

Durch NEA akzeptiertes Wort/definierte Sprache

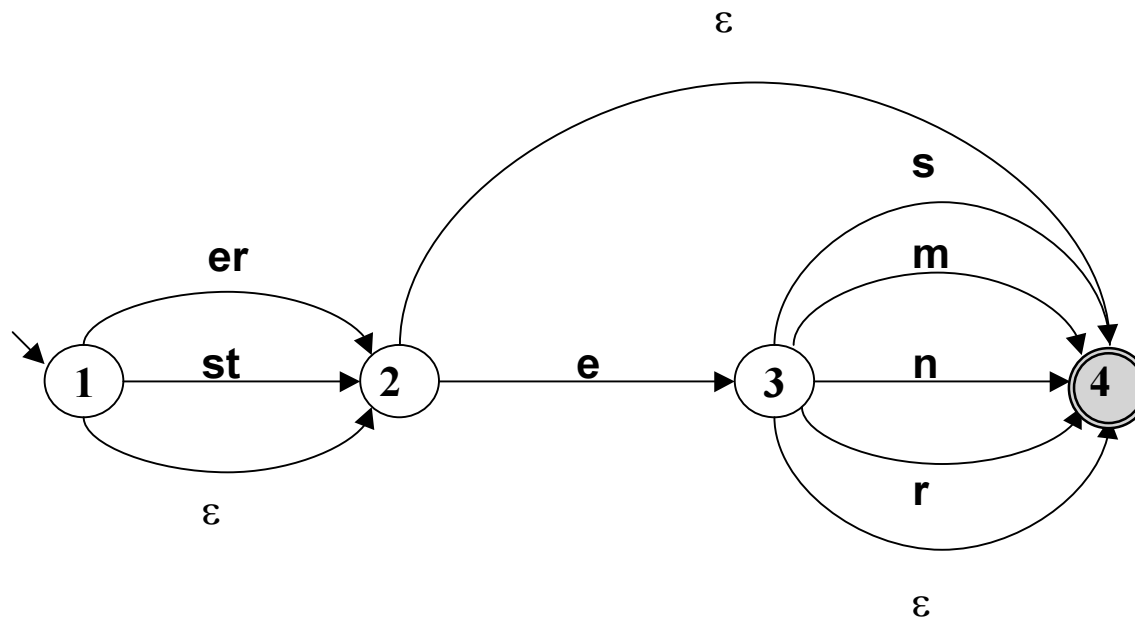
- Ein Wort $w \in \Sigma^*$ wird durch den NEA $A = \langle K, \Sigma, \Delta, s, F \rangle$ akzeptiert
gdw. es eine Folge von Kanten (einen Pfad durch den NEA)
 $\langle s, u_1, k_1 \rangle, \langle k_1, u_2, k_2 \rangle, \dots, \langle k_{n-1}, u_n, k_n \rangle$ gibt, sodass $k_n \in F$
und $u_1 u_2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

Eine allgemeine 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.

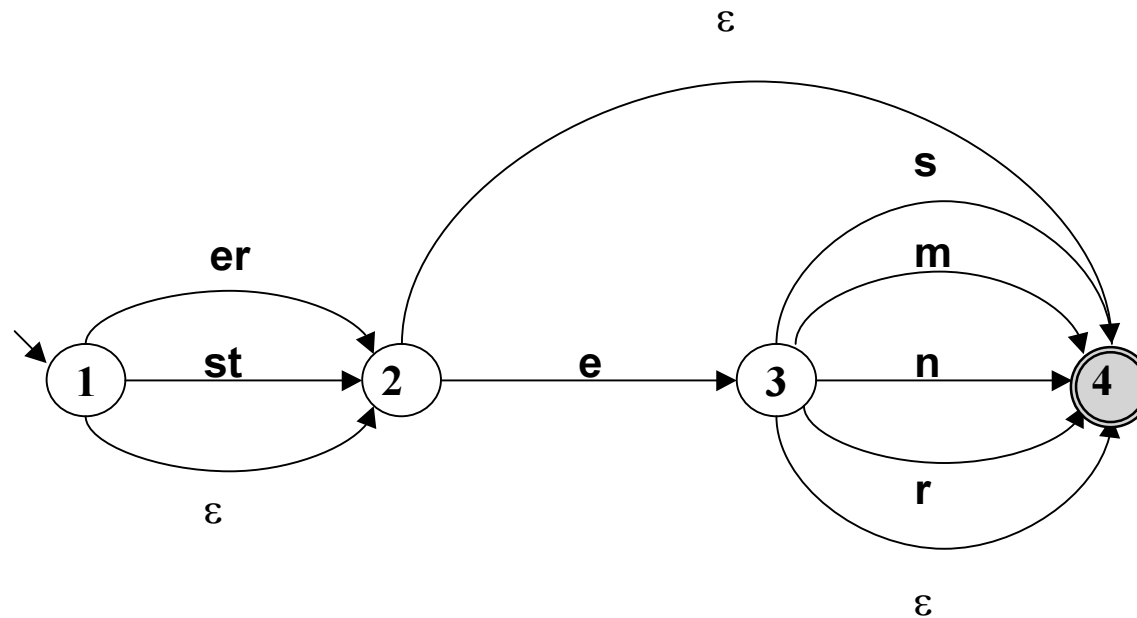
Diese drei Schritte sind für die Modellierung in der Computerlinguistik (und der Informatik) charakteristisch.

Ein Weg durchs Diagramm



klein eres_

Ein alternativer Weg durchs Diagramm



klein eres

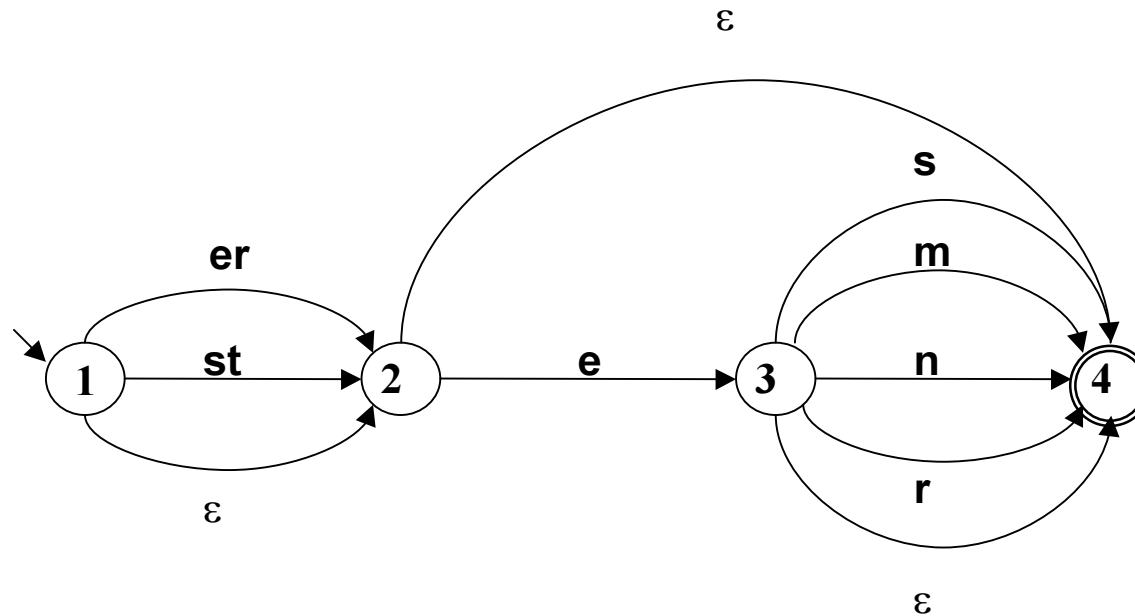
Das Problem

- 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 gleich vor, sondern legen die alternativen Resultatkonfigurationen – Zustand und Position im Wort – in einer Liste noch zu erledigender Teilaufgaben, der Agenda, ab.
- Dann nehmen wir einen Eintrag aus der Agenda, den letzten oder obersten, wenn wir uns die Agenda als Stapel („Stack“) vorstellen, betrachten ihn als neue aktuelle Konfiguration, testen mögliche nächste Schritte, legen die Resultate auf die Agenda usw. – bis wir die Eingabe erfolgreich abgearbeitet haben (akzeptieren!) oder die Agenda leer ist (zurückweisen!).

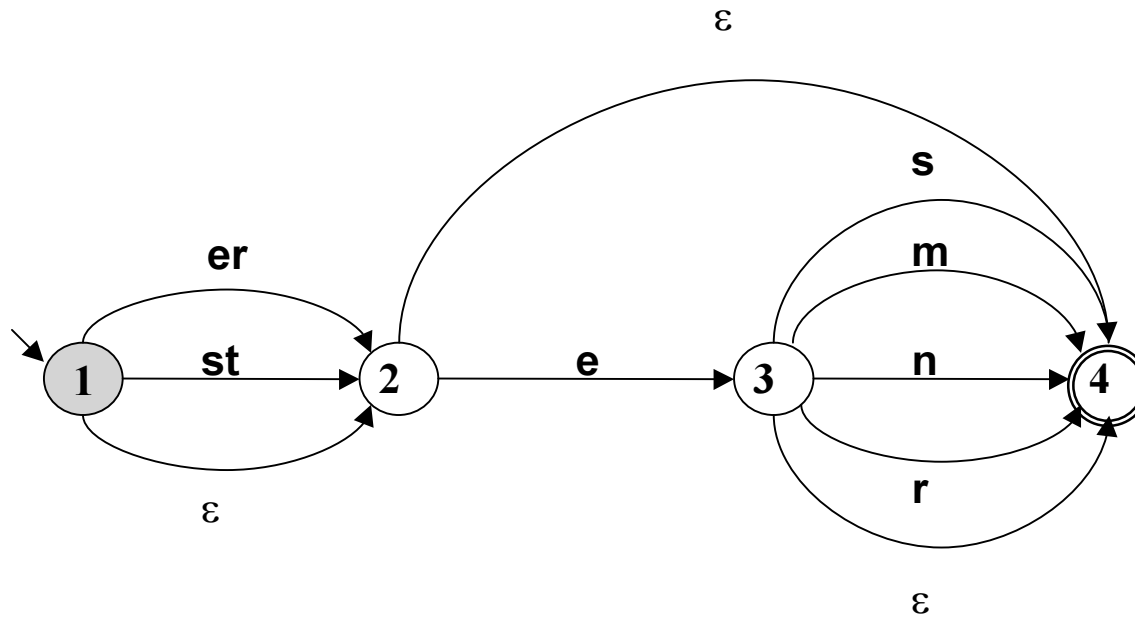
Pfadsuche: Startkonfiguration: Startzustand und Eingabewort auf der Agenda



Eingabewort:

Agenda: 1 -- klein eres

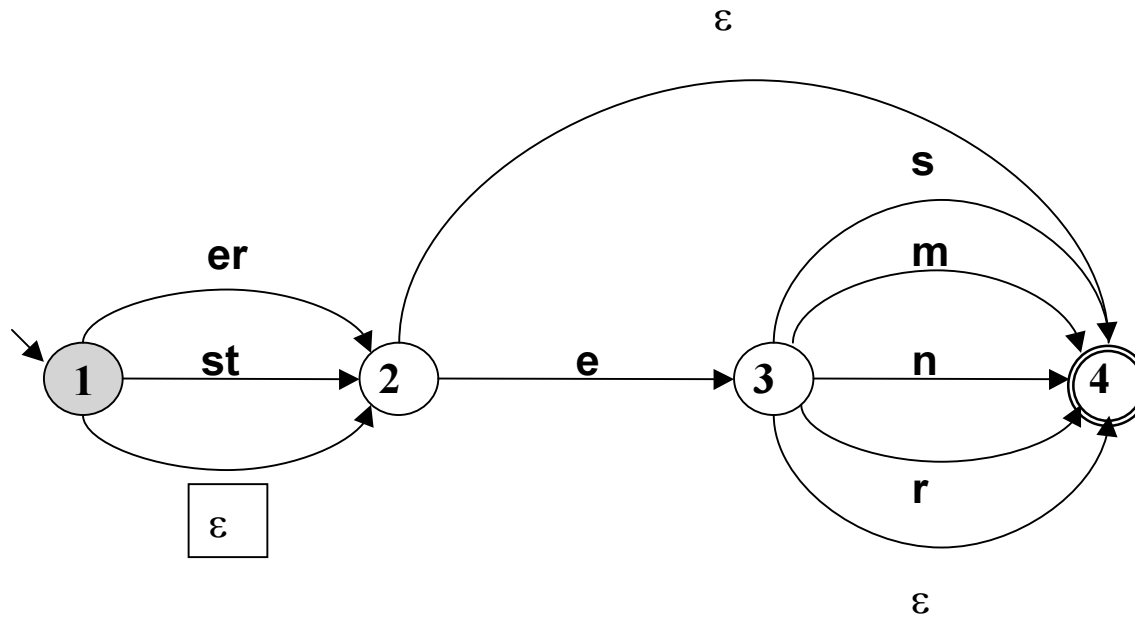
Pfadsuche: Nimm Aufgabe von der Agenda



Eingabewort: klein eres

Agenda: _____

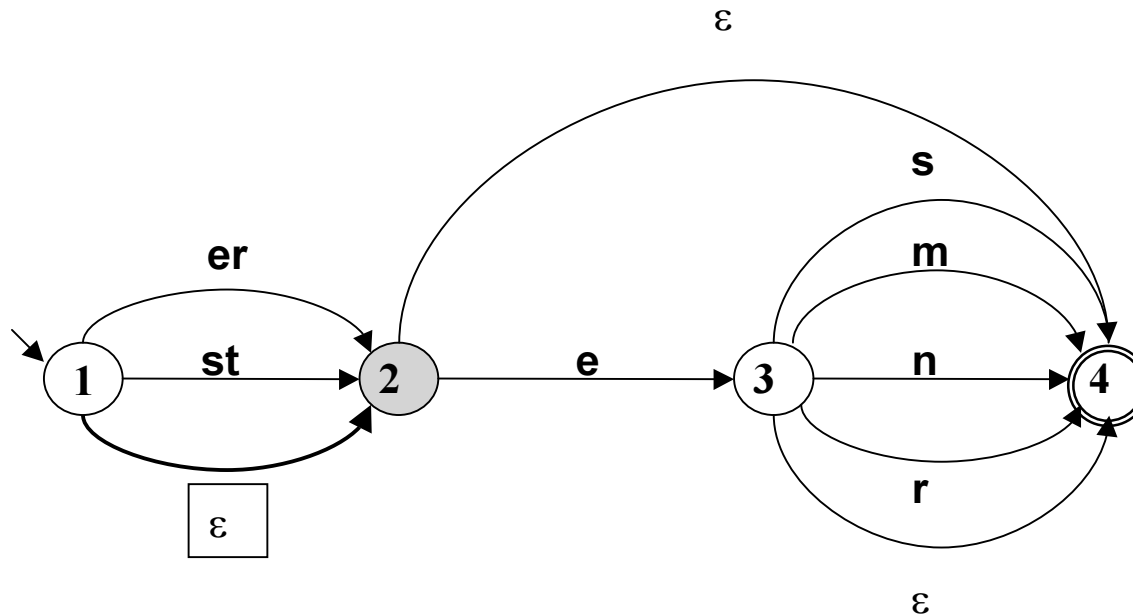
Pfadsuche: Generiere neue Aufgaben



Eingabewort: klein eres

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

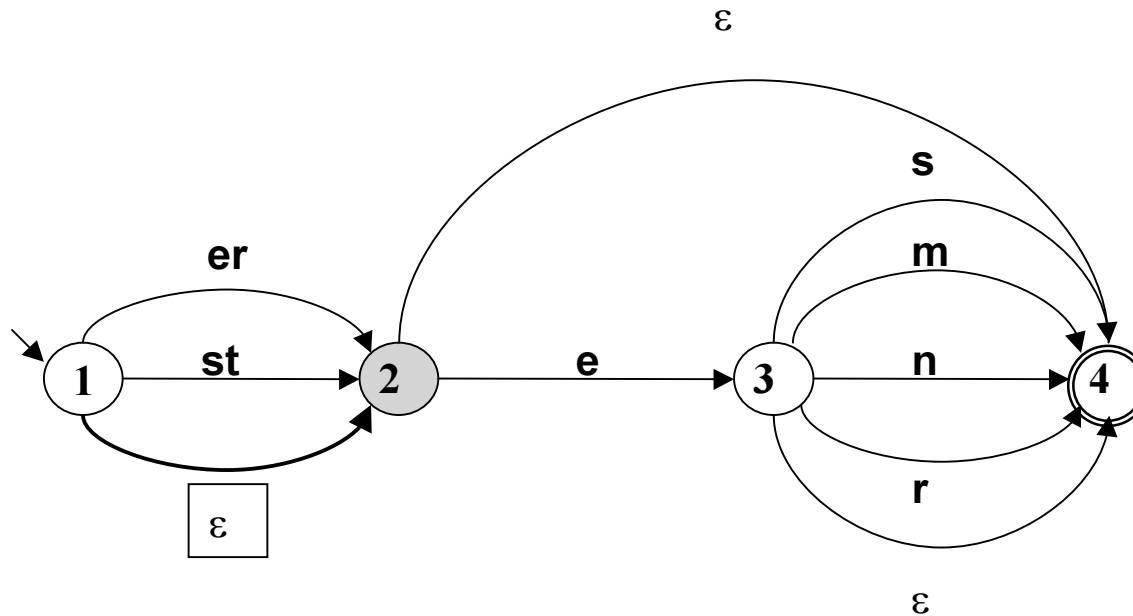
Pfadsuche: Nimm Aufgabe von der Agenda



Eingabewort: klein eres

Agenda: 2 -- klein eres

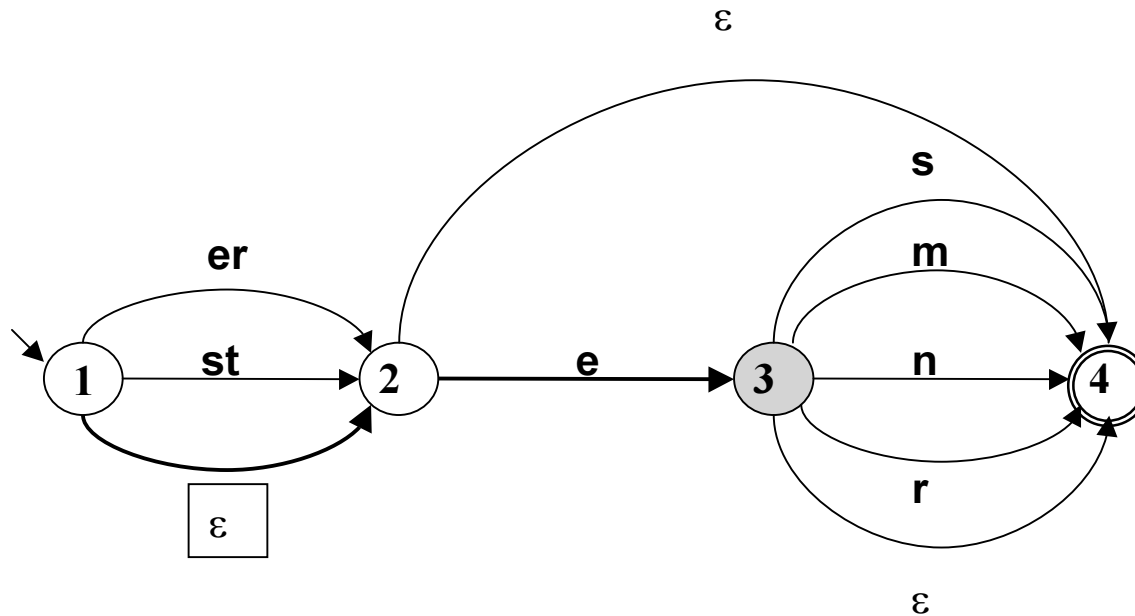
Pfadsuche: Generiere neue Aufgaben



Eingabewort: klein eres

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

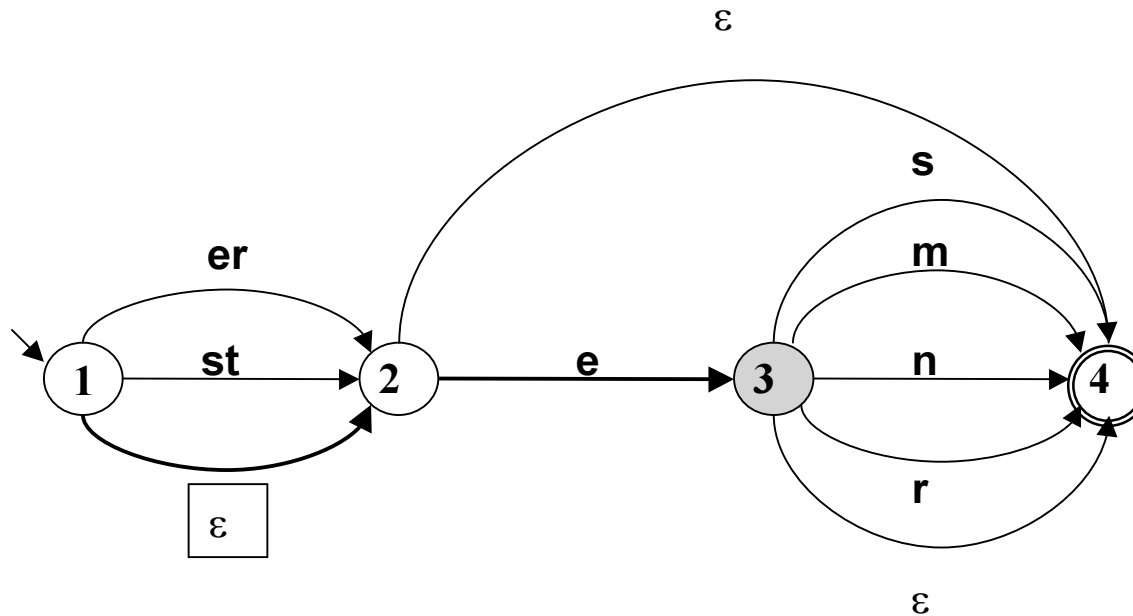
Pfadsuche: Nimm Aufgabe von der Agenda



Eingabewort: klein eres

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

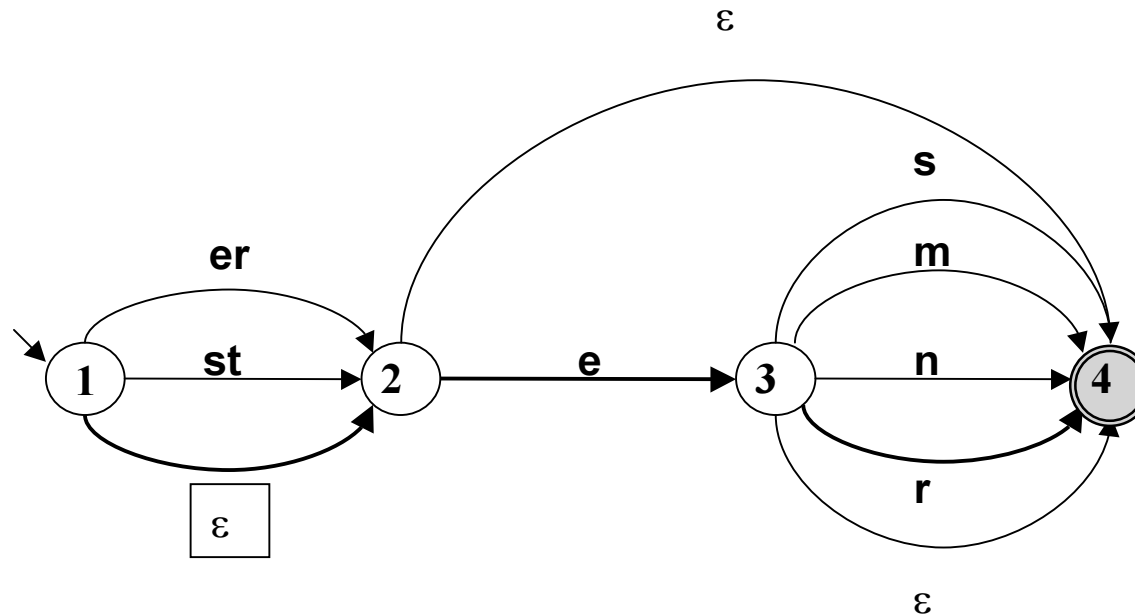
Pfadsuche: Generiere neue Aufgaben



Eingabewort: klein eres

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

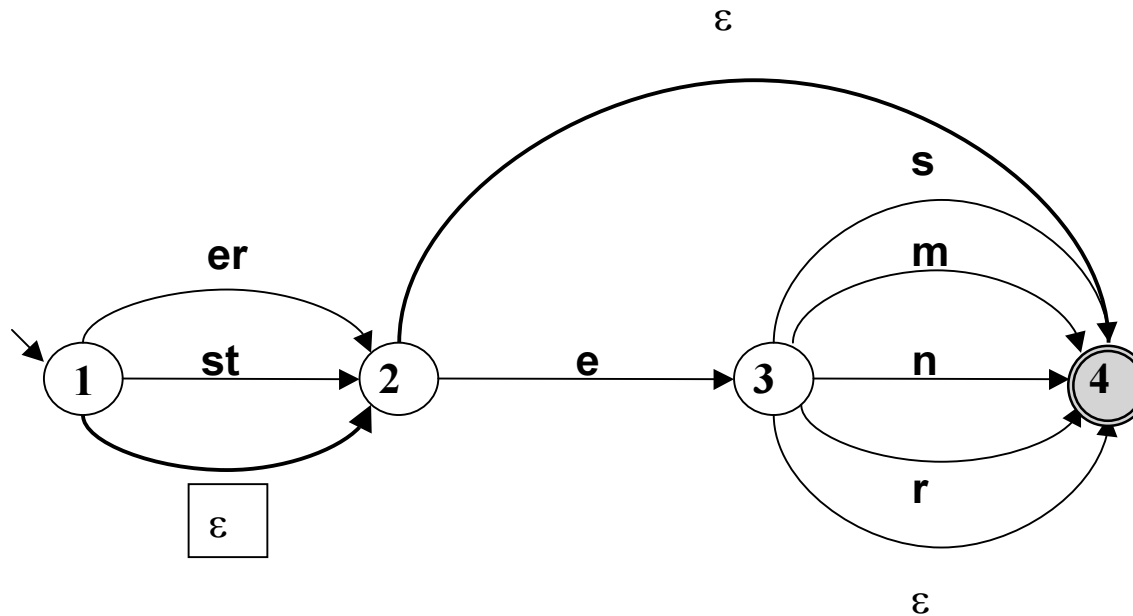
Pfadsuche: Nimm Aufgabe von der Agenda
Keine neue Aufgabe!



Eingabewort: klein eres

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

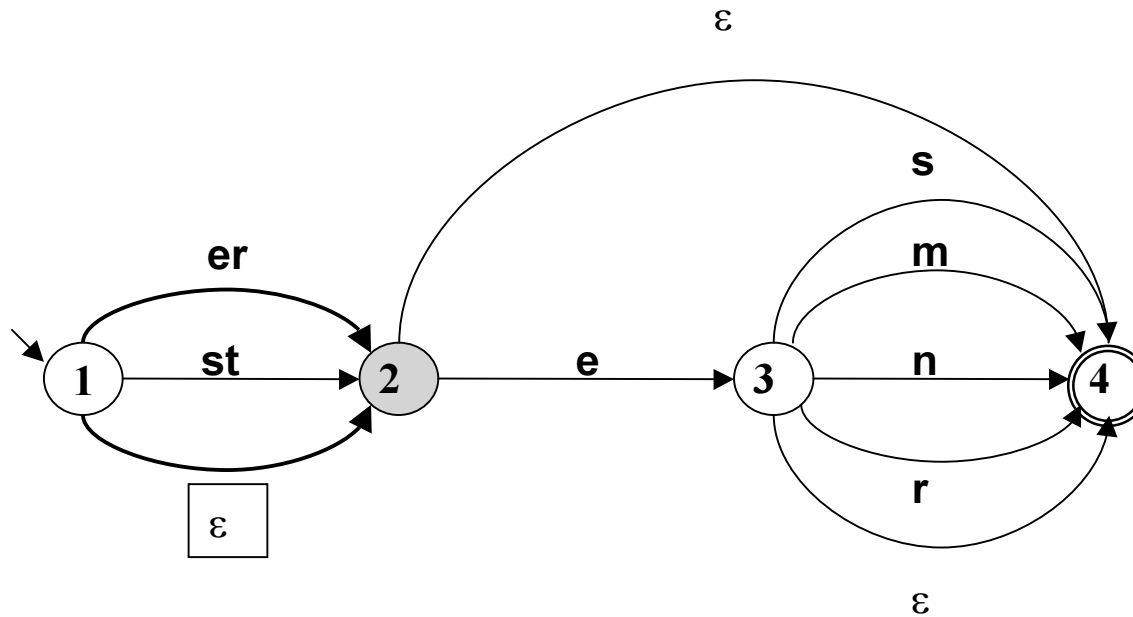
Pfadsuche: Nimm Aufgabe von der Agenda
Keine neue Aufgabe!



Eingabewort: klein eres

Agenda: 2 -- klein eres

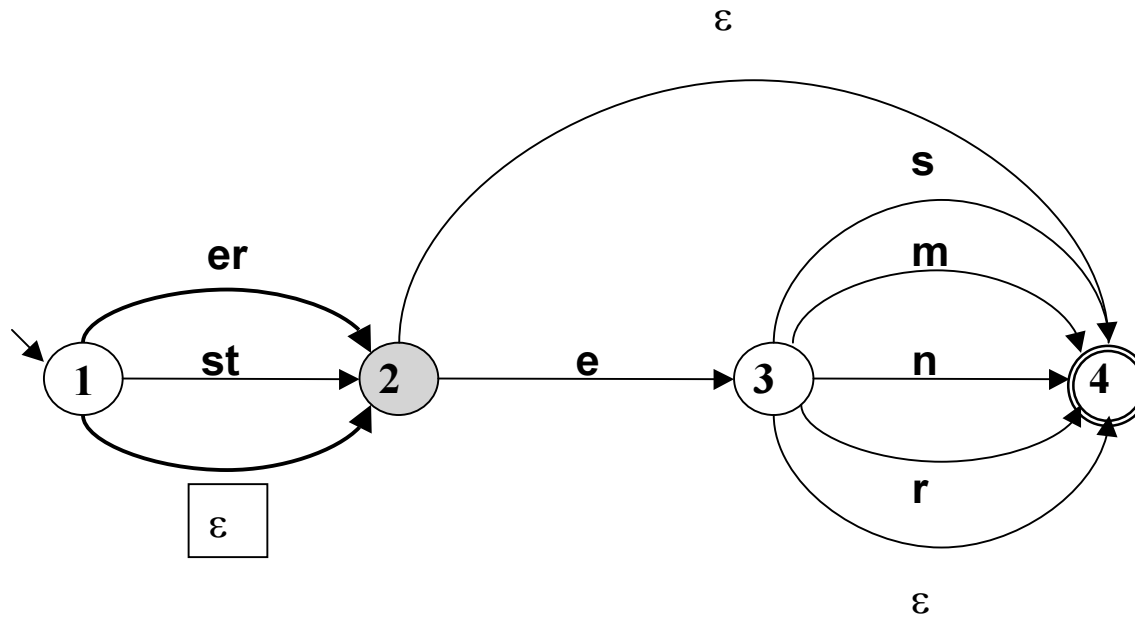
Pfadsuche: Nimm Aufgabe von der Agenda Backtracking!



Eingabewort: klein eres

Agenda: _____

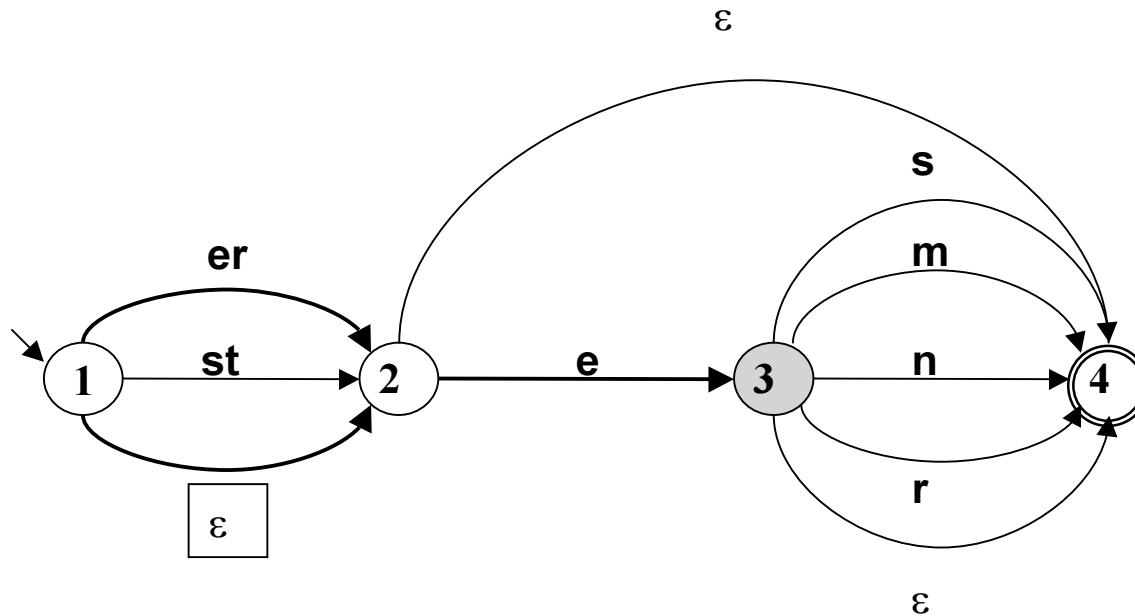
Pfadsuche: Generiere Aufgaben



Eingabewort: klein eres

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

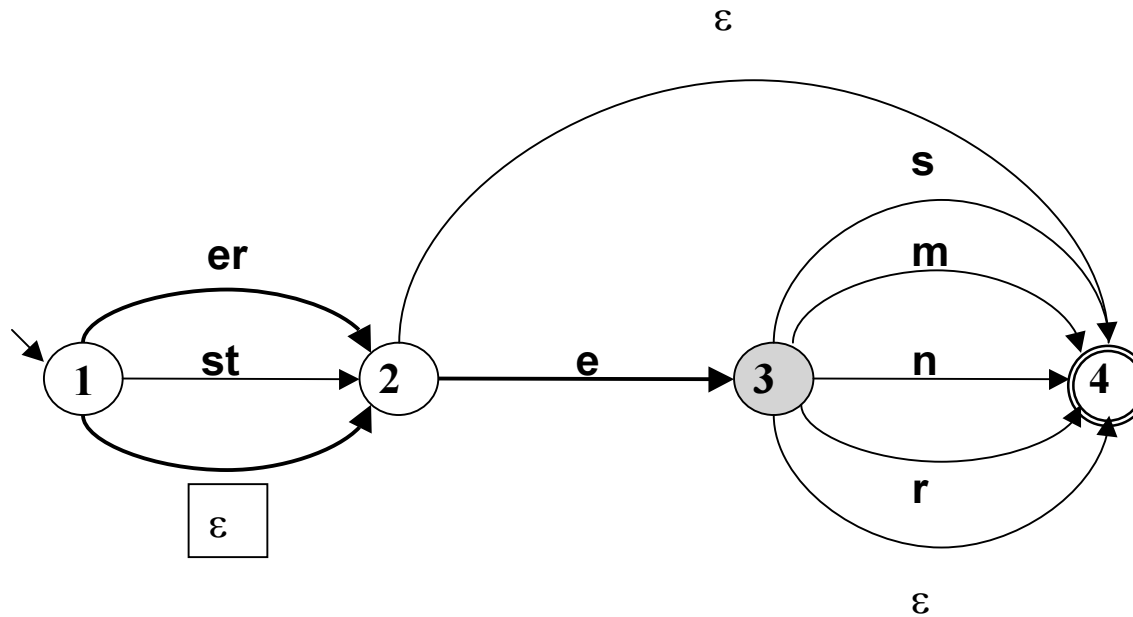
Pfadsuche: Nimm Aufgabe von der Agenda



Eingabewort: klein eresu

Agenda: 4 -- klein eresu

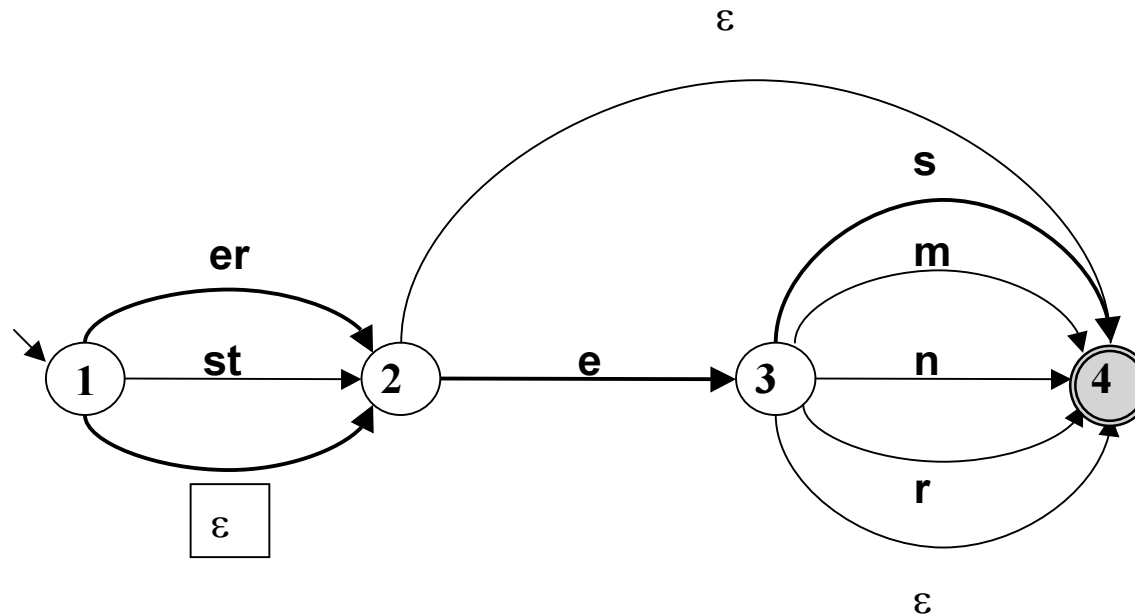
Pfadsuche: Generiere Aufgabe



Eingabewort: klein eresu

Agenda: 4 -- klein eres_
4 -- klein eresu

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



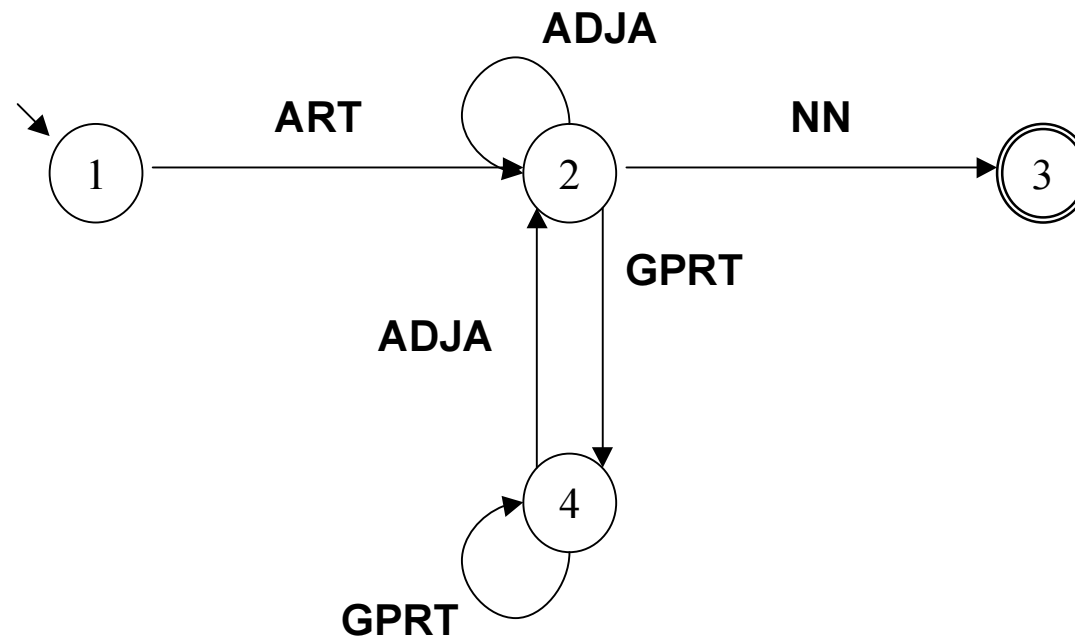
Eingabewort: klein eres_

Agenda: 4 -- klein eres

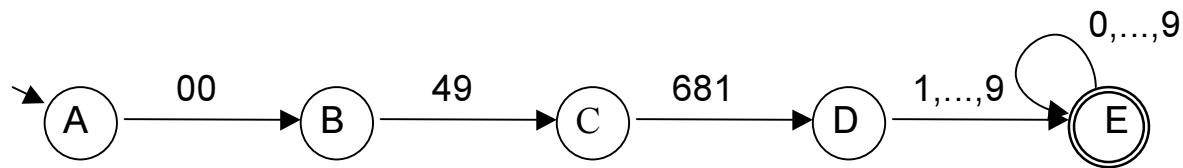
Anmerkung zum Pfadsuche-Algorithmus

- Der vorgestellte Pfadsuche-Algorithmus („Tiefensuche mit Backtracking“) ist vollständig – wenn das Diagramm/der Automat keine Leerwort-Zyklen enthält (Schleifen, bei deren Durchlaufen das leere Wort abgearbeitet wird).
- Der Pfadsuche-Algorithmus ist ineffizient: Wenn es für jede Konfiguration (Kombination aus Zustand und Eingabewort) durchschnittlich zwei erlaubte Kanten gibt, benötigt der Algorithmus zur vollständigen Abarbeitung eines Eingabewortes w im schlechtesten Fall etwa $2^{|w|}$ Schritte. Der Zeitbedarf wächst exponentiell mit der Wortlänge.
- Man kann die Situation verbessern, indem man andere Algorithmen verwendet, den Algorithmus optimiert, oder das Diagramm geschickt schreibt („teure“ Strukturen vermeiden). Wir werden eine grundsätzlich andere Möglichkeit betrachten.

Das Zustandsdiagramm für Wortartmuster: Ein deterministisches Diagramm



Internationale Vorwahl für Saarbrücken



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.

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: 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

Beispiel: Das Zustandsdiagramm 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

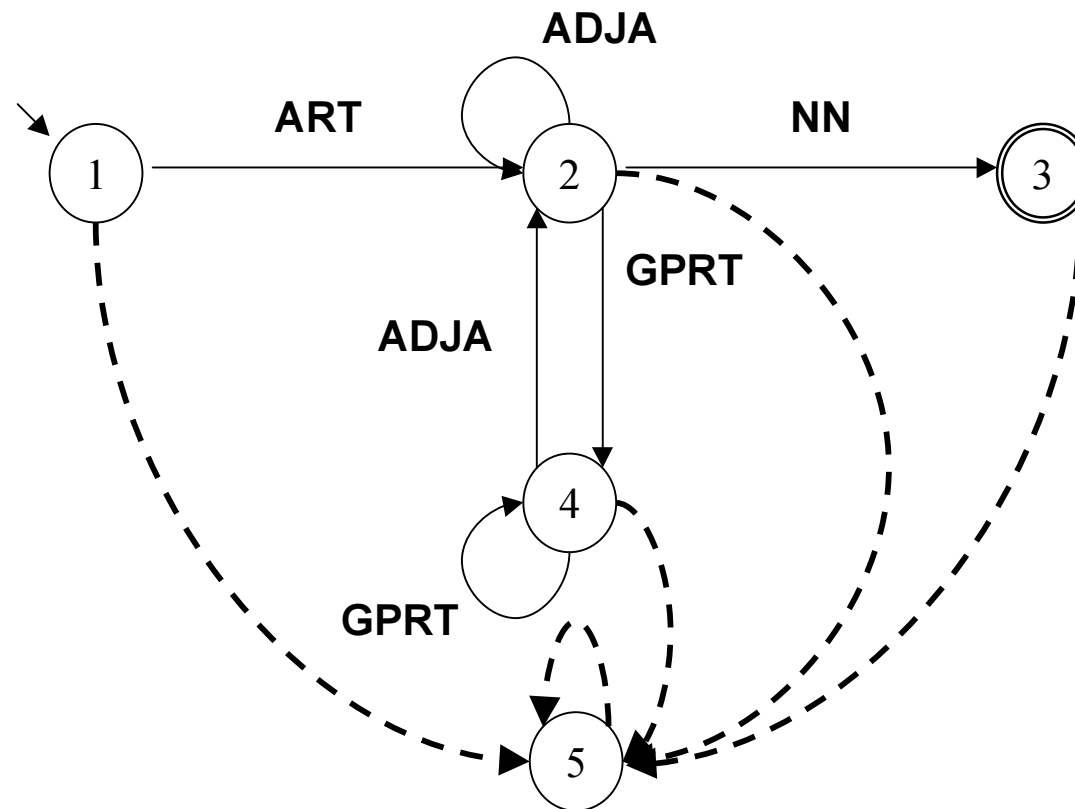
δ :	ART	ADJA	NN	GPRT
1	2			
2		2	3	4
3				
4		2		4

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: Nein.

Deterministische und nicht-deterministische Automaten [2]

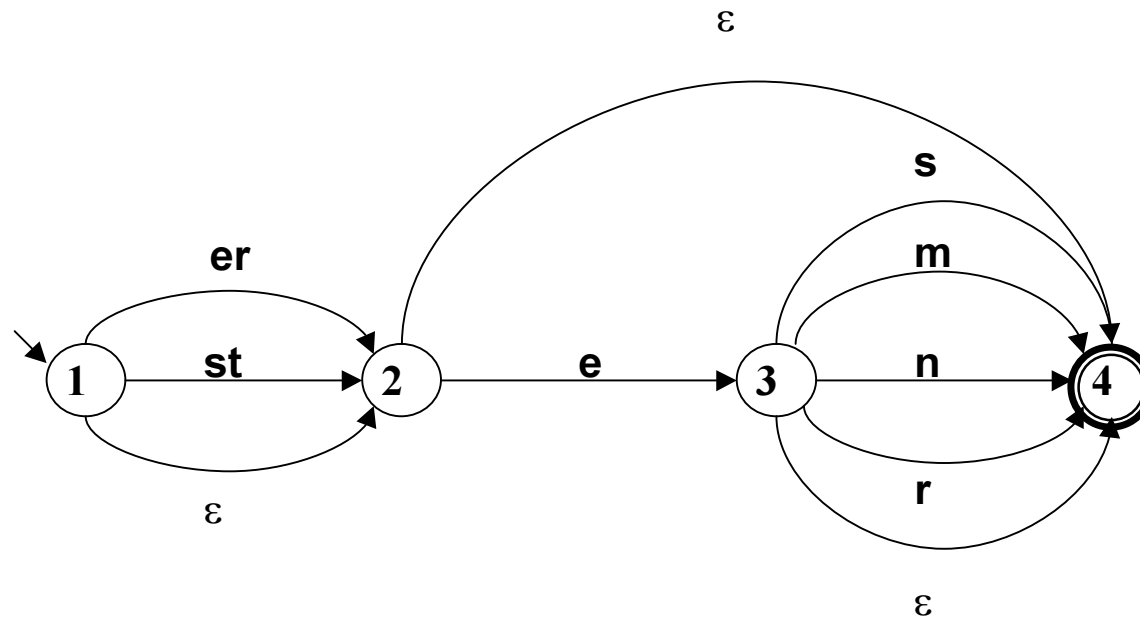
- Jede Sprache, die von einem NEA akzeptiert wird, kann auch durch einen DEA beschrieben werden (und, trivialerweise, auch umgekehrt: ein DEA ist ein spezieller NEA).
- Es gibt einen konstruktiven Beweis, d.h., ein Verfahren, das es erlaubt, zu jedem NEA A einen DEA A' zu konstruieren, so dass $L(A') = L(A)$.

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

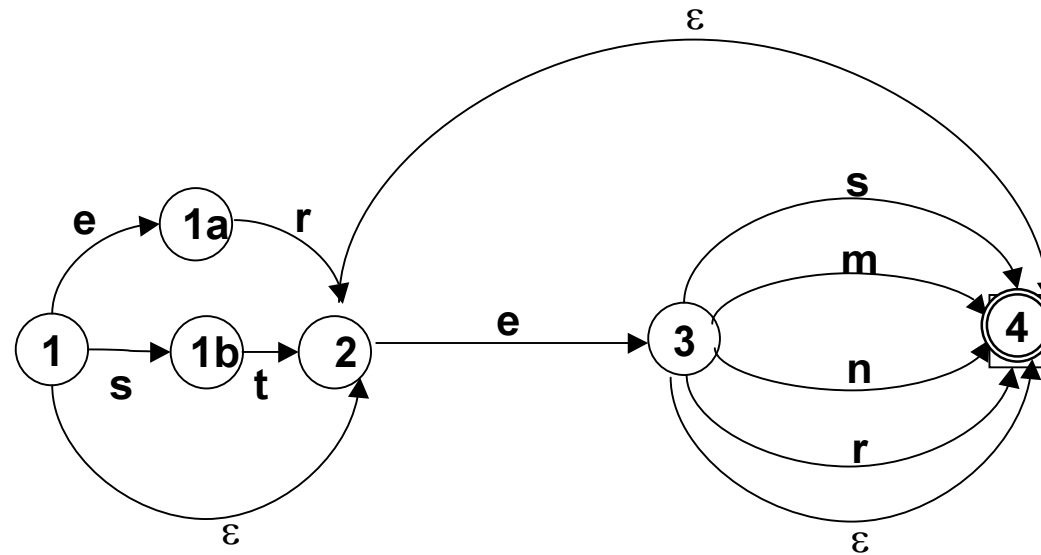
- Für alle Kanten $\langle q, w, q' \rangle$ mit $w = a_1 \dots a_n$, $n > 1$:

Entferne $\langle q, w, q' \rangle$ aus Δ .

- Erweitere K um neue Zustände q_1, \dots, q_{n-1} .
- Erweitere Δ um neue Kanten

$\langle q, a_1, q_1 \rangle, \langle q_1, a_2, q_2 \rangle, \dots, \langle q_{n-1}, a_n, q' \rangle$

Beispiel-Automat nach Schritt 1:



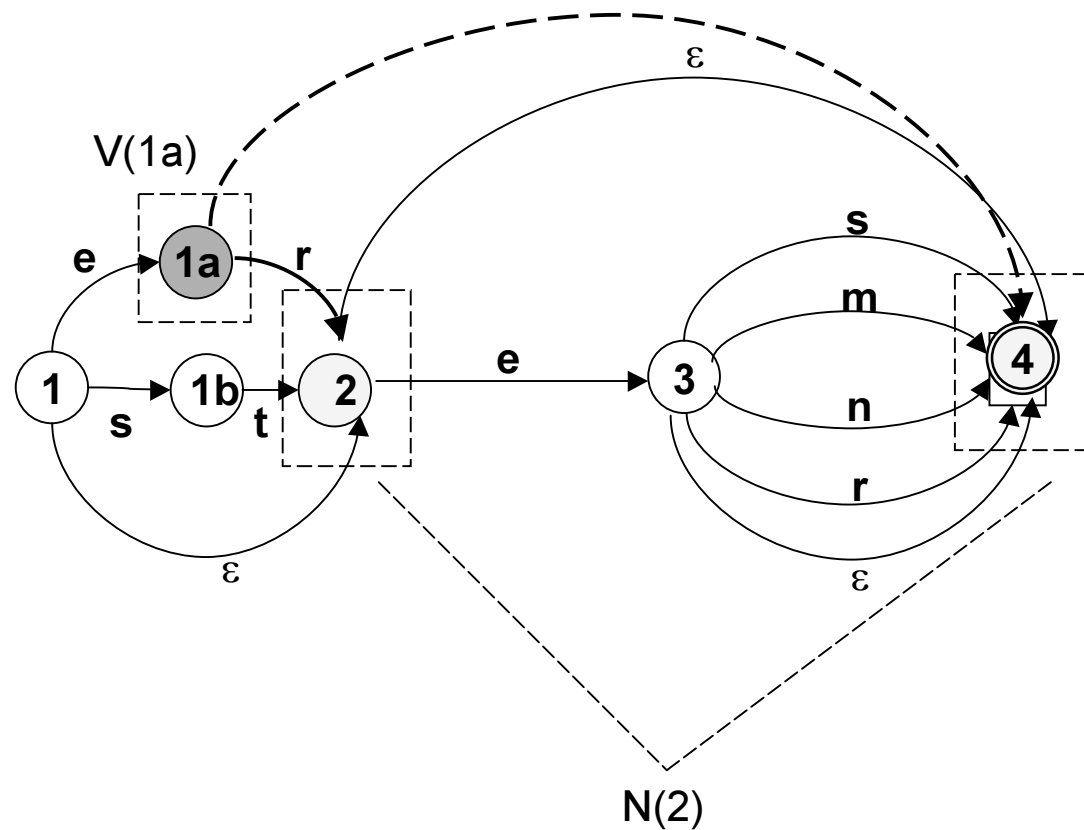
Schritt 2: Beseitigung von ε -kanten

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

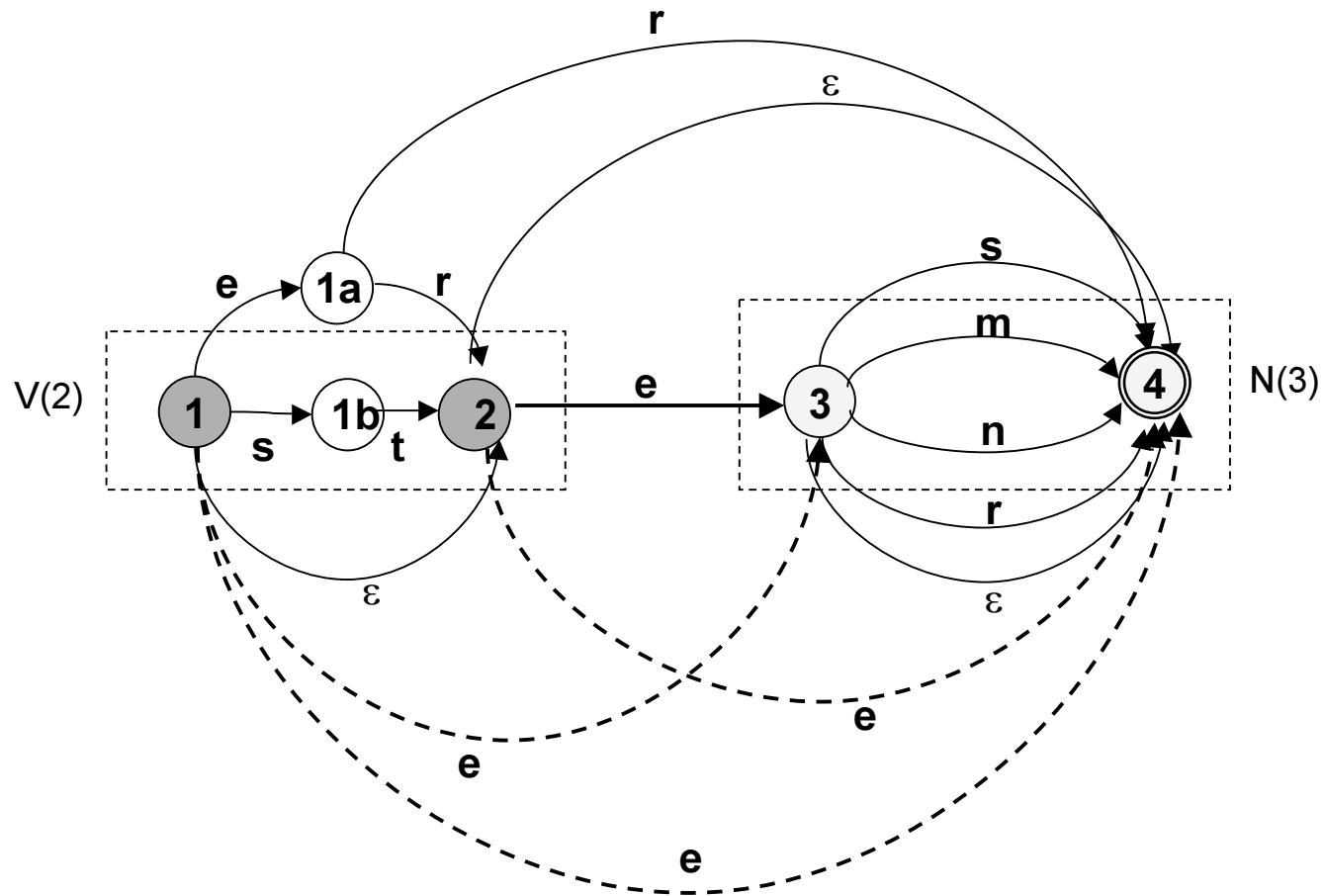
Anmerkung: $V_\varepsilon(p)$ und $N_\varepsilon(p)$ enthalten insbesondere p selbst.

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

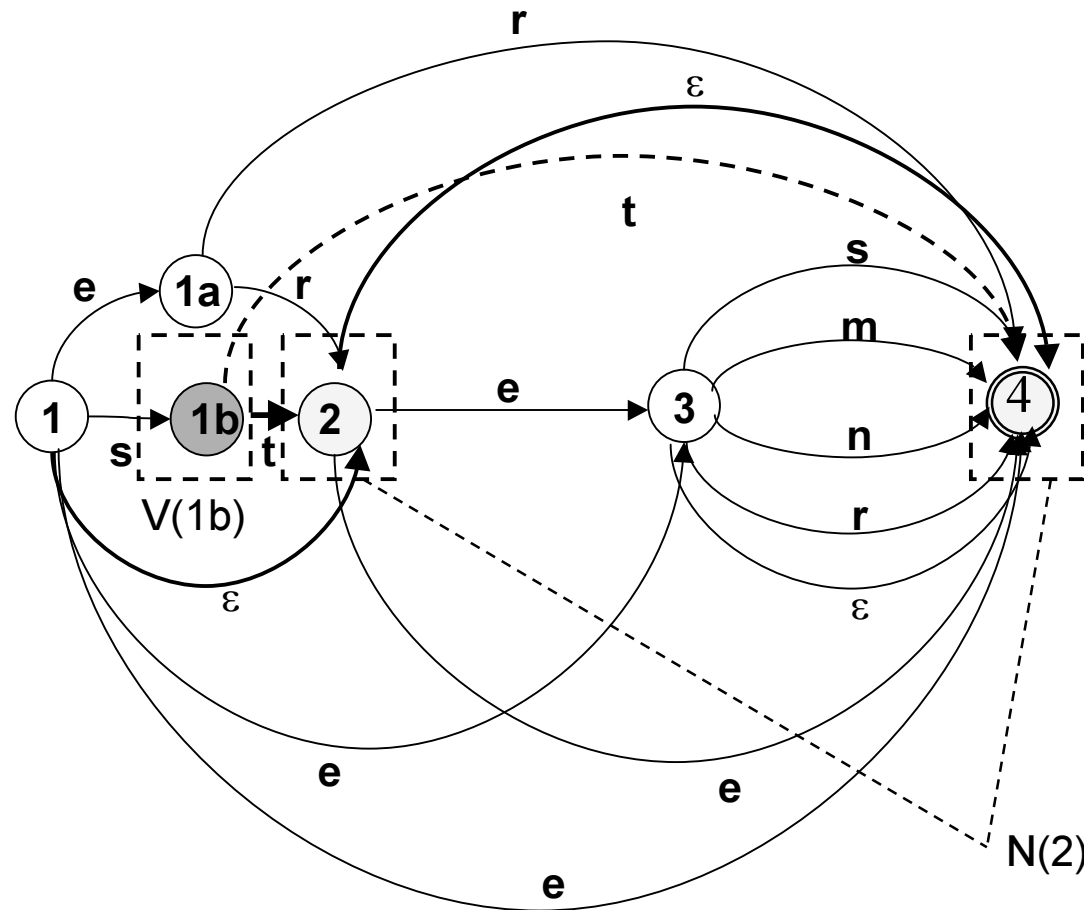
Schritt 2: Beseitigung von ε -Kanten



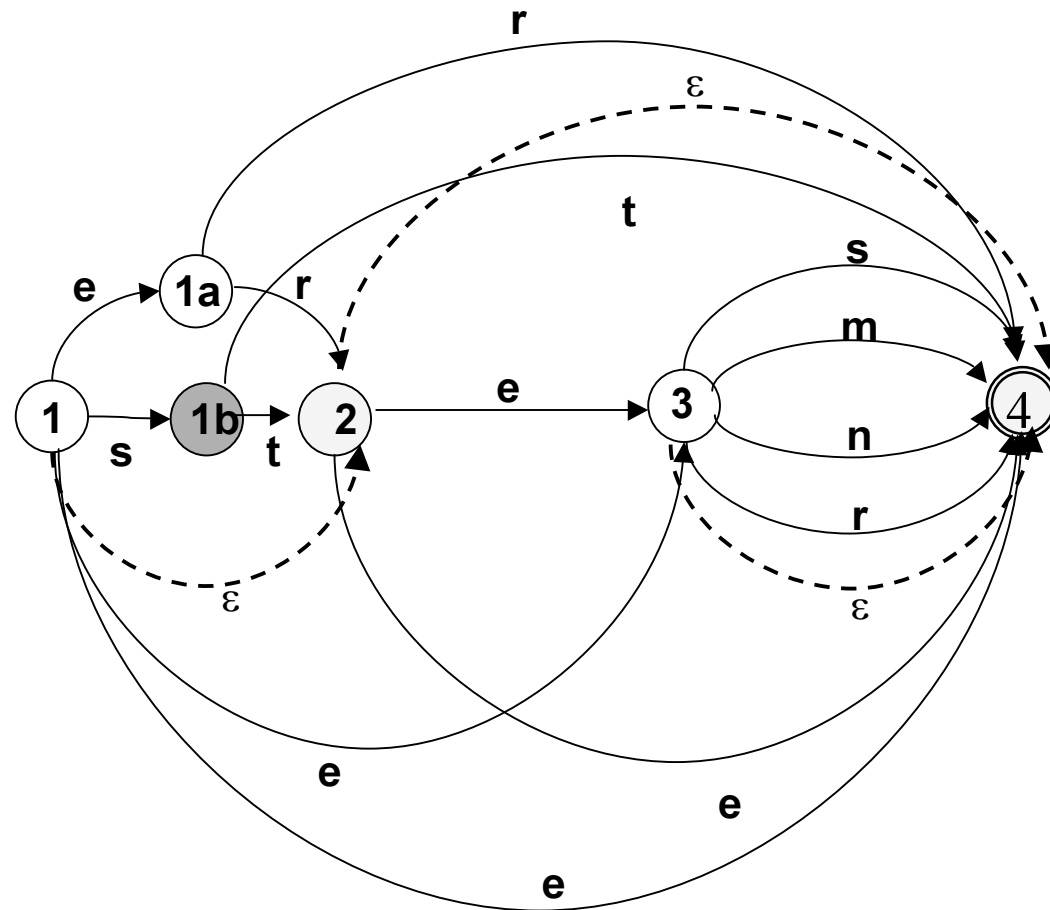
Schritt 2: Beseitigung von ε -kanten



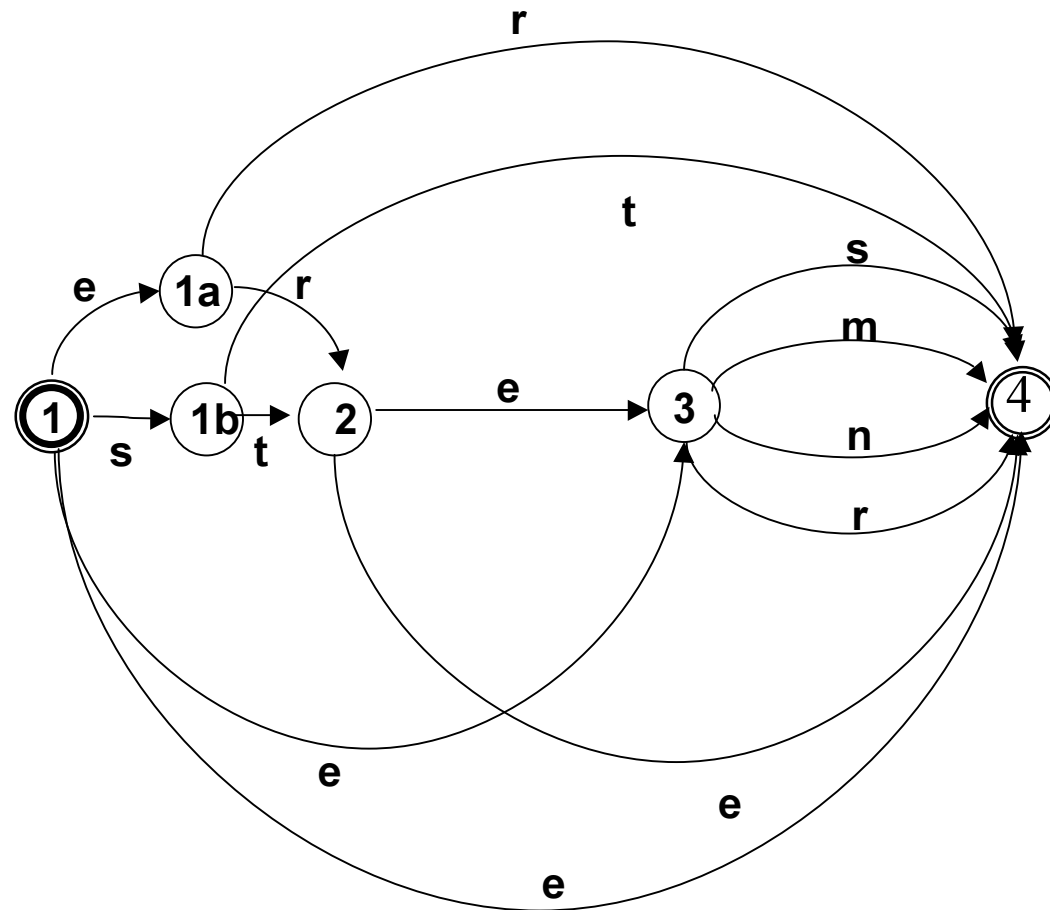
Schritt 2: Beseitigung von ε -kanten



Schritt 2: Beseitigung von ε -kanten



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

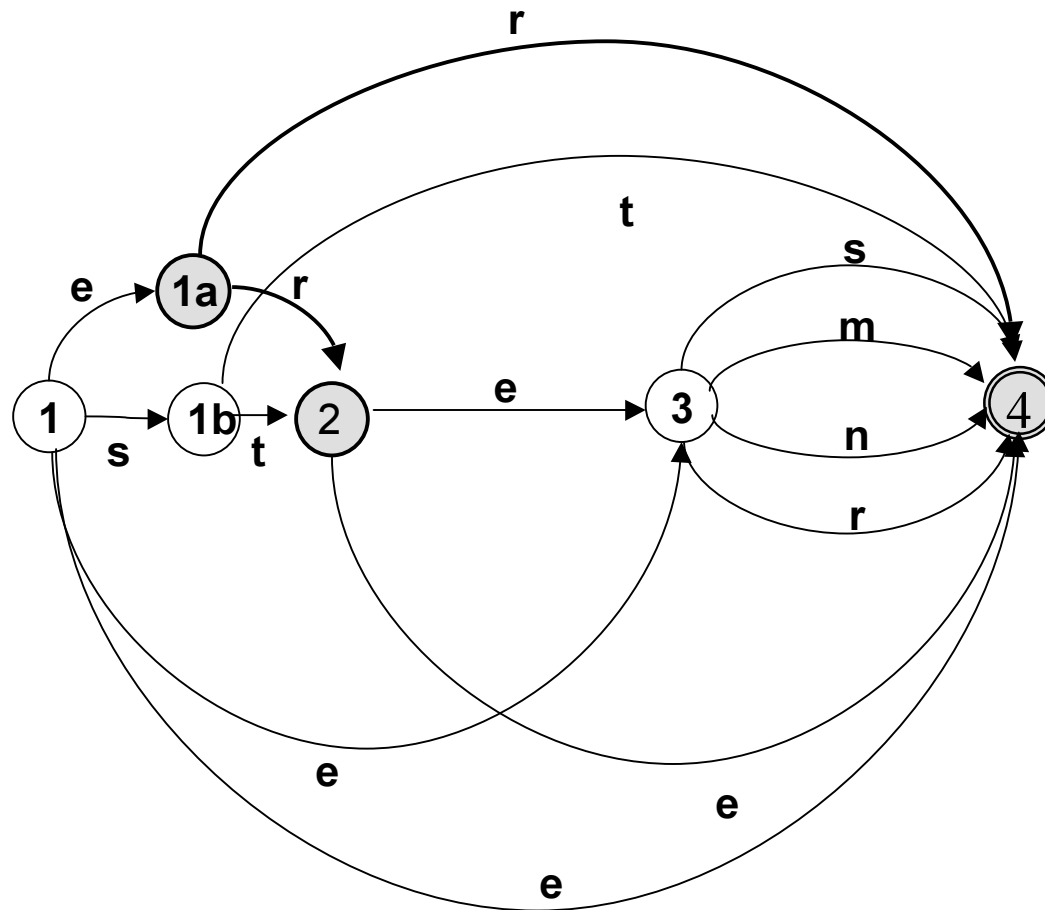


Schritt 3: Potenzautomaten-Konstruktion, Vorüberlegung

- 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.

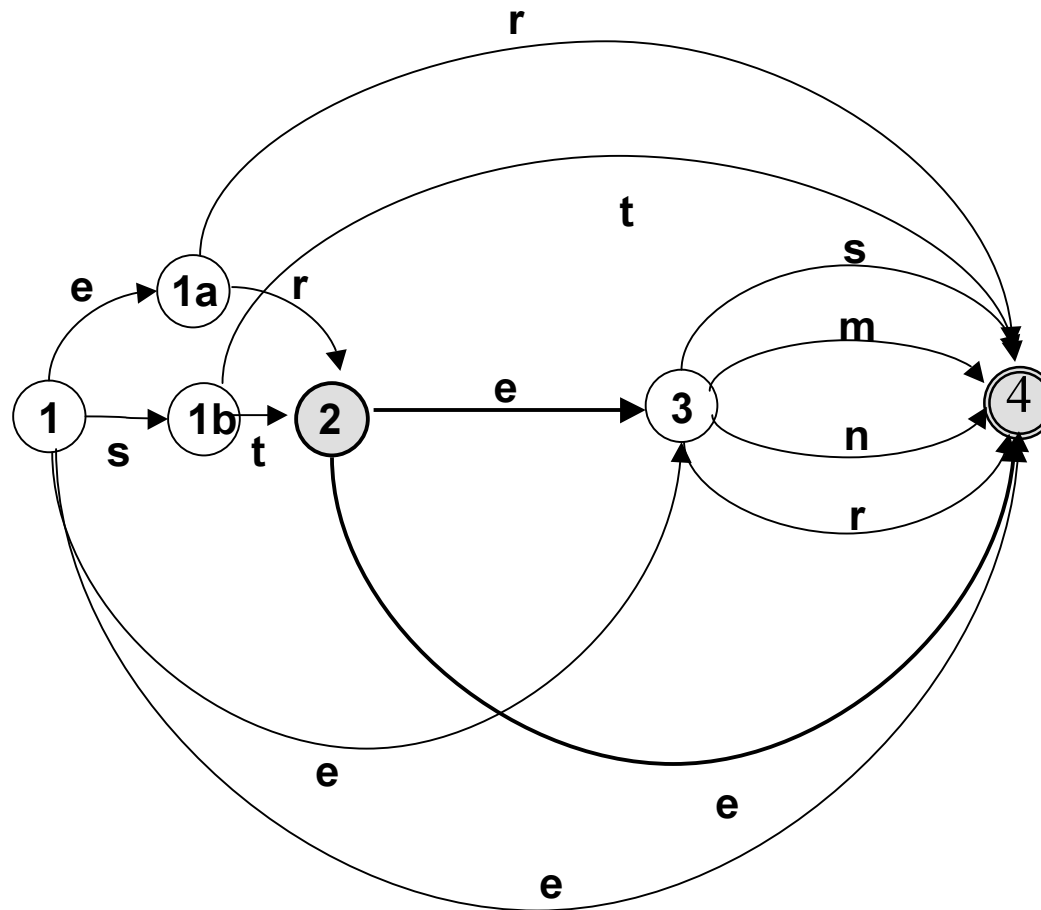
1a -- klein eres
3 -- klein eres
Agenda: 4 -- klein eres

Pfadsuche als Breitensuche



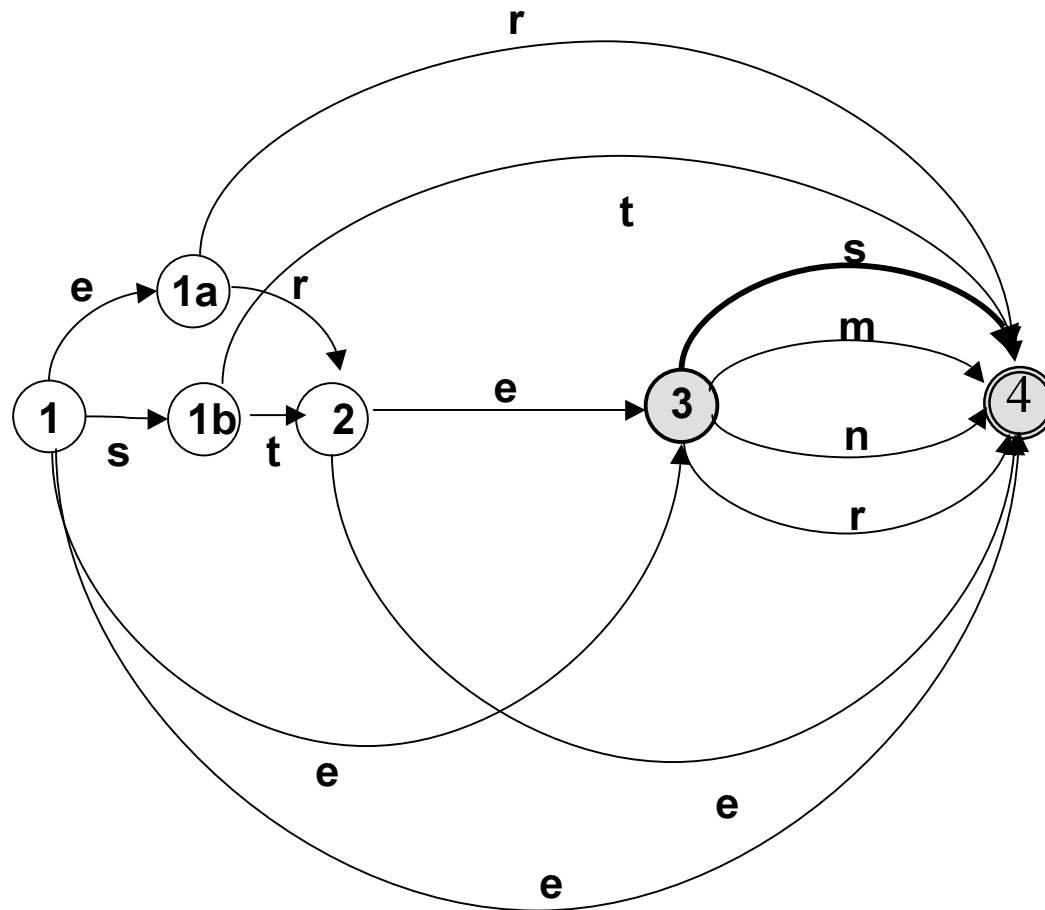
Eingabewort: klein eres

Pfadsuche als Breitensuche



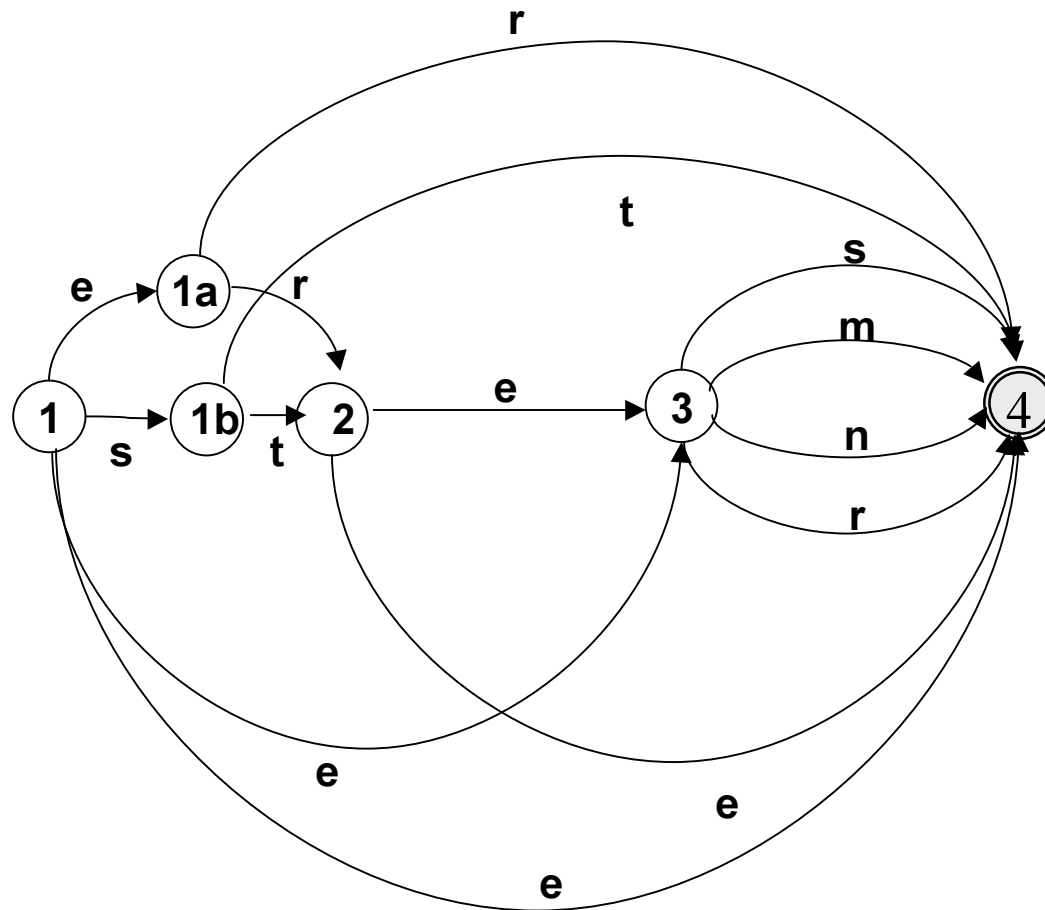
Eingabewort: klein eres

Pfadsuche als Breitensuche



Eingabewort: klein eresu

Pfadsuche als Breitensuche



Eingabewort: 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 der ersten 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 das Suchverfahren 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 $A' = \langle K', \Sigma, \delta, s', F' \rangle$
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. $q' \cap F \neq \emptyset$

Praktisches Vorgehen

Der Potenzautomat A' zu $A = \langle K, \Sigma, \Delta, s, F \rangle$ hat $2^{|\Delta|}$ 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.

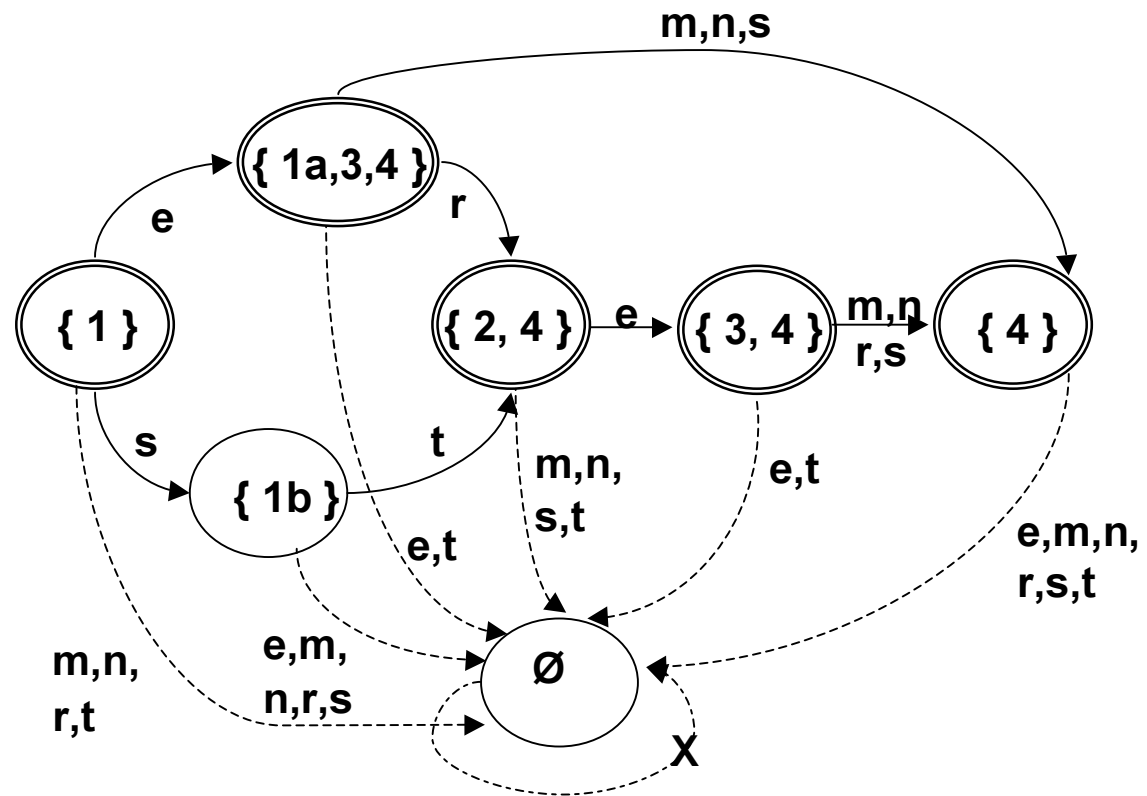
Beispiel: DEA für Adjektiv-Endungen

- 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 Folie 42
- 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

DEA für Adjektiv-Endungen, Übergangstabelle

$\delta:$	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

Das Diagramm



Noch einmal: Der DEA für Adjektiv-Endungen

- Die Zustandsmenge des Potenzautomaten A' ist eigentlich $K' = \wp(K)$, er hat in unserem Beispiel also $2^6 = 64$ Zustände. Wie die Übergangstabelle zeigt, sind vom Startzustand $\{1\}$ aus aber nur 7, ohne den „trap state“ \emptyset 6 echte Zustände erreichbar. Die übrigen Zustände können ignoriert werden.

Wir können also, wie im Diagramm, ausgehen von:

$$K' = \{\{1\}, \{1a,3,4\}, \{1b\}, \{2,4\}, \{3,4\}, \{4\}, \emptyset\} \text{ und}$$

$$F' = \{\{1\}, \{1a,3,4\}, \{2,4\}, \{3,4\}, \{4\}\}$$

- Im Diagramm können wir außerdem noch, per Konvention, den Zustand \emptyset und alle hinführenden Kanten unterschlagen, und erhalten dann das vereinfachte Diagramm auf der folgenden Folie.

Das Diagramm, vereinfacht

