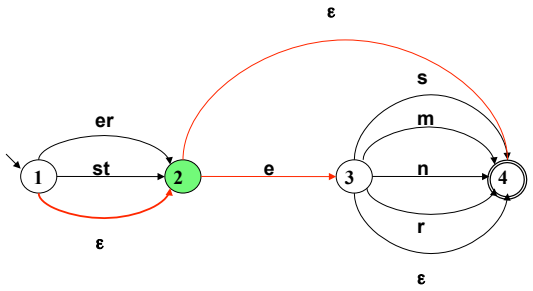
Pfadsuche: Nimm Aufgabe von der Agenda



Eingabewort: klein eres

Agenda:
2 -- klein eres

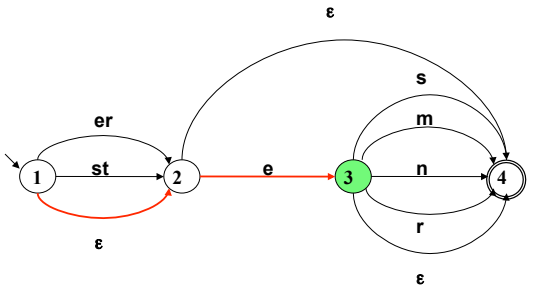
Pfadsuche: Generiere neue Aufgaben



Eingabewort: klein eres

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

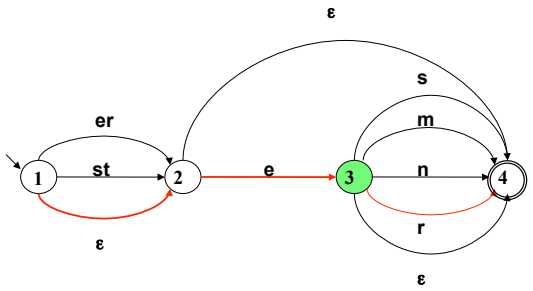
Pfadsuche: Nimm Aufgabe von der Agenda



Eingabewort: klein eres

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

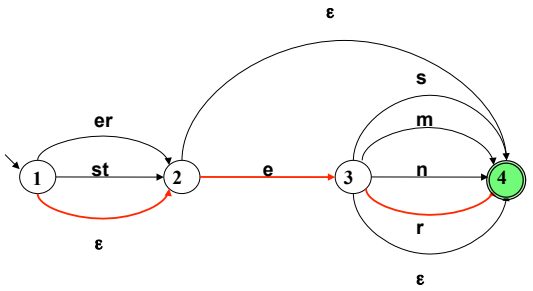
Pfadsuche: Generiere neue Aufgaben



Eingabewort: klein eres

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

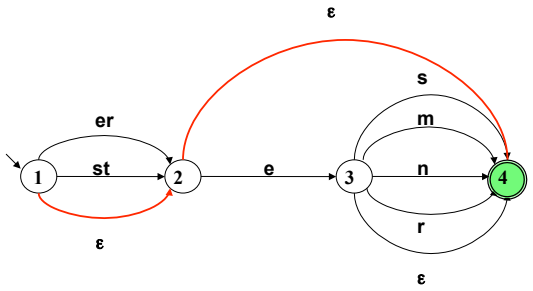
Pfadsuche: Nimm Aufgabe von der Agenda
Keine neue Aufgabe!



Eingabewort: klein eres

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

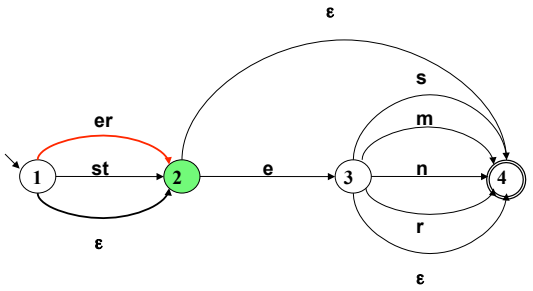
Pfadsuche: Nimm Aufgabe von der Agenda
Keine neue Aufgabe!



Eingabewort: klein eres

Agenda:
2 -- klein eres

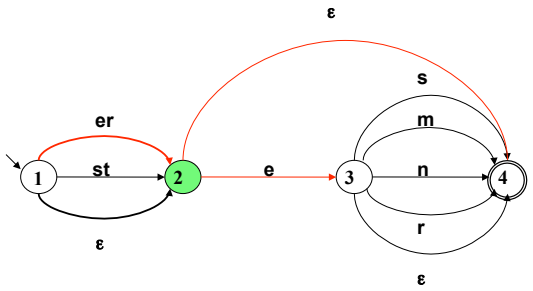
Pfadsuche: Nimm Aufgabe von der Agenda
Backtracking!



Eingabewort: klein eres

Agenda:

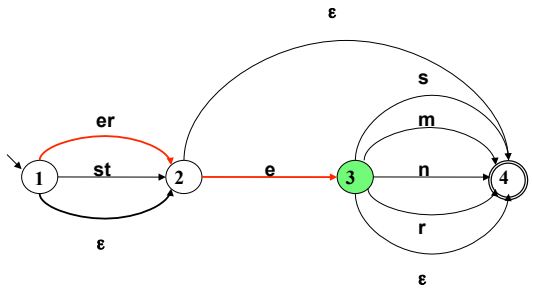
Pfadsuche: Generiere Aufgaben



Eingabewort: klein eres

Agenda: 3 -- klein eres
4 -- klein eres

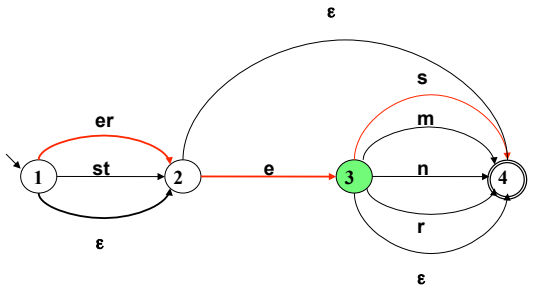
Pfadsuche: Nimm Aufgabe von der Agenda



Eingabewort: klein eres

Agenda: 4 -- klein eres

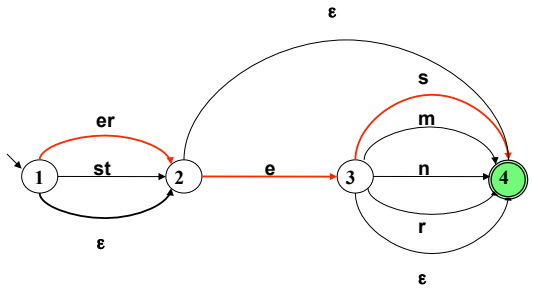
Pfadsuche: Generiere Aufgabe



Eingabewort: klein eres

Agenda: 4 -- klein eres_
4 -- klein eres

Pfadsuche: Nimm Aufgabe von der Agenda: Eingabe abgearbeitet, Zielzustand: Akzeptiere!



Eingabewort: klein eres_

Agenda: 4 -- klein eres

Anmerkung zum Pfadsuche-Algorithmus

- Der vorgestellte Pfadsuche-Algorithmus ist ein Beispiel für „Tiefensuche mit Backtracking“: 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 systematisch die in der Agenda gespeicherten, noch nicht begangenen Verzweigungen ausprobiert.
- Der Algorithmus ist vollständig nur dann, wenn das Diagramm/der Automat keine Leerwort-Zyklen enthält (Schleifen, bei deren Durchlaufen das leere Wort abgearbeitet wird).
- Der Algorithmus ist ineffizient: Bei einem maximalen Verzweigungsfaktor n 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 anordnet
  - den Suchalgorithmus 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