

Einführung in die Computerlinguistik

Finite State Transducers und Morphologie

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Summer 2018



Morphologische Grundbegriffe (1)

- Wort / Lexem: abstrakte Einheit, die verschiedenen Formen zugrunde liegt.
Bsp.: *komm-*
- Wortform: verschiedene Formen eines Lexems
Bsp.: *kommen, komme, kommst, kommt, kommend, kommenden, ...*
- Paradigma: Menge von Wortformen eines Lexems.
Bsp.: *geh-e*
geh-st
geh-t
geh-en
- Synkretismus: Zusammenfallen verschiedener Wortformen
Bsp.: 1. und 3. Pers. Plural des Lexems *geh*: *gehen*

Morphologische Grundbegriffe (2)

Wortbildungsprozesse:

- **Komposition:** Wort + Wort \mapsto Wort
Kinder + Garten = Kindergarten
- **Derivation:** Wort + (gebundenes Morphem) \mapsto Wort
Zwerg + -lein = Zwerglein
- **Flexion (Beugung):** Wort + Flexionsmorphem \mapsto Wortform
Zwerg + -e = Zwerge

Morphologische Grundbegriffe (3)

Welche Wortbildungsprozesse liegen hier vor?

Kindchen	Haustür
gibst	Balls
Dummheit	Vereinsamung
Lauf	Jagdhund
Bälle	verkaufte
Diebe	Hausboot
Verlierer	Korkenzieher

Morphologische Grundbegriffe (4)

- Morphem: Kleinste bedeutungstragende Einheit
Bsp.: Zwerg, garten in *Gartenzwerg*, *geh* und *-st* in *gehst*
- freies Morphem: ein Morphem, das auch isoliert als Wortform auftreten kann
Bsp.: Hund, Kind, auf, und, ...
- gebundenes Morphem: Morphem, das nicht frei ist.
Bsp.: -s, ver-, -lein, ...
- Wurzel: Morphem, das Ausgangspunkt für Flexion und Derivation ist.
Bsp.: lauf, Kind, Hund, schwarz, ...

Morphologische Grundbegriffe (5)

- **Affixe:** periphere gebundene Morpheme. Präfixe/Suffixe, je nachdem, ob sie vor oder hinter der Wurzel stehen.
Bsp.: Suffix *-st*, Präfix *ent-*
- **Stamm:** Morphemcluster ohne Flexionsaffixe
Bsp.: *vergleich*, *verkauf*, *enthalt*, ...
- **Allomorphe:** bedeutungsgleiche Morpheme.
Bsp. Pluralmorphem: $\{-e, -er, -s, \dots\}$ (*-e*, *-er* und *-s* sind allomorph)

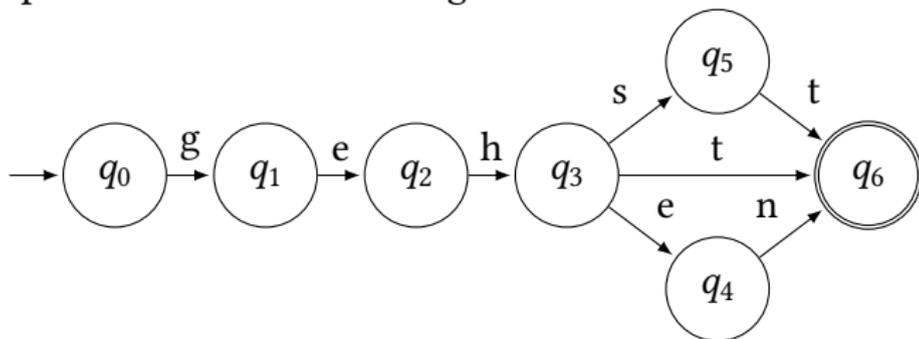
Morphologische Grundbegriffe (6)

Aus welchen Morphemen sind die folgenden Wortformen entstanden?

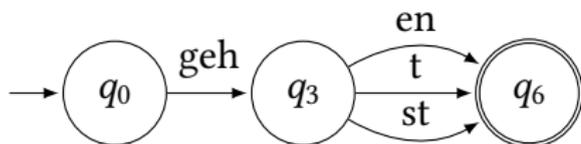
Kindchen	Haustür
gibst	Balls
Dummheit	Vereinsamung
Lauf	Jagdhund
Bälle	verkaufte
Diebe	Hausboot
Verlierer	Korkenzieher

Morphologie mit endlichen Automaten (1)

Bsp.: flektierte Formen von *geh-*



Vereinfachung: Kanten können Konkatenationen von Eingabesymbolen als Labels haben.

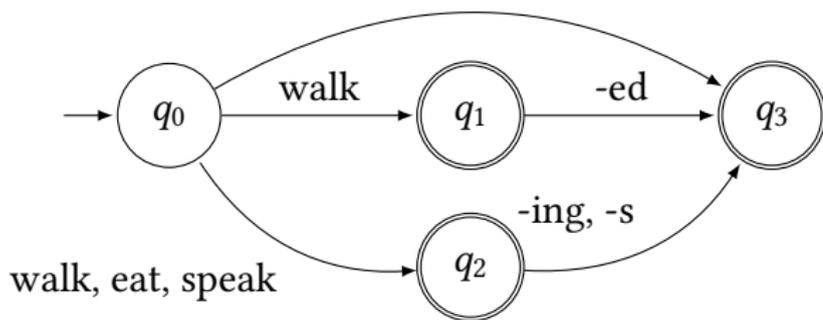


Morphologie mit endlichen Automaten (2)

Englische Verben:

walk speak, eat

eaten, ate, spoke, spoken



Finite State Transducer (1)

Ziel: morphologisches Parsing.

Eingabe	Ausgabe
---------	---------

cats	cat N Pl
------	----------

cat	cat N Sg
-----	----------

Eingabe	Ausgabe
---------	---------

gehst	geh V 2.Sg
-------	------------

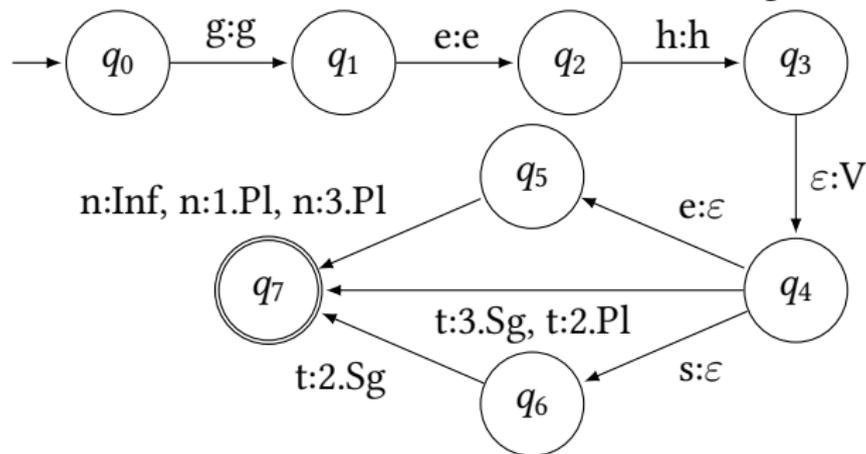
gehen	geh V Inf
-------	-----------

gehen	geh V 3.Pl
-------	------------

gehen	geh V 1.Pl
-------	------------

Finite State Transducer (2)

Idee: Der FSA wird mit einer zusätzlichen Ausgabe versehen.



String “gehst” wird akzeptiert und erzeugt “gehV2.Sg”

Finite State Transducer (3)

Ein **finite state transducer** (FST) M ist ein Tupel $\langle Q, \Sigma, \Delta, \delta, q_0, F \rangle$ so dass

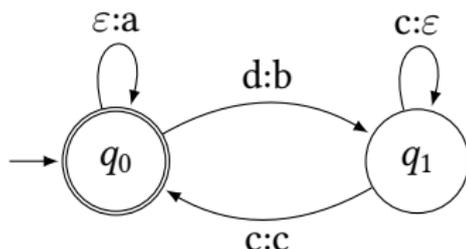
- $\langle Q, \Sigma, q_0, F \rangle$ sind wie bei einem NFA,
- Δ ist das Ausgabealphabet, und
- $\delta : Q \times (\Sigma \cup \{\varepsilon\} \times \Delta \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q)$ ist die Übergangsfunktion.

D.h., δ gibt für einen Zustand und ein Paar aus Eingabe- und Ausgabesymbol eine Menge von möglichen neuen Zuständen an. Dabei ist ε bei Ein- und Ausgabesymbol mit eingeschlossen.

Bsp.: $\delta(q_4, \langle t, 2.Pl \rangle) = \{q_7\}$

Finite State Transducer (4)

Beispiel:



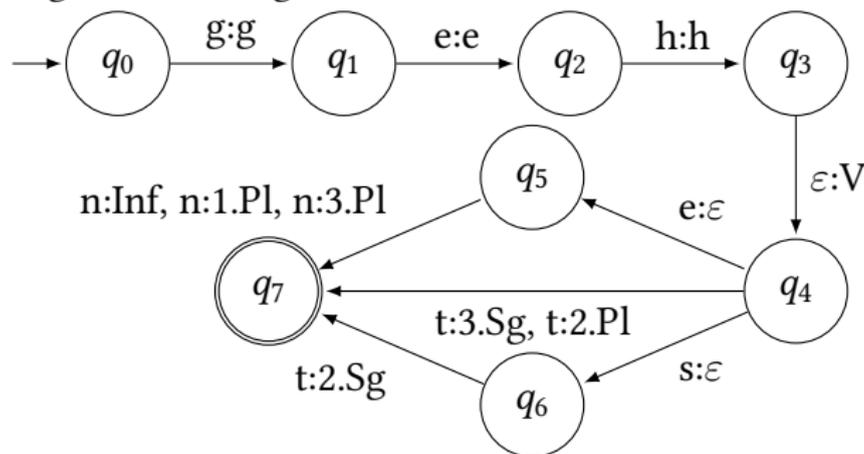
Eingabe	Ausgabe
ε	a^*
dc	$a^* bca^*$
$dccc$	$a^* bca^*$
$dcdcc$	$a^* bca^* bca^*$

Dieser FST macht folgendes:

- Akzeptiert wird die von $(dc^+)^*$ denotierte reguläre Sprache.
- Ausgabe ist jeweils ein Wort, in dem
 - jedes d in der Eingabe durch ein b ersetzt wird,
 - von jeder c -Folge wird nur das letzte c in die Ausgabe übernommen, die anderen werden gelöscht,
 - und es werden am Wortanfang und Ende und vor jedem b im Ausgabewort beliebig viele (eventuell auch 0) as eingefügt.

Finite State Transducer (5)

Ein FSA kann auch als Generierer verwendet werden. D.h., Ein- und Ausgabe werden gewissermaßen vertauscht.



String “gehV2.Sg” wird akzeptiert und erzeugt “gehst”

Finite State Transducer (6)

- Ein FST kann als ein Automat mit zwei Bändern betrachtet werden.

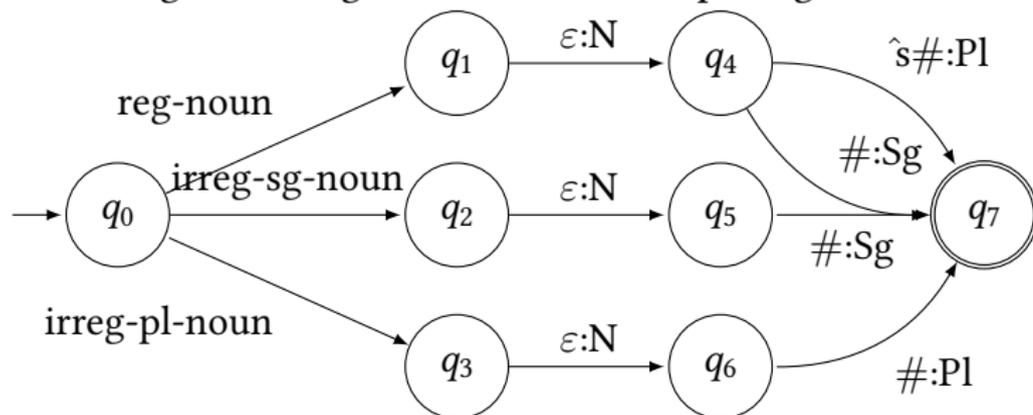
Lexikalische Ebene:	...	c	a	t	N	Pl	...
Oberfläche:	...	c	a	t	s		...

⇒ Two-level morphology (Koskenniemi)

- Zusätzlich führen wir im Eingabealphabet Symbole $\hat{}$ und $\#$ ein für Morphemgrenze und Wortgrenze.

Finite State Transducer (7)

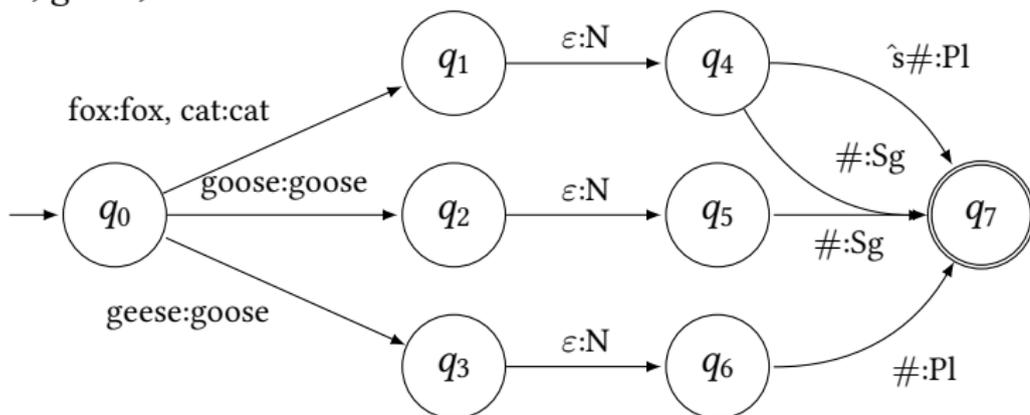
FST für englische Singular- und Pluralmorphologie:



Für die Platzhalter reg-noun, irreg-sg-noun und irreg-pl-noun müssen noch entsprechende Nomen eingesetzt werden. Außerdem müssen wir dafür sorgen, dass bei einem unregelmäßigen Plural der richtige Stamm ausgegeben wird.

Finite State Transducer (8)

fox, goose, cat:



Außerdem benötigen wir orthographische Regeln der folgenden Art:
e-insertion: “e” hinzufügen nach einem -s, -z, -x, -ch, -sh und vor -s

Finite State Transducer (9)

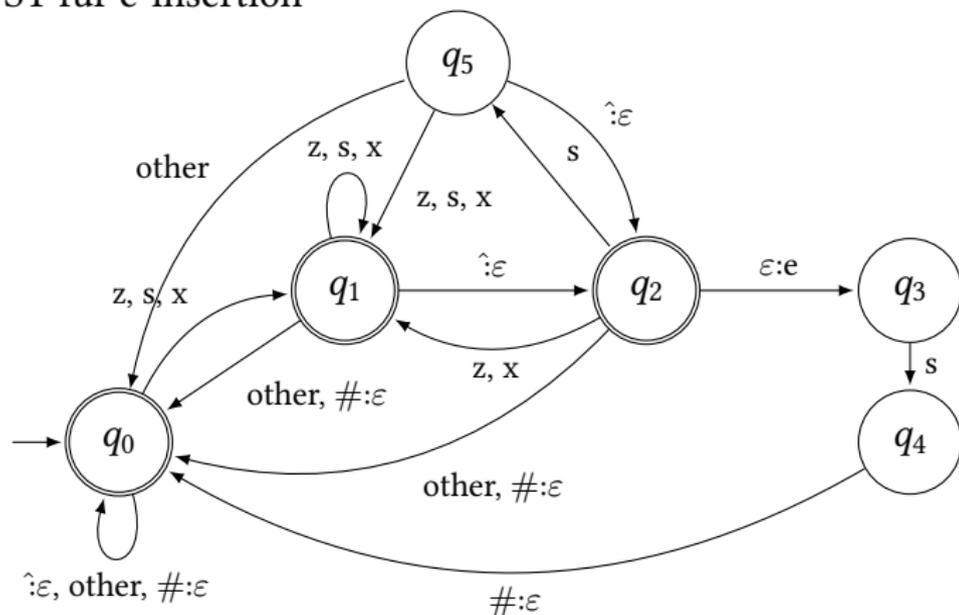
Wir haben also drei Ebenen:

Lex. Ebene:	...	f	o	x	N	Pl		...
Zwischenebene:	...	f	o	x	^	s	#	...
Oberfläche:	...	f	o	x	e	s		

- Orthographische Regeln können ebenfalls als FSTs realisiert werden.
- Der Lexikalische FST und die FST für orthographische Regeln werden hintereinandergeschaltet (*cascaded transducers*)

Finite State Transducer (10)

FST für e-insertion



(ein einzelnes a steht hier für das Paar $a : a$, z.B. "z" für "z:z". "other" bezeichnet alles außer $z, s, x, \hat{\epsilon}, \#$.)