Einführung in die Computerlinguistik:

Morphologie und Automaten

WS 2021/2022

Prof. Vera Demberg

Morphologie

Letztes Mal haben wir gesehen, wie Wörter aufgebaut werden.

- Was ist ein *Morphem*?
- Wie unterscheiden sich Flexion, Derivation und Komposition?
- Was sind Beispiele für Präfix, Suffix und Stamm?
- Was sind Beispiele für Grundwörter, Bestimmungswörter und Fugenelemente?

Morphologie ist wichtig, weil nicht alle Wörter und Wortformen im Lexikon gespeichert werden können.

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 Forstspezialrücke-schlepper, 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

Es ist **unmöglich**, in einem Lexikon alle Formen aufzulisten und nachzuschauen!

Daher müssen brauchen wir eine Vorgehensweise, wie wir neue Wortformen automatisch zerlegen können.

Wie können wir morphologische Analysen automatisch durchführen?

Inhalt heute: Automaten

1. Morphologie
2. Automaten (Intuition)
3. Automaten (Formalisierung)
4. Suche in Automaten
5. Deterministische vs. Nicht-deterministische Automaten

Morphologische Verarbeitung in der Computerlinguistik

- Gängigste Methode:
  AUTOMATEN

Morphologische Verarbeitung in der Computerlinguistik

Eingaben:

•Geld

•Wahl per Taste

•Geldrückgabeknopf

Zustände:

•Die Eingaben werden intern als Zustände repräsentiert und verarbeitet.

Output:

•Cola / Schokolade...

•ERROR

Morphologische Verarbeitung in der Computerlinguistik

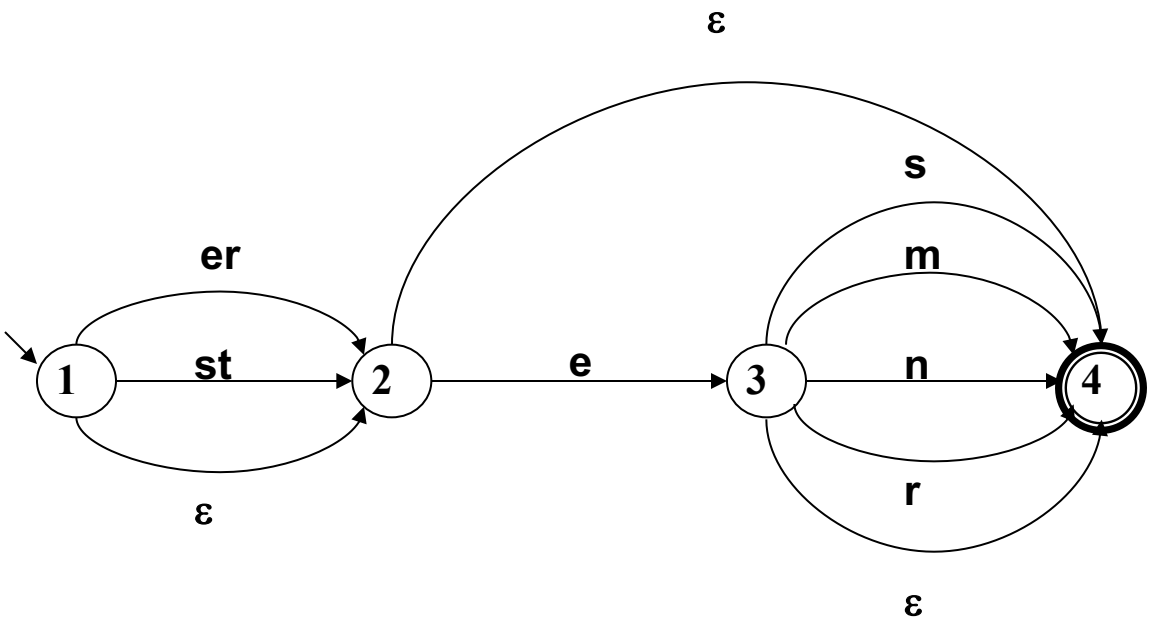
- Beispielproblem: Adjektivflexion

- Was für Eingaben / Zustände / Ausgaben müsste ein **Automat** haben, der korrekte von inkorrekten Adjektivformen unterscheiden kann?

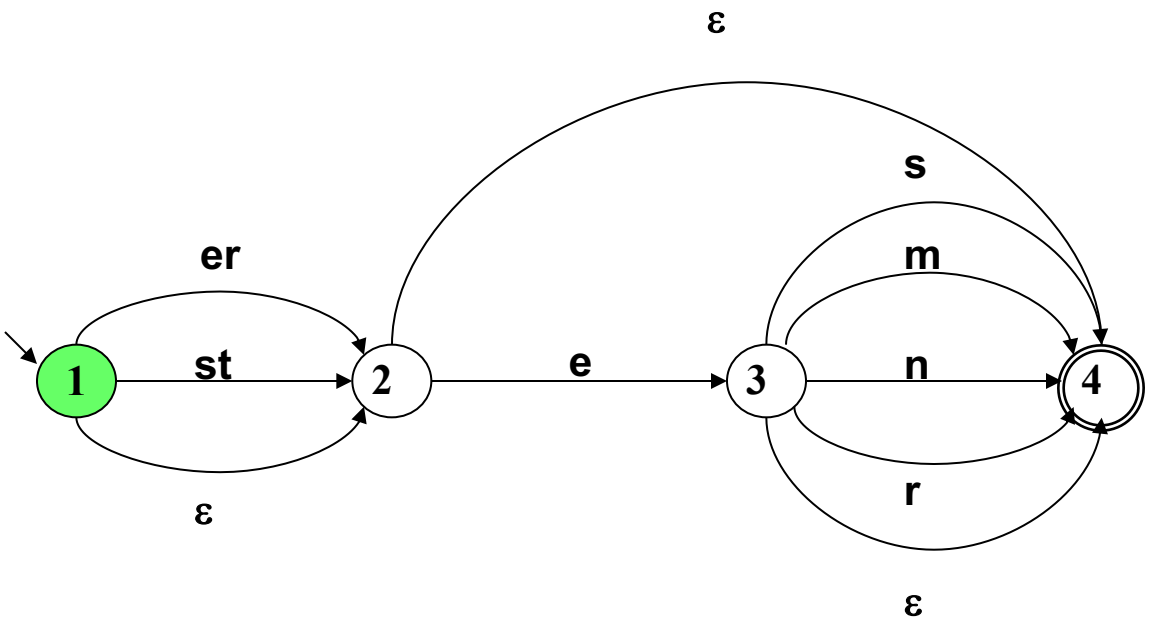
Adjektivflexion: Paradigma
(nur sog. „starke Flexion“)

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

Adjektivendungsautomat als „Zustandsdiagramm“

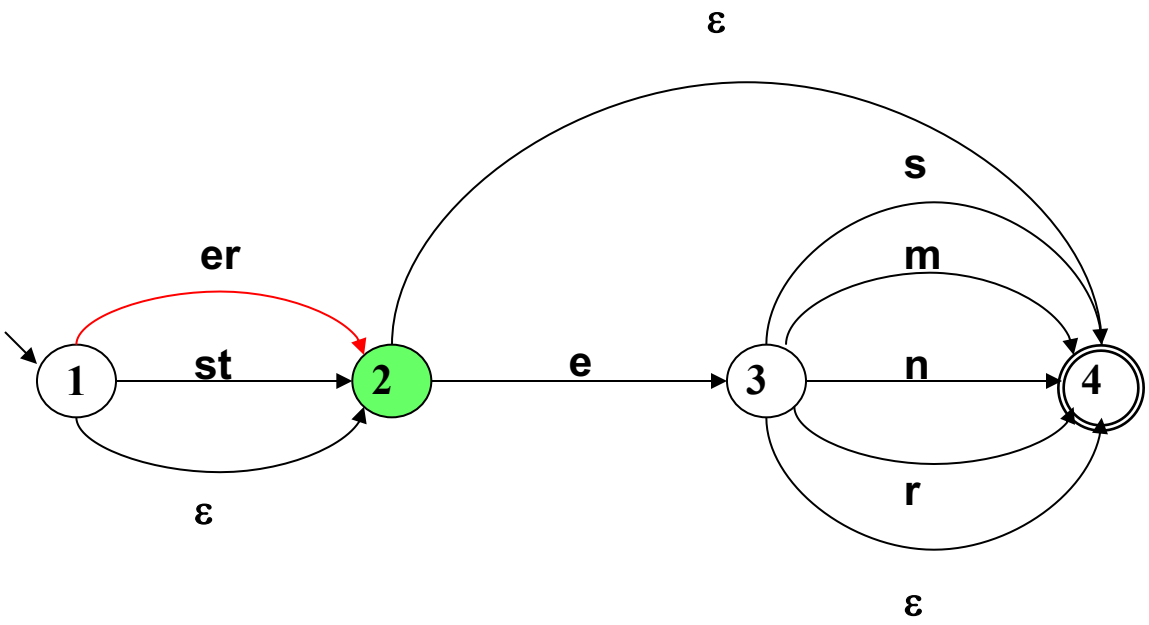


Funktion des Zustandsdiagramms [1]



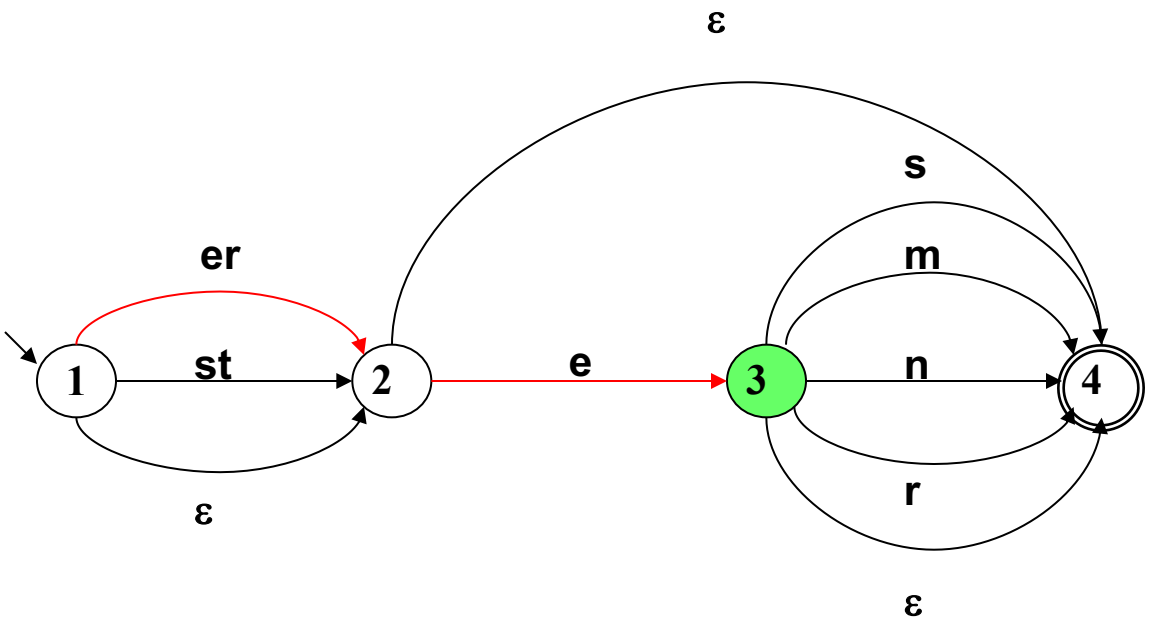
klein eres

Funktion des Zustandsdiagramms [2]



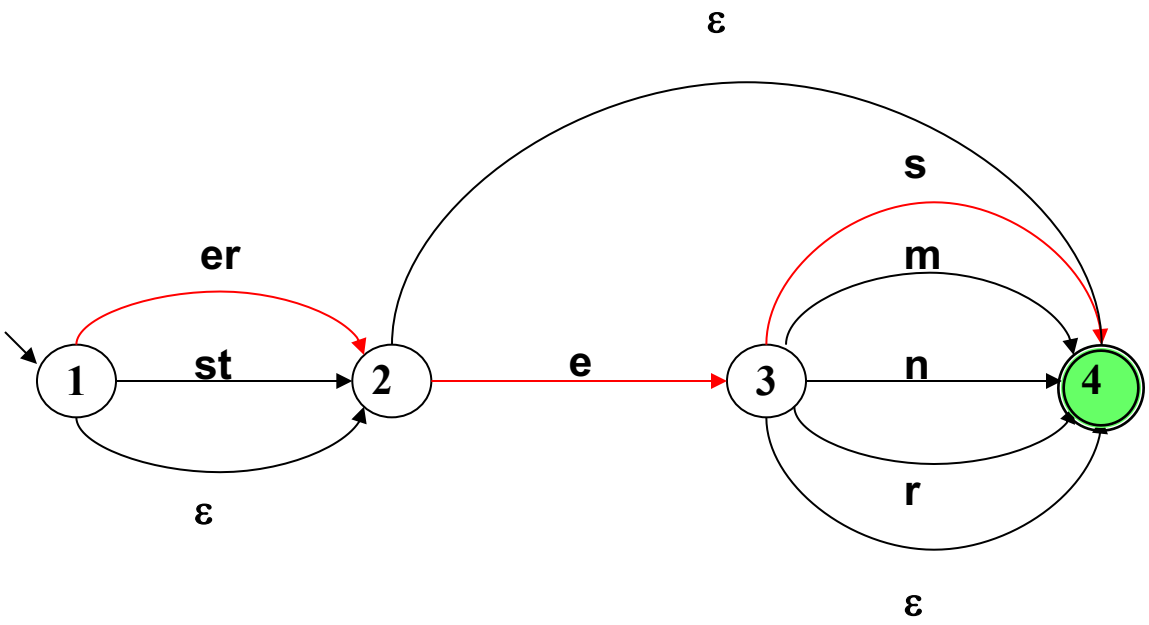
klein er<u>e</u>s

Funktion des Zustandsdiagramms [3]



klein eres

Funktion des Zustandsdiagramms [4]



klein eres_

Zustandsdiagramme: Ein zweites Beispiel [1]

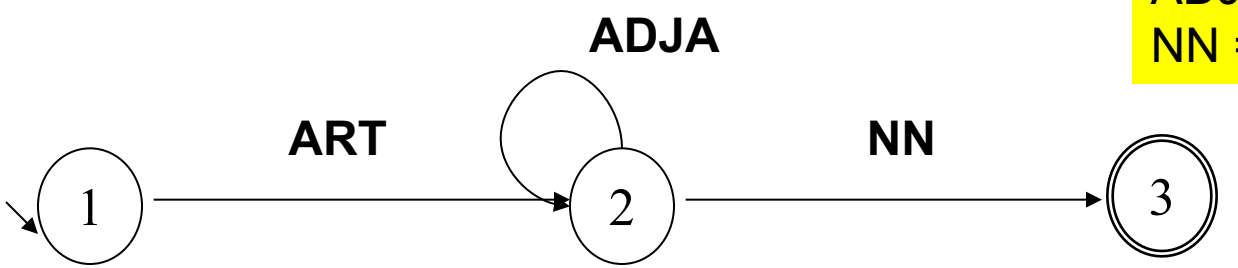
- Wortarten kombinieren sich in bestimmter Weise zu Satzteilen
- Um zu testen, ob eine Wortfolge in einem Dokument eine erlaubte Abfolge von Wortarten darstellt, können Zustandsdiagramme benutzt werden.
- Das folgende Zustandsdiagramm akzeptiert bestimmte einfache Nominalausdrücke, wie

  der Wagen

  eine interessante Vorlesung

  das neue schöne rote Dach

ART = Artikel
ADJA = Adjektiv
NN = Nomen

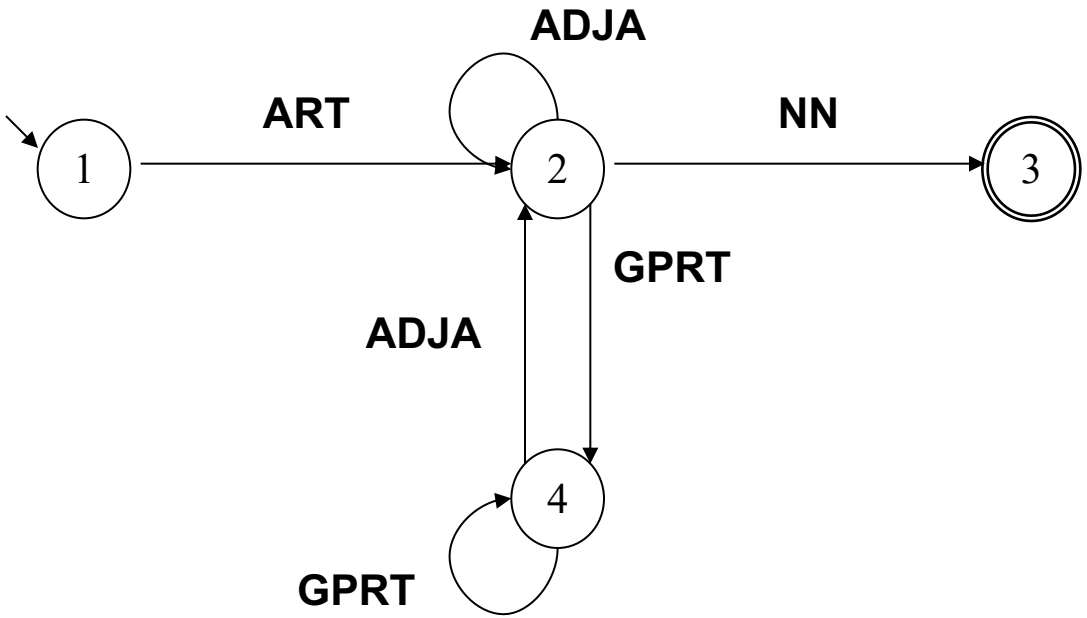


Zustandsdiagramme: Ein zweites Beispiel [2]

Wie kann das Diagramm erweitert werden, damit es auch auch Adjektive, die mit (einer oder mehreren) Gradpartikeln (GPRT) akzeptiert?

z.B. *eine ziemlich interessante Vorlesung*

das recht neue sehr sehr schöne rote Dach



Inhalt heute: Automaten

1. Morphologie
2. Automaten (Intuition)
3. Automaten (Formalisierung)
4. Suche in Automaten
5. Deterministische vs. Nicht-deterministische Automaten

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 L 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“ über Σ).

  Anmerkung:

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

Beispiele

Beispiel 1:

Alphabet Σ = {e, m, n, r, s, t}

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

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

Beispiel 2:

Σ = {ART, ADJA, NN}

L = {ART NN, ART ADJA NN, ART ADJA ADJA NN, ART ADJA ADJA ADJA NN, ...}

Alternative Formulierung:

L = {ART ADJA^n NN| 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 Hintereinanderschreiben des Symbols a entsteht (für n=0 ist $a^n = \varepsilon$)

Beispiele

Beispiel 1:

Alphabet Σ = {e, m, n, r, s, t}

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

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

Beispiel 2:

Σ = {ART, ADJA, NN}

L = {ART NN, ART ADJA NN, ART ADJA ADJA NN, ART ADJA ADJA ADJA NN, ...}

Alternative Formulierung:

L = {ART ADJA^n NN| n $\in \mathbb{N}$}

Im Gegensatz zum Adjektivendungsautomaten akzeptiert der Nominalausdrucksautomat beliebig lange Worte: er beschreibt eine **unendliche** Sprache.

Grund: Er enthält eine Schleife, er ist zyklisch.

Beispiele

Beispiel 3:

$\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$

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

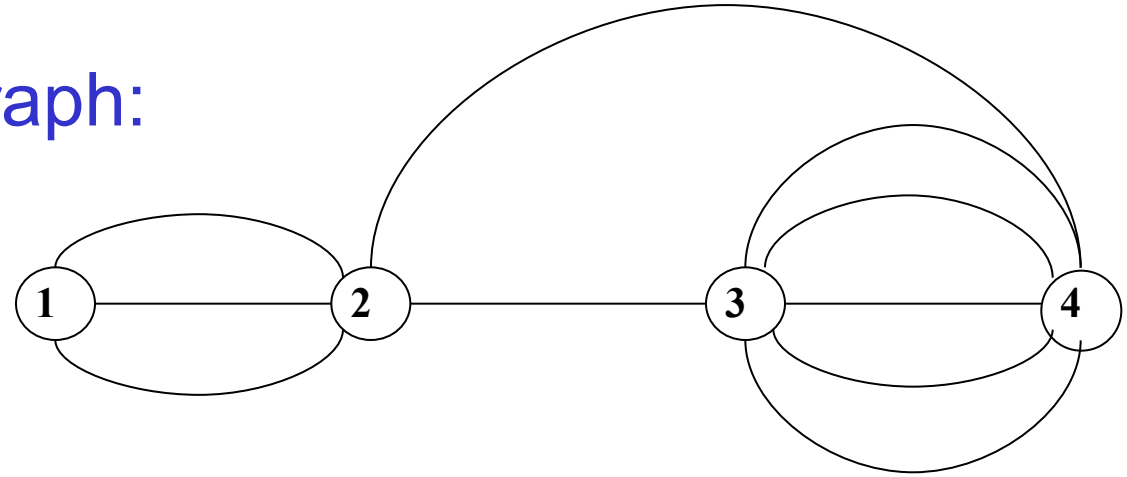
(die Menge der durch 5 teilbaren natürlichen Zahlen, wenn wir Ziffernfolgen mit 0-Präfixen ebenfalls zulassen)

Definition: Zustandsdiagramm [1]

Wir haben vorhin ja schon ein Zustandsdiagramm als Darstellung für einen Automaten gesehen. Jetzt beschreiben wir so ein Zustandsdiagramm formell.

Ein Zustandsdiagramm ist ein gerichteter Graph mit Kantenbeschriftungen.

ein Graph:

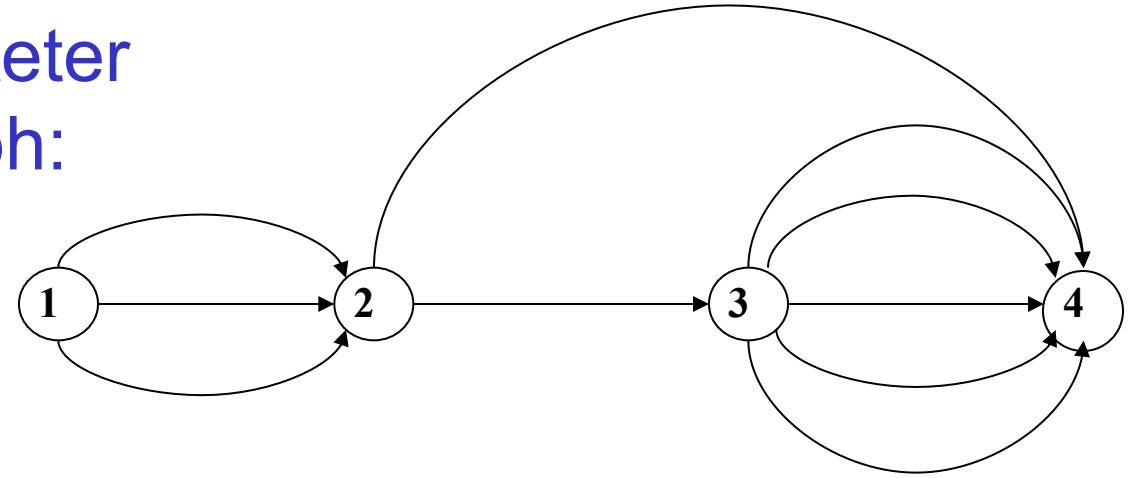


Definition: Zustandsdiagramm [1]

Wir haben vorhin ja schon ein Zustandsdiagramm als Darstellung für einen Automaten gesehen. Jetzt beschreiben wir so ein Zustandsdiagramm formell.

Ein Zustandsdiagramm ist ein gerichteter Graph mit Kantenbeschriftungen.

ein gerichteter Graph:

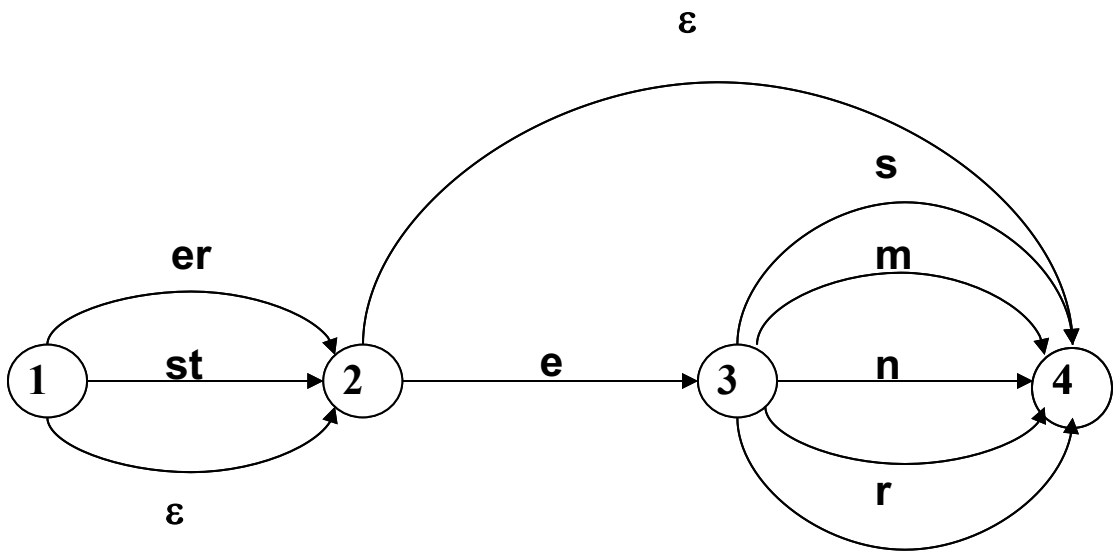


Definition: Zustandsdiagramm [1]

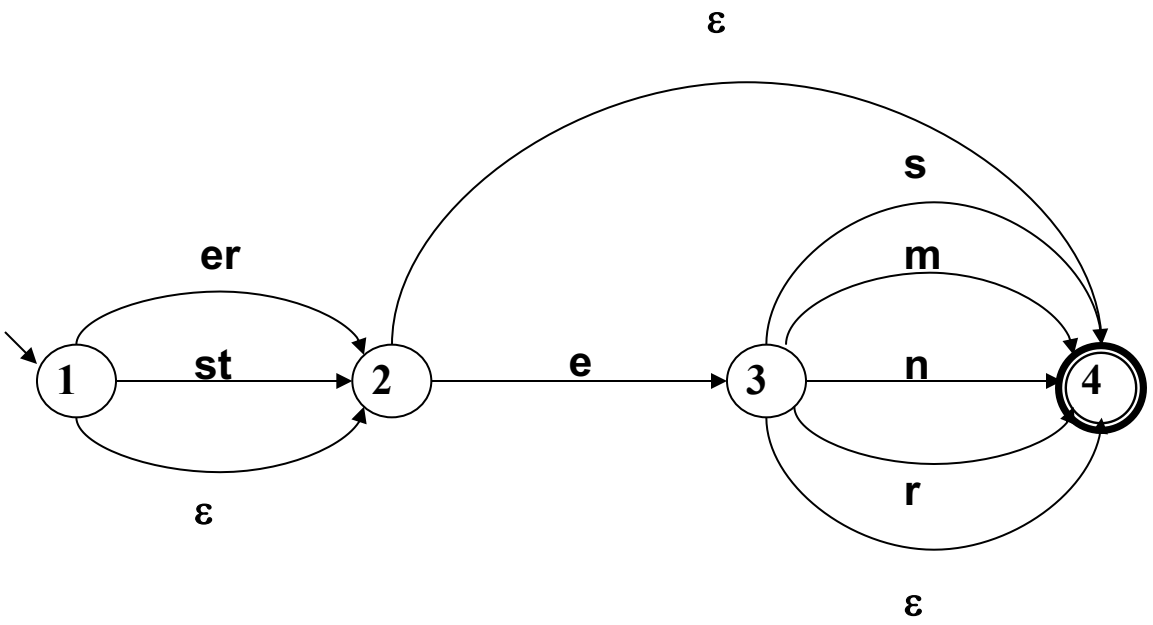
Wir haben vorhin ja schon ein Zustandsdiagramm als Darstellung für einen Automaten gesehen. Jetzt beschreiben wir so ein Zustandsdiagramm formell.

Ein Zustandsdiagramm ist ein gerichteter Graph mit Kantenbeschriftungen.

ein gerichteter Graph mit Kantenbe-schriftungen:



Ein Zustandsdiagramm hat Start und Endzustände.



Wie kann das komplette Zustandsdiagramm formell beschrieben werden? (Ohne Malen?)

Definition Zustandsdiagramm

Ein Zustandsdiagramm besteht aus

- Knoten (Zuständen) (im Beispiel: 1,2,3,4)
- davon ein Startknoten (1)
- einem oder mehreren Endknoten (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

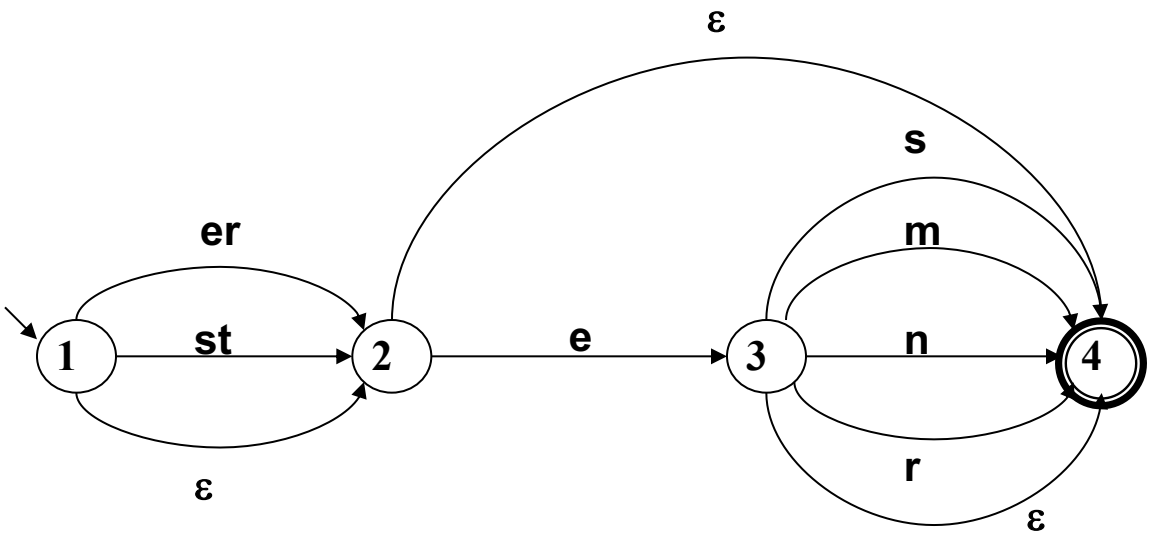
- K nicht-leere endliche Menge von Knoten (Zuständen)
- Σ nicht-leeres Alphabet
- $s \in K$ Startzustand
- $F \subseteq K$ Menge von Endzuständen
- $\Delta : K \times \Sigma^* \times K$ Menge von beschrifteten Kanten (Übergangsrelation)

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

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

  $A = <K,\Sigma,\Delta,s,F>$ 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 $A = <K,\Sigma,\Delta,s,F>$ 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.

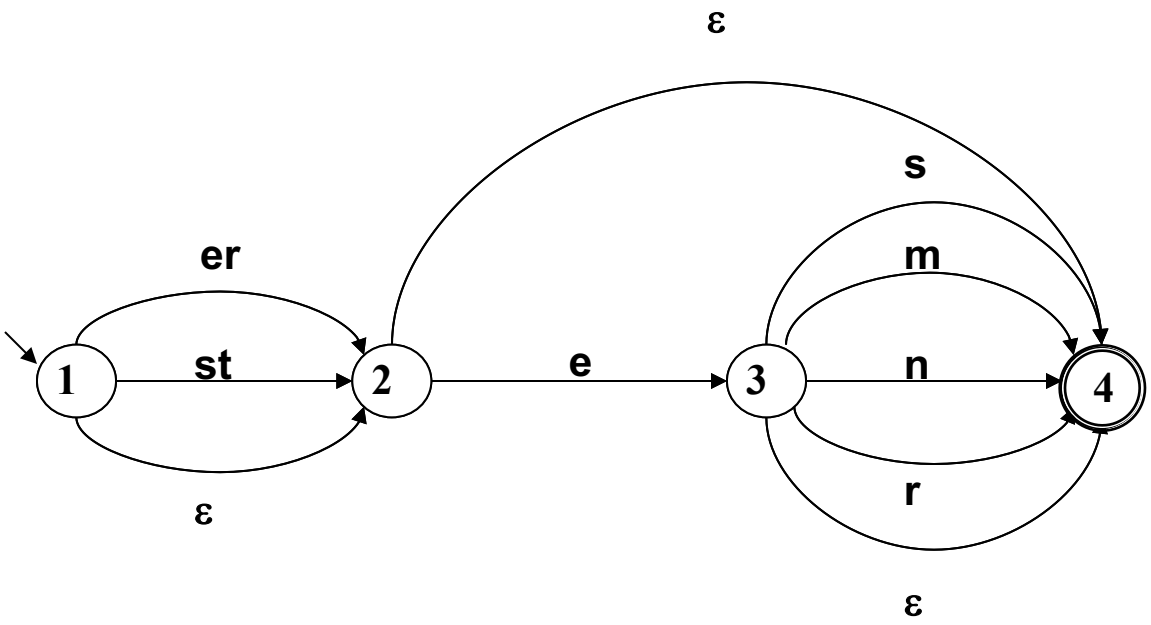
- Durch die Definitionen der letzten Folie (akzeptiertes Wort/definierte Sprache) wird diese Datenstruktur interpretiert: Wir verwenden Zustandsdiagramme, um die Zugehörigkeit von Symbolketten zu Sprachen zu definieren und zu testen.

- Hinzu kommen muss ein handhabbares Verfahren, ein Algorithmus, um den Zugehörigkeitstest tatsächlich durchzuführen.

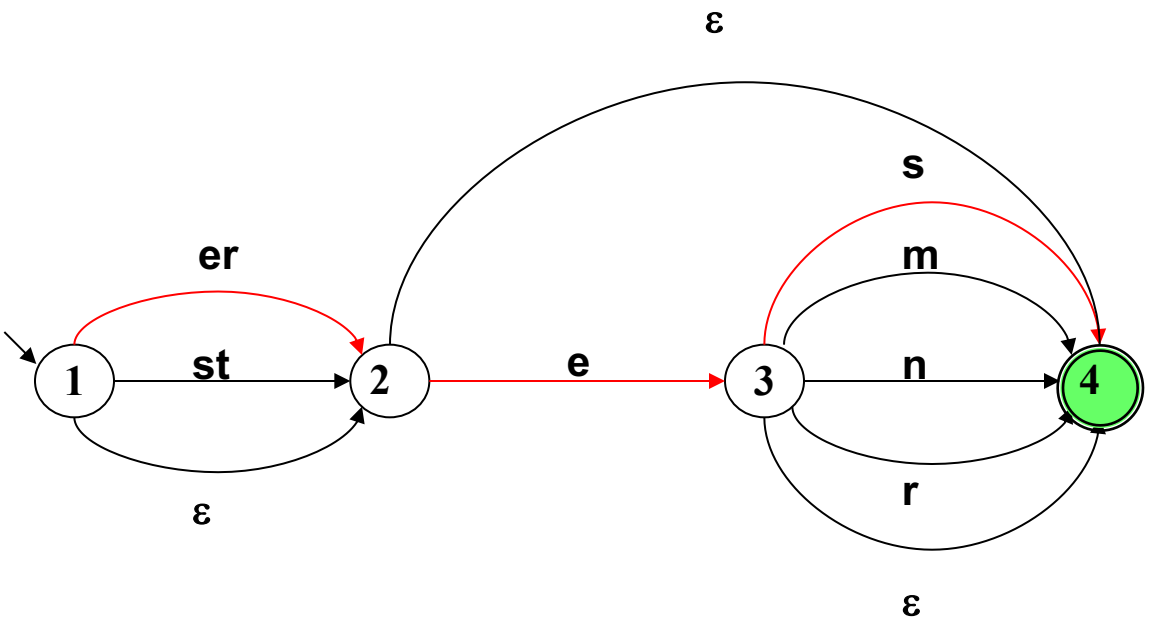
Inhalt heute: Automaten

1. Morphologie
2. Automaten (Intuition)
3. Automaten (Formalisierung)
4. Algorithmen: Suche in Automaten
5. Deterministische vs. Nicht-deterministische Automaten

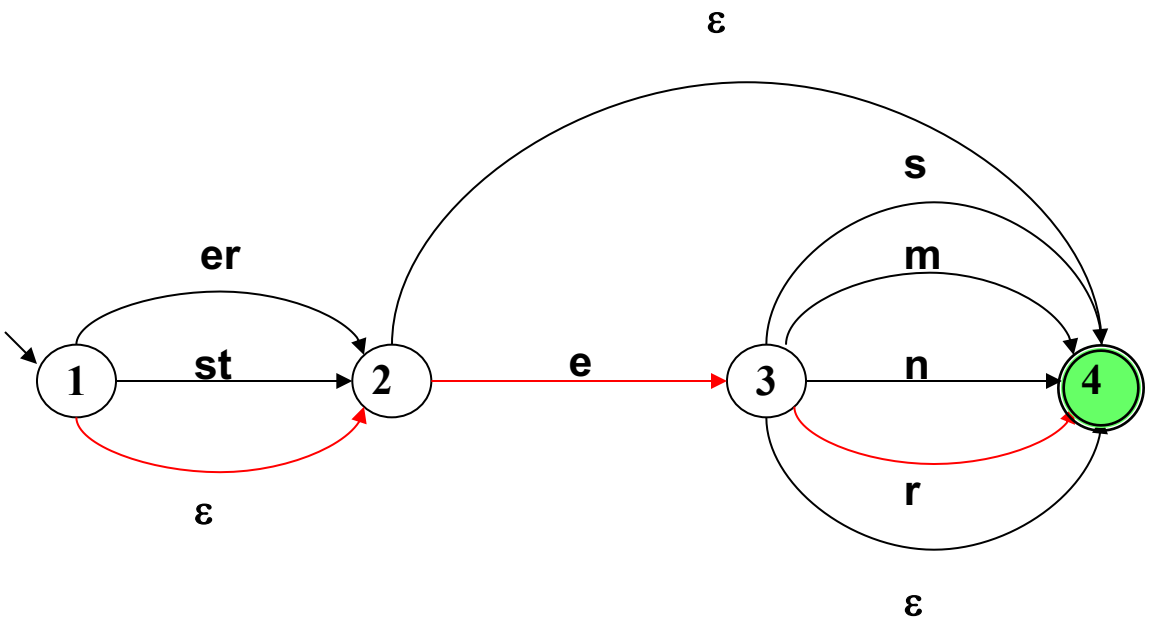
Ein Suchproblem



Ein Weg durchs Diagramm



Ein alternativer Weg durchs Diagramm



Ein Suchproblem

- Das Diagramm erlaubt typischerweise an einem Knoten/ einem Zustand **mehrere Übergänge** bei derselben Eingabe (deshalb „**nicht-deterministisch**“).

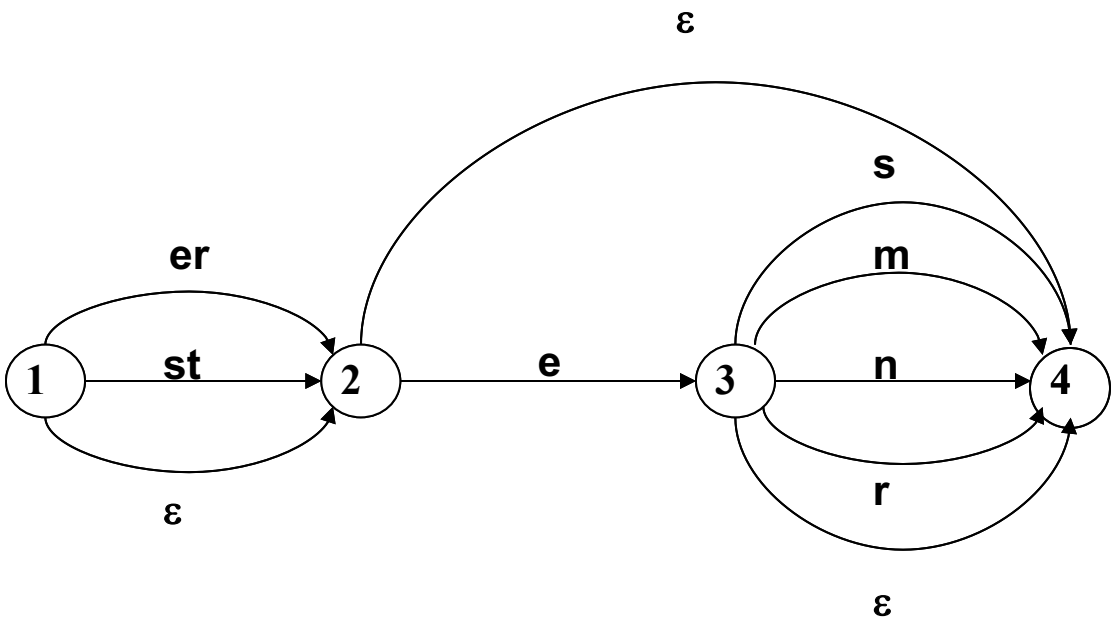
- Die **zufällige Wahl** einer Kante kann sich erst viel später als falsch herausstellen. Wir wissen nicht, ob tatsächlich alle möglichen Wege ausprobiert wurden.

- Wir benötigen ein Verfahren, das uns die **vollständige** Suche garantiert.

Herausforderung Suche:

Wie kann man systematisch alle Möglichkeiten durchsuchen?

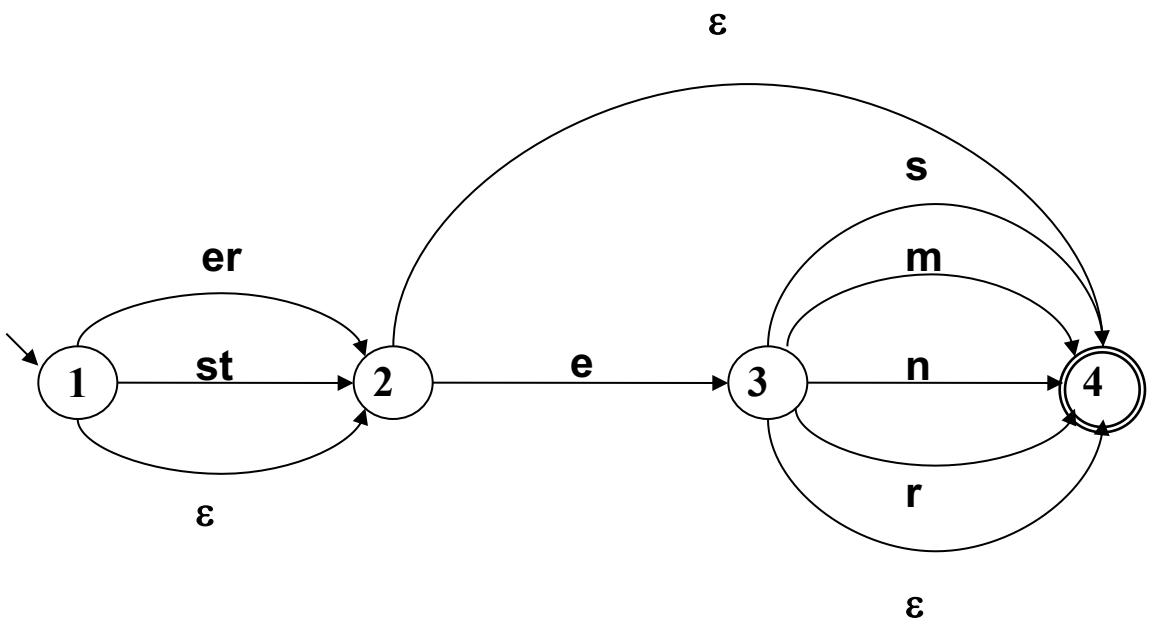
Überlegen und diskutieren Sie mit Ihrem Nachbarn.



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!).

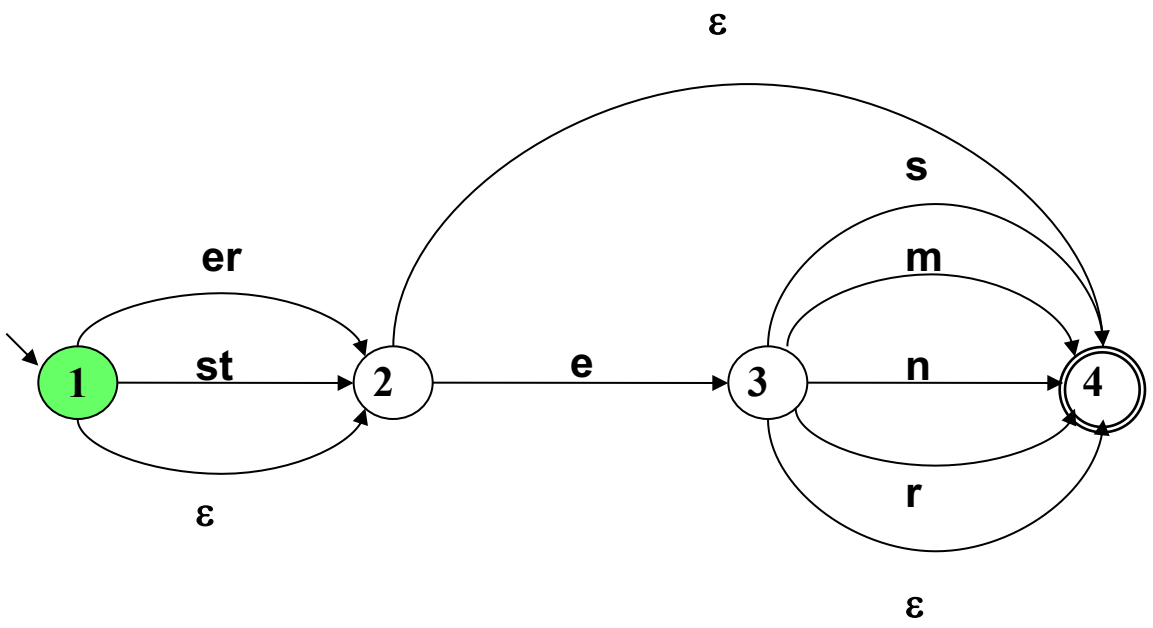
Initialisierung der Agenda



Eingabewort:

Agenda: 1 -- klein eres

Beispiel: klein**eres**



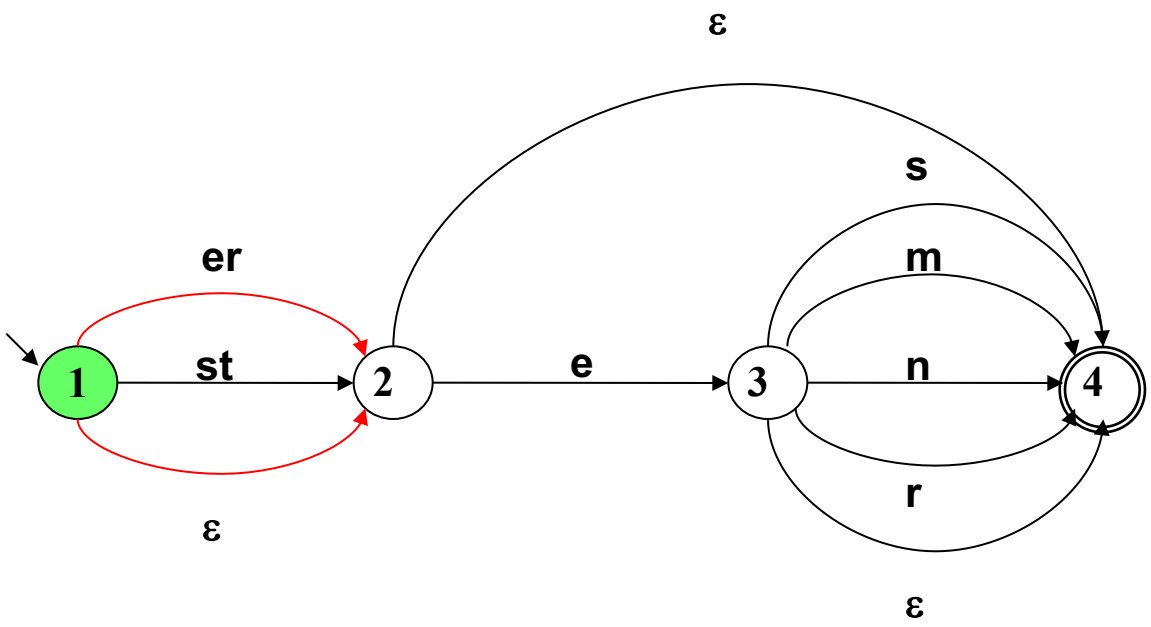
Welche Kanten können wir beschreiten?

Wir müssen uns alle begehbaren Kanten merken!

Eingabewort: klein eres

Agenda: _____

Pfadsuche: Generiere neue Aufgaben



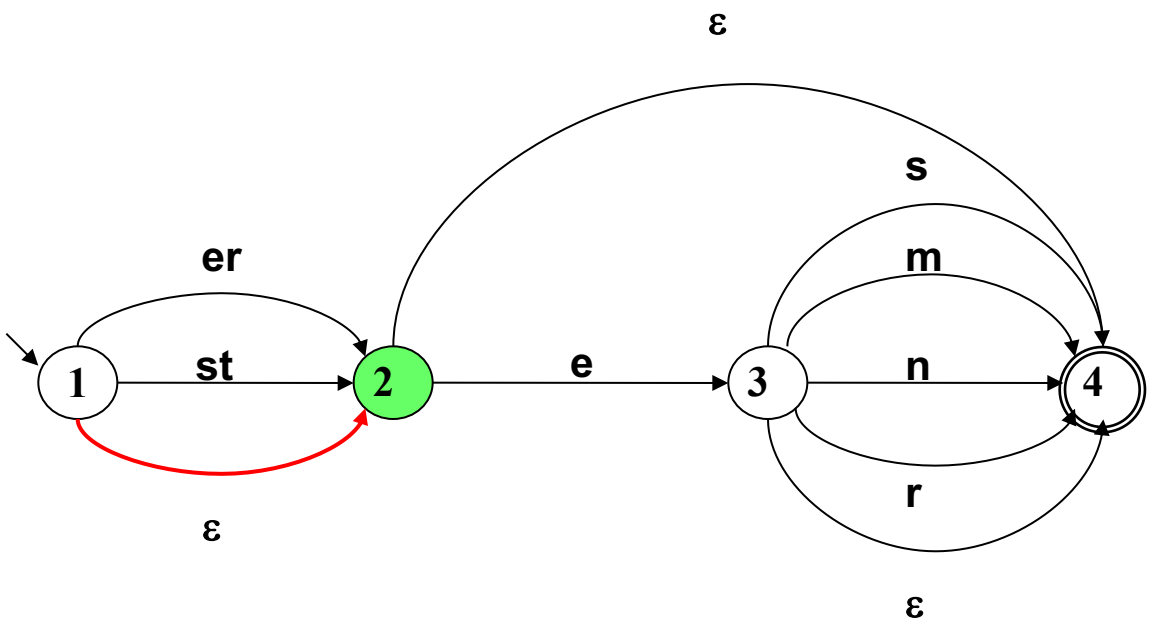
Wir nehmen eine der abgelegten Kanten von der Agenda.

Eingabewort:

Agenda:

2 -- klein eres
2 -- klein eres

Nimm Aufgabe von der Agenda und führe aus

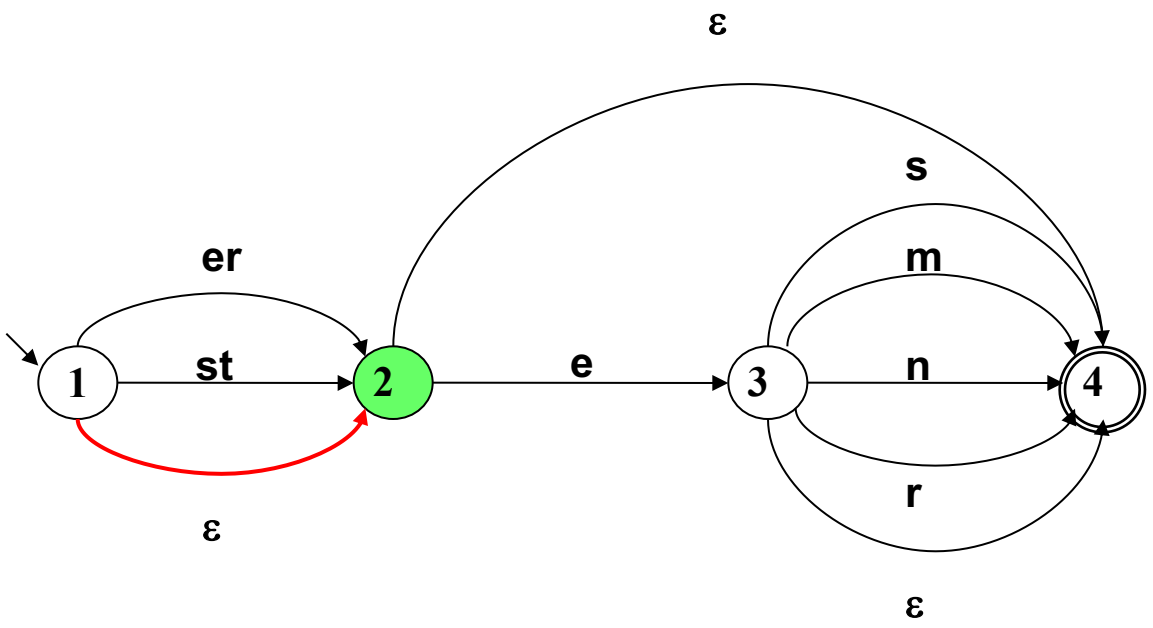


Wir führen die Aufgabe aus!

Eingabewort: klein eres

Agenda: 2 -- klein eres

Nimm Aufgabe von der Agenda und führe aus



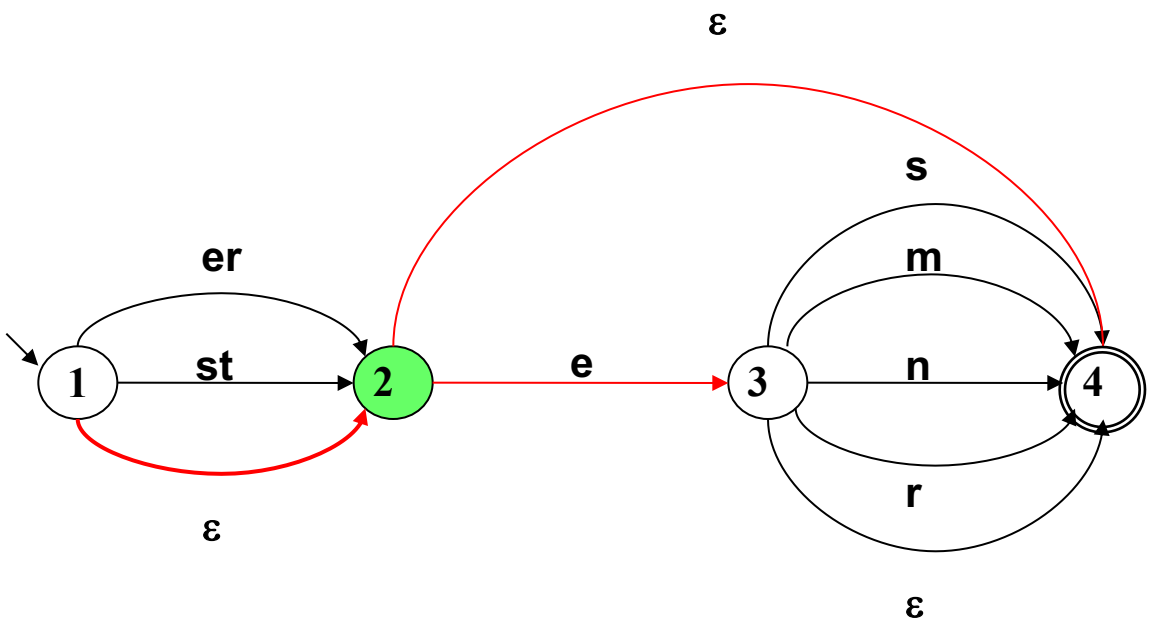
Welche Kanten können wir beschreiten?

Wir müssen uns alle begehbaren Kanten merken!

Eingabewort: klein eres

Agenda: 2 -- klein eres

Pfadsuche: Generiere neue Aufgaben



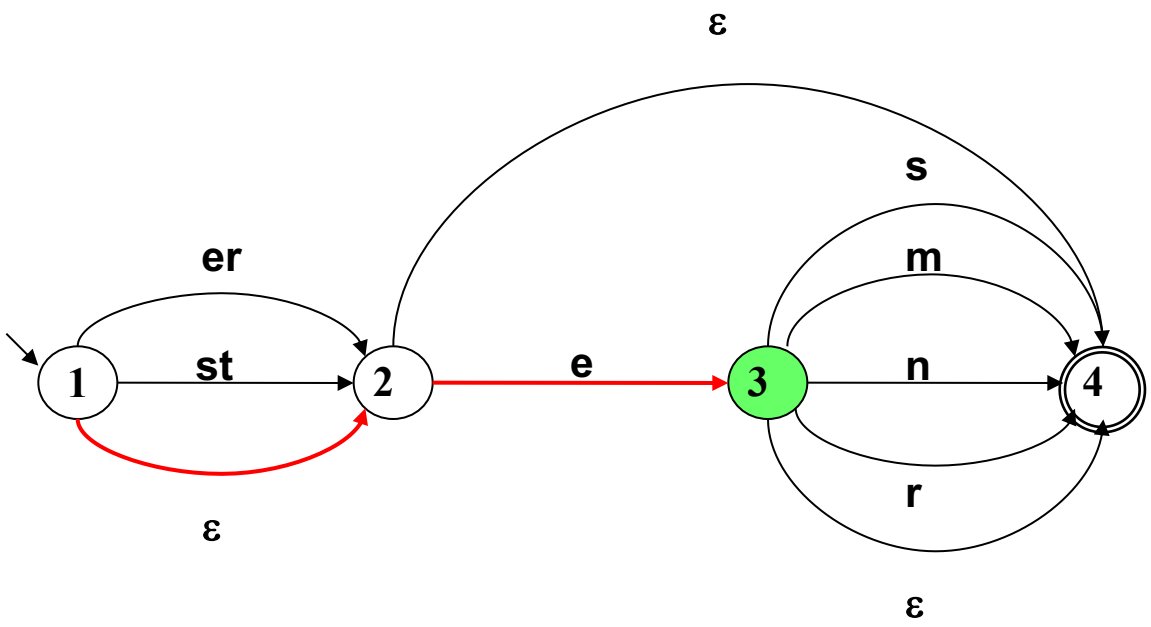
Wir nehmen eine der abgelegten Kanten von der Agenda.

Eingabewort:

Agenda:

3 -- klein eres
4 -- klein eres
2 -- klein eres

Pfadsuche: Nimm Aufgabe von der Agenda



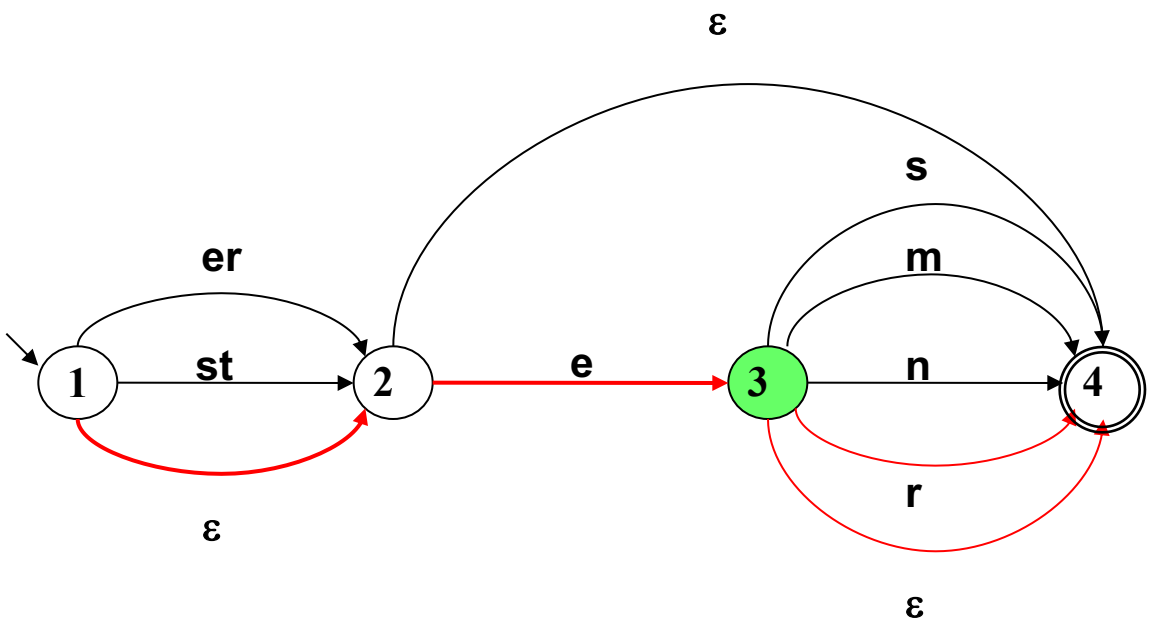
Wir führen die Aufgabe aus!

Eingabewort: klein eres

Agenda:

4 -- klein eres
2 -- klein eres

Pfadsuche: Generiere neue Aufgaben

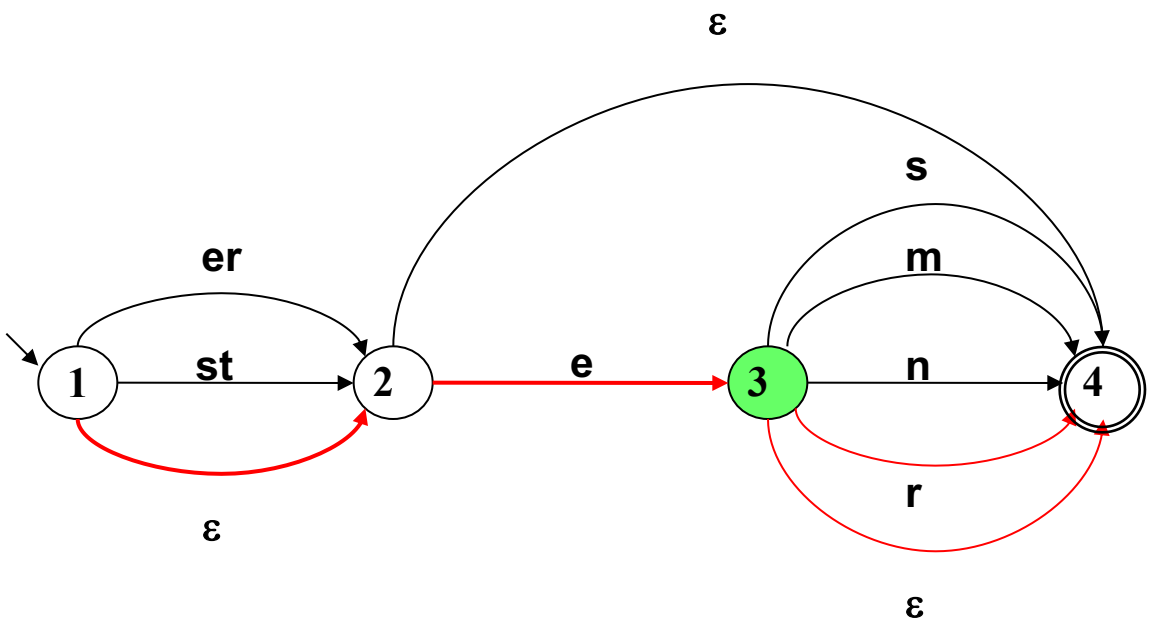


Eingabewort: klein eres

Agenda:

4 -- klein eres
2 -- klein eres

Pfadsuche: Generiere neue Aufgaben

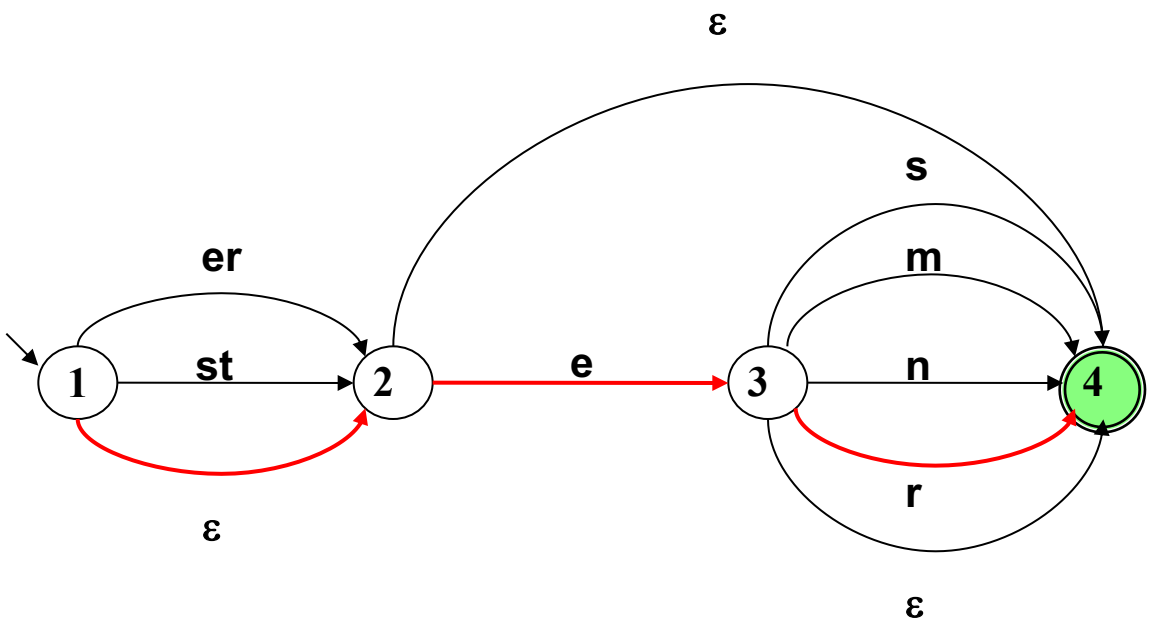


Eingabewort:

Agenda:

4 -- klein eres
4 -- klein eres
4 -- klein eres
2 -- klein eres

Pfadsuche: Nimm Aufgabe von der Agenda



Generiere!

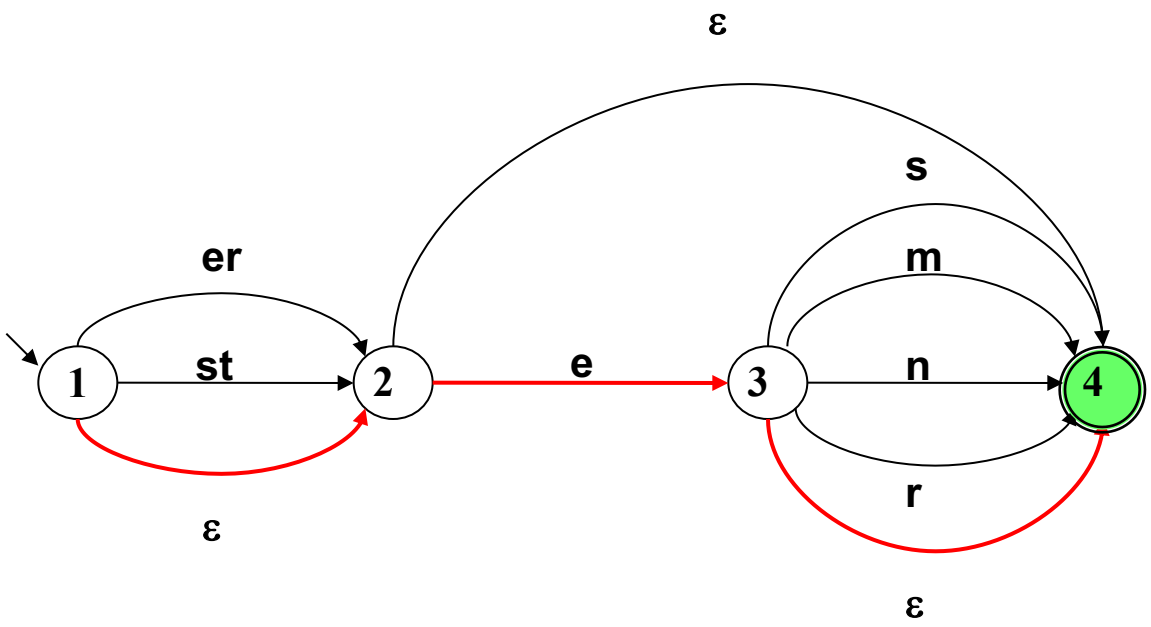
Keine neue Aufgabe!

Eingabewort: klein eres

Agenda:

4 -- klein eres
4 -- klein eres
2 -- klein eres

Pfadsuche: Nimm Aufgabe von der Agenda
Keine neue Aufgabe!

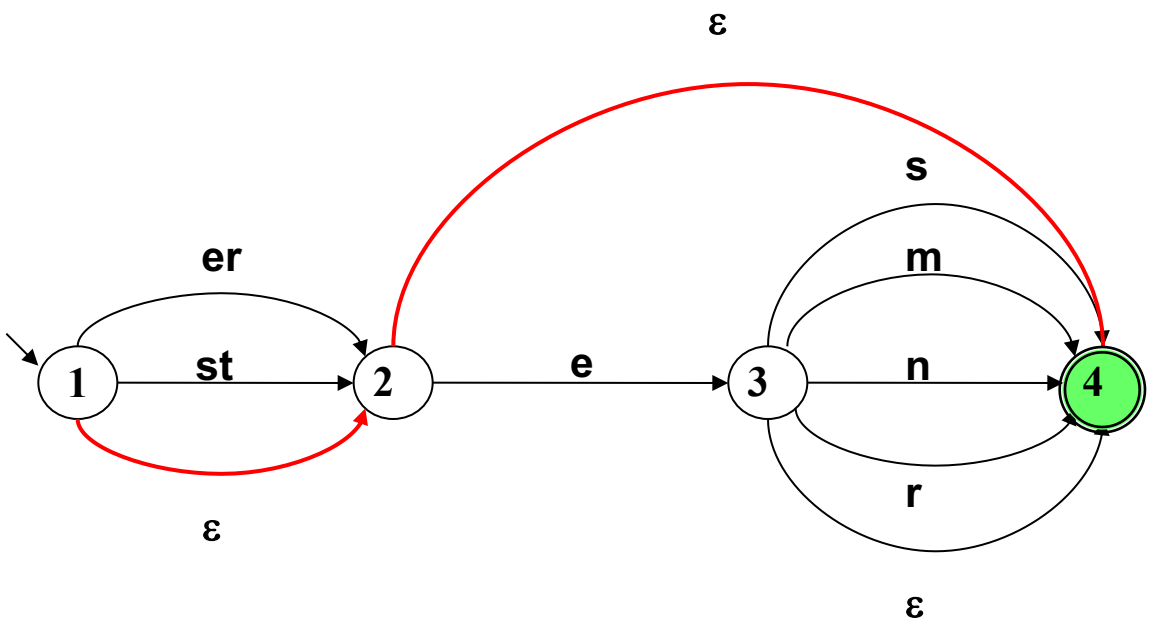


Eingabewort: klein eres

Agenda:

4 -- klein eres
2 -- klein eres

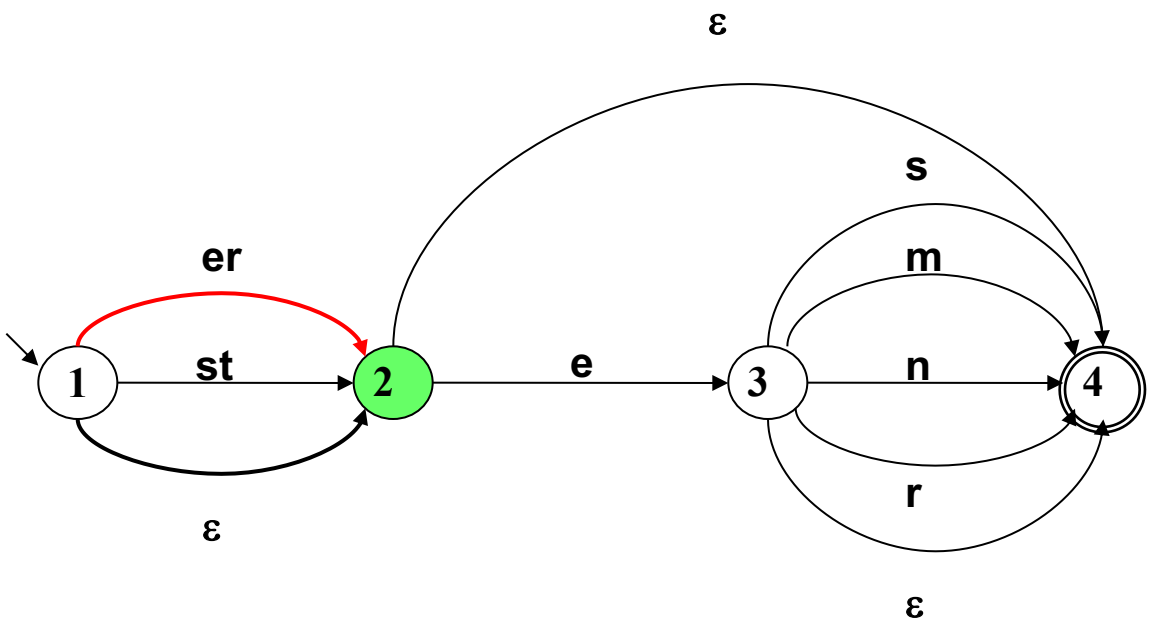
Pfadsuche: Nimm Aufgabe von der Agenda
Keine neue Aufgabe!



Eingabewort: klein eres

Agenda: 2 -- klein eres

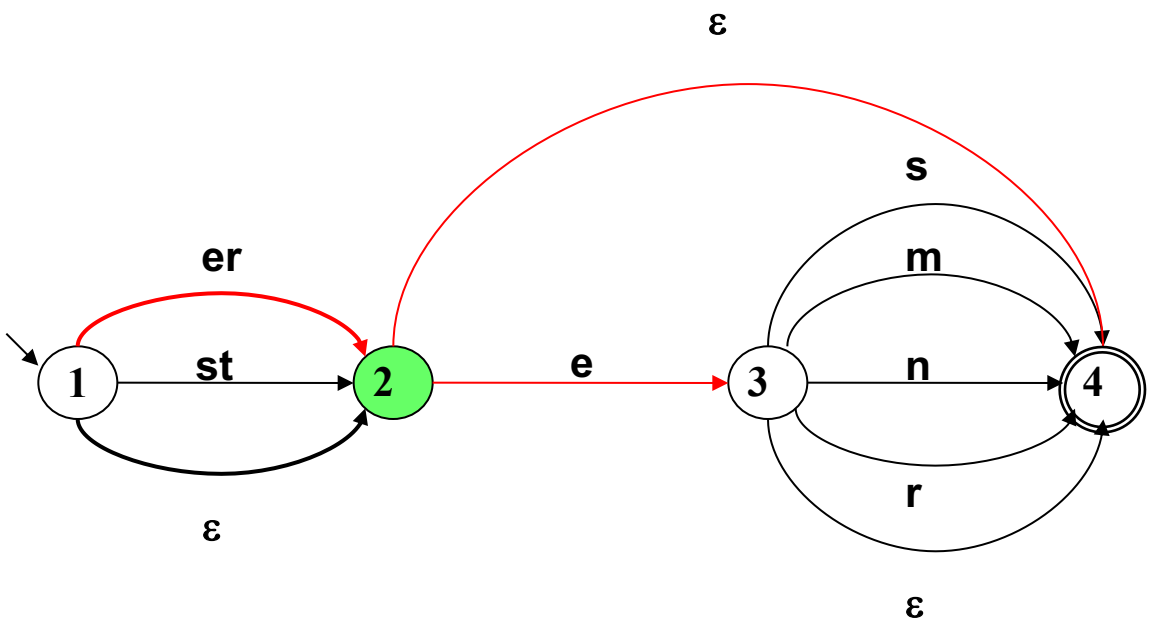
Pfadsuche: Nimm Aufgabe von der Agenda
Backtracking!



Eingabewort: klein eres

Agenda: _____

Pfadsuche: Generiere Aufgaben

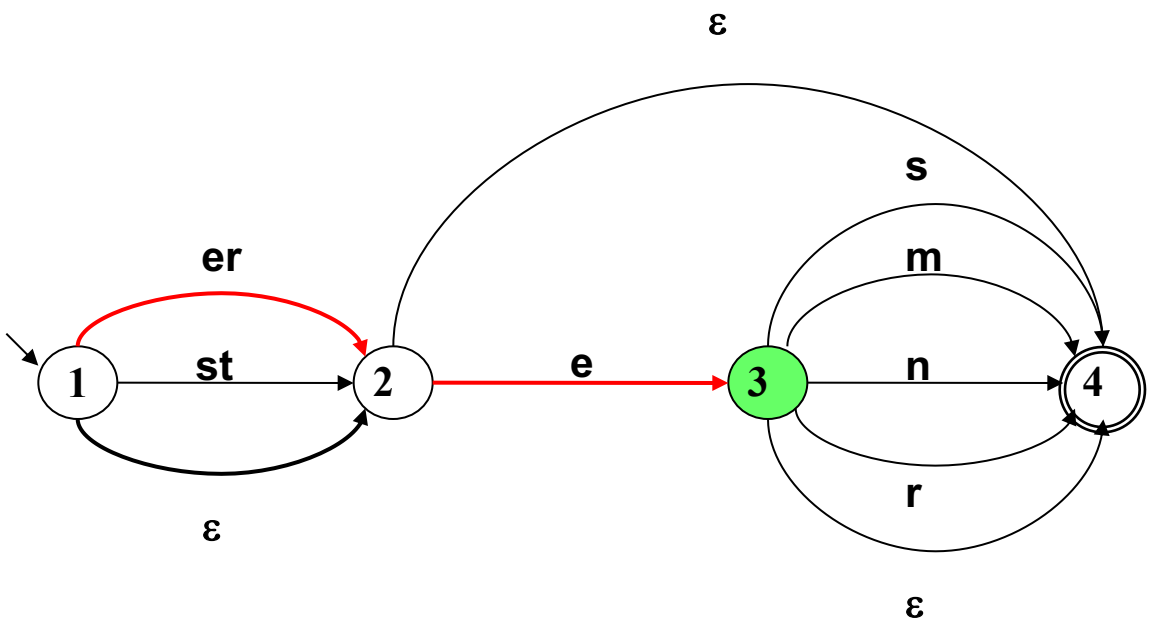


Eingabewort: klein eres

Agenda:

3 -- klein eres
4 -- klein eres

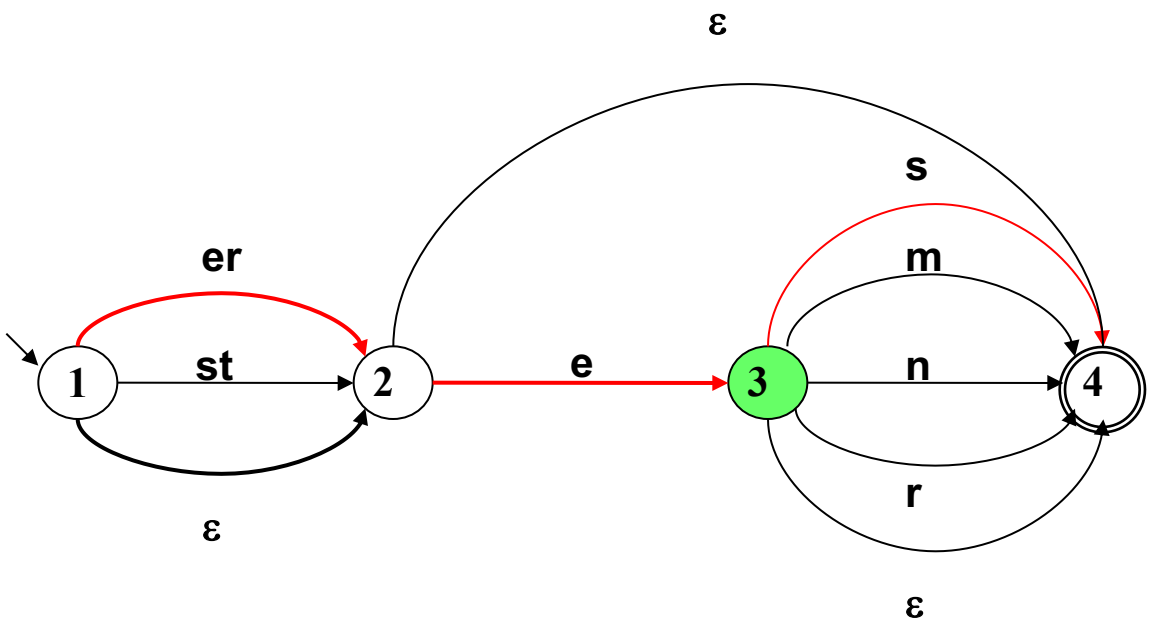
Pfadsuche: Nimm Aufgabe von der Agenda



Eingabewort: klein eres

Agenda: 4 -- klein eres

Pfadsuche: Generiere Aufgabe



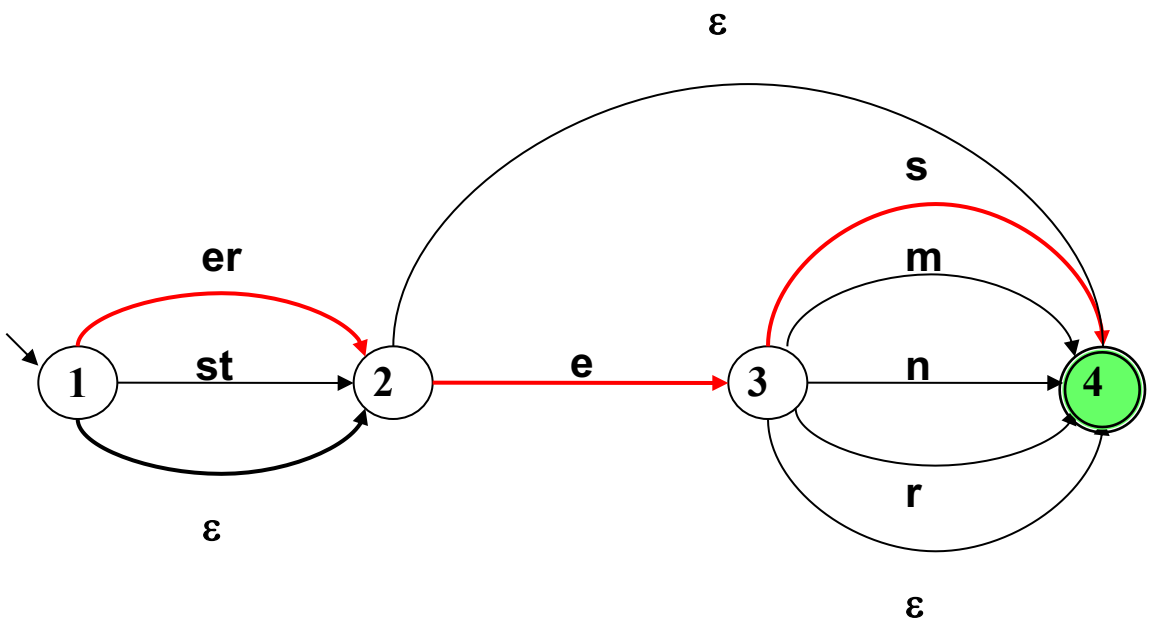
Eingabewort: klein eres

Agenda:

4 -- klein eres_

4 -- klein eres

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



Eingabewort: klein eres_

Agenda: 4 -- klein eres

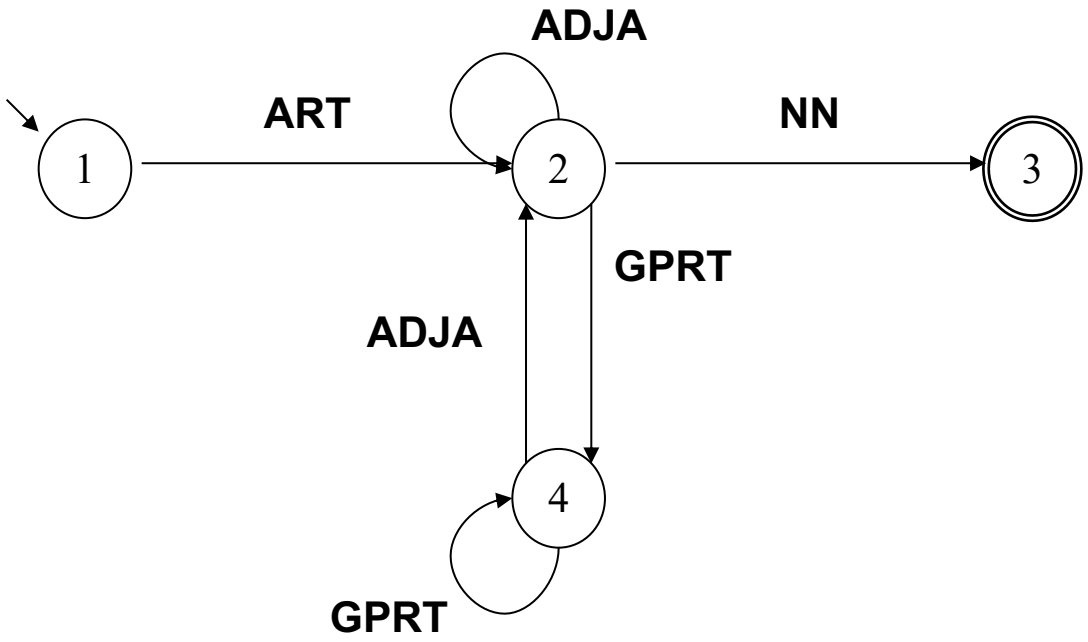
Anmerkung zum Pfadsuche-Algorithmus

- 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 nur dann *vollständig*, 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

Inhalt heute: Automaten

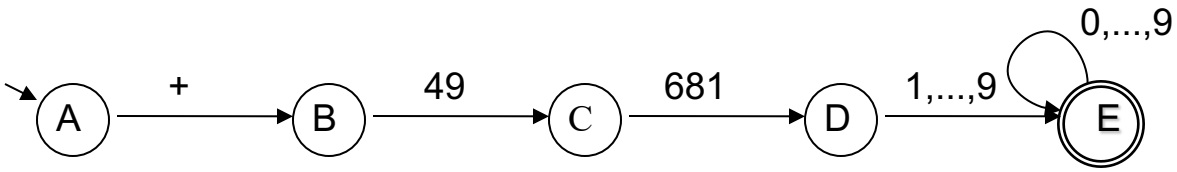
1. Morphologie
2. Automaten (Intuition)
3. Automaten (Formalisierung)
4. Algorithmen: Suche in Automaten
5. Deterministische vs. Nicht-deterministische Automaten
6. Anforderungen an Morphologiesysteme in der Praxis

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.

Nicht-deterministische und deterministische endliche Automaten

- NEA erlaubt beliebige Worte (incl. □) 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)

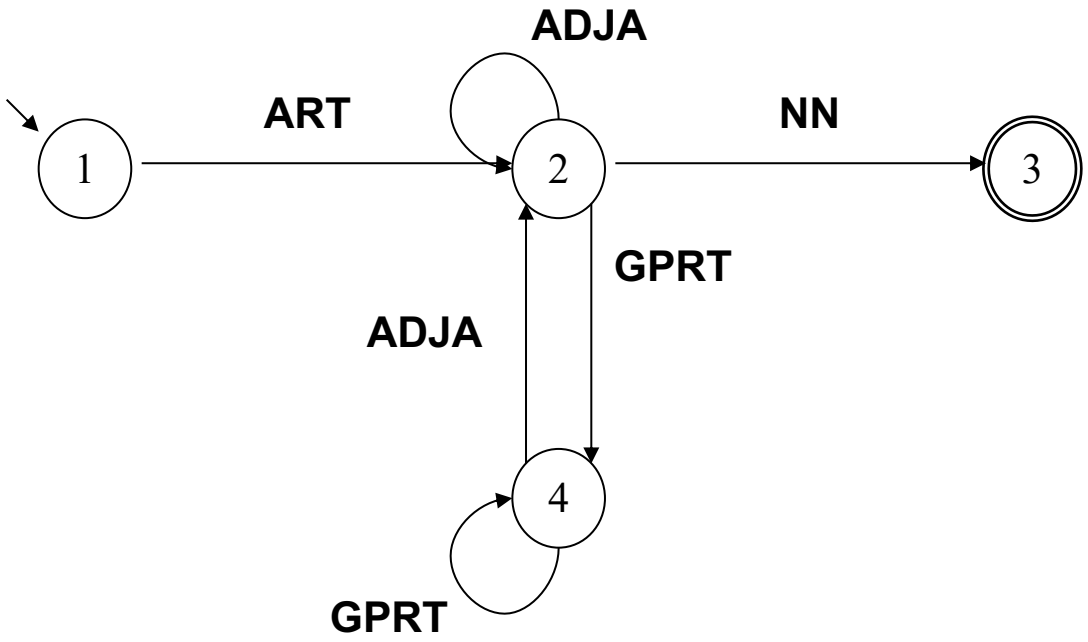
Definition: Deterministischer Endlicher Automat

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: $\delta(1,ART) = 2$
  $\delta(2, ADJA) = 2$
  $\delta(2,NN) = 3$
  $\delta(2,GPRT) = 4$
  ... ...

Beispiel: Übergangstabelle für δ

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

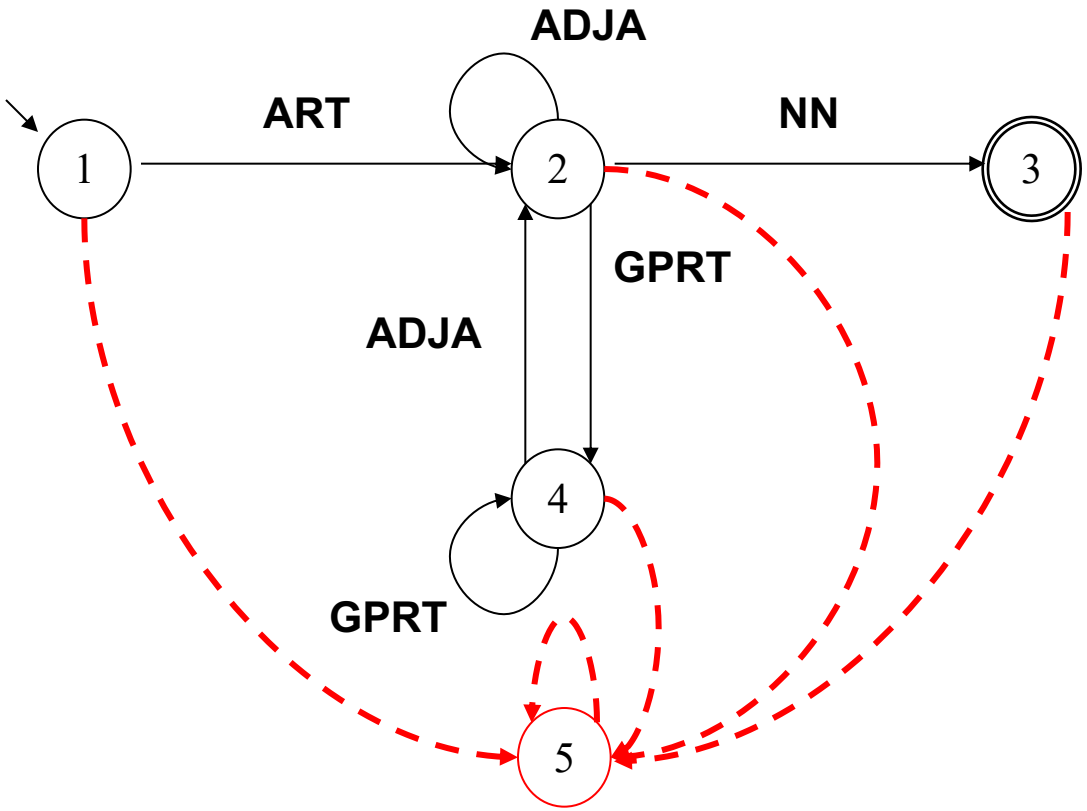
„Ein DEA hat zu jedem Zustand und jedem Symbol genau eine wegführende Kante“.
Deshalb kann man die Übergänge gut als Tabelle darstellen.

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 Beschreibungsinventar als NEAs.
- Frage: Ist deshalb die Ausdrucksstärke des DEA-Formalismus eingeschränkter als die von NEAs?
- Umgekehrt gefragt: Kann jede Sprache, die durch einen NEA beschrieben wird, auch durch einen DEA beschrieben werden?
- Die Antwort lautet: Ja!

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.

- Es gibt ein Konstruktionsverfahren, das es erlaubt, zu jedem NEA A einen DEA A' zu konstruieren, so dass L(A') = L(A).
- => Mathematische Grundlagen II

Inhalt heute: Automaten

1. Morphologie
2. Automaten (Intuition)
3. Automaten (Formalisierung)
4. Algorithmen: Suche in Automaten
5. Deterministische vs. Nicht-deterministische Automaten
6. Anforderungen an Morphologiesysteme in der Praxis

Anforderungen an Morphologiesysteme

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

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

Übergenerierung: Beispiele aus der Praxis

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

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

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 μs-Bereich pro Wortform
- Durch Vorverarbeitung, Zwischenspeichern von Analysen, Indexierung etc. lässt sich für größere Dokumente die Zeit pro Textwort weiter drücken.
- Das ist
  - exzellent für Online- und Offline-Rechtschreibkorrektur
  - akzeptabel für begrenzte Datensammlungen (große Textkorpora, Firmen-Intranet etc.)
  - zu langsam für allgemeine Websuche