

Morphologische Analyse und Generierung

FINITE STATE MORPHOLOGY

C.D. Johnson: *Formal Aspects of Phonological Description* (1972):

Phonologische Regeln werden üblicherweise als kontext-sensitive Regeln notiert

$$\alpha \rightarrow \beta/\gamma\text{-}\delta$$

Ihre Anwendung unterliegt jedoch einer starken *Einschränkung*:

Der Anwendungsbereich muss sich nach links oder rechts in der Zeichenkette bewegen, d.h. Regeln dürfen nicht im selben Zyklus auf ihr Ergebnis angewandt werden.

⇒ Ein-/Ausgabepaare bilden aus algebraischer Sicht **reguläre Relationen**
(Kay und Kaplan (1981, 1994))

Beispiel: Generierung mit der Regel $\epsilon \rightarrow ab/_b$

1. unzulässig:

$$\rightarrow ab \rightarrow aabb \rightarrow aaabbb \rightarrow \dots \rightarrow aa^n b^n$$

2. zulässig:

$$\rightarrow ab \rightarrow aabb \rightarrow aababb \rightarrow \dots \rightarrow a[ab]^*b$$

Finite-State Transducer (FST)

Eine reguläre Relation kann als endlicher überführender Automat (FST) dargestellt werden.

Kantenbeschriftungen $x:y$ sind E/A-Paare ($x:x = x$)

Im *Beispiel*:

- 0 ist Anfangszustand und einziger Endzustand
- ϵ ist Nullsymbol

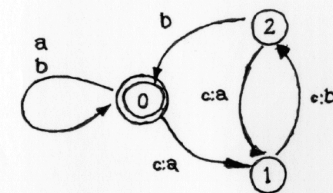


Figure 2. Transducer implementing $\epsilon \rightarrow ab / _b$

	a	ϵ	ϵ	b
0	0	1	2	0
a	a	b	b	

Figure 3: Transducer mapping ab to $aabb$. (or vice versa)

Serielle Komposition

FSTs sind abgeschlossen unter serieller *Komposition* (Schützenberger 1961)

Gegeben:

Zwei nacheinander anzuwendende Regeln R1 und R2, so dass die Ausgabe von R1 zur Eingabe von R2 wird.

Korrespondierende Transducer können zu einem neuen Transducer komponiert werden (ohne Zwischenrepräsentation)

Beispiel: Seien zwei Regeln gegeben:

(1) $N \rightarrow m / _ p; \text{ elsewhere, } n$

besagt: N auf der lexikalischen Ebene (Input) soll, wenn dort ein p folgt, auf der Oberfläche als m realisiert werden, sonst als n (Output)

(2) $p \rightarrow m / m _$

besagt: p auf der lexikalischen Ebene soll auf der Oberflächenebene als m realisiert werden, wenn ihm auf dieser ein m vorangeht.

(a) $N \rightarrow m / _ p; \text{ elsewhere, } n.$
 (b) $p \rightarrow m / m _$

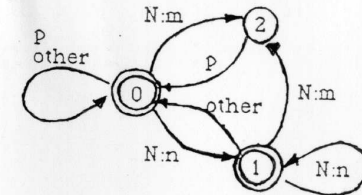
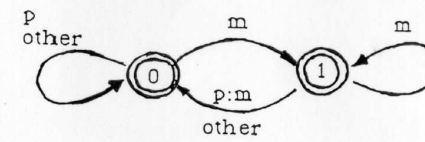


Figure 4: Transducer for $N \rightarrow m / _ p$



Komposition: Grundidee

Jeder Zustand im Kompositionsautomaten T3 entspricht einem Paar von Zuständen in den ursprünglichen Automaten T1 und T2, beginnend mit den beiden Anfangszuständen.
 $x:y$ in T1 und $y:z$ in T2 $\Rightarrow x:z$ in T3

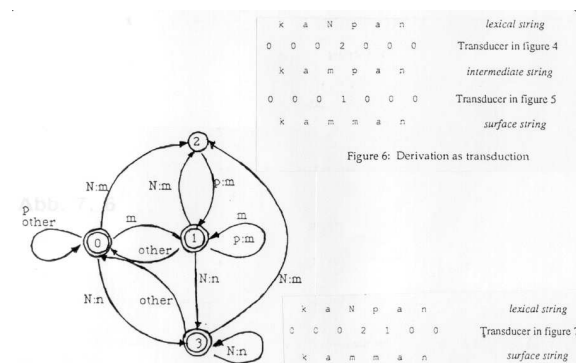


Figure 7: Composition of the transducers in Figures 4 and 5

Figure 8: Transduction with a single transducer

Kaskadenkomposition: allgemein

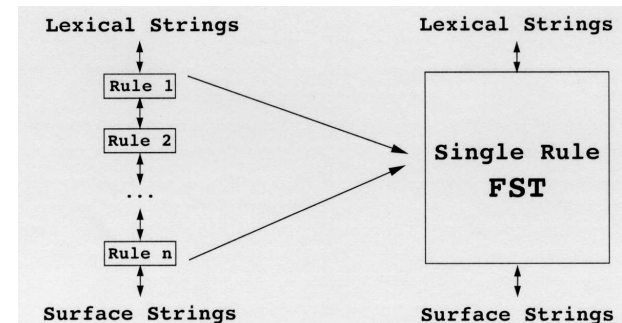


Figure 1: A Cascade of Rewrite Rules Composed into a Single FST

Effizienz!

Bemerkung: Es gibt keine korrespondierende Ersetzungsregel

Anmerkungen zur Komposition

Die Konstruktion eines Compilers für Ersetzungsregeln setzt eine vollständige Implementation der FS-Grundoperationen wie Vereinigung, Durchschnitt, Komplementierung und Komposition voraus.

Allerdings besteht ein Problem mit der Praktikabilität des Ansatzes für die morphologische *Analyse*.

Traditionelle phonologische Ersetzungsregeln beschreiben die Korrespondenz zwischen lexikalischen und Oberflächenformen als eine gerichtete sequentielle Abbildung von lexikalischen in Oberflächenformen, also die *Generierung*.

Ob ein effizientes Generierungsverfahren auch zu einer effizienten *Analyseprozedur* — also in umgekehrter Richtung — führen würde, war unklar.

Beispiel (Karttunen):

Seien die beiden Ersetzungsregeln (1) $N \rightarrow m / _ p$ und (2) $p \rightarrow m / m _$ sequentiell angewandt. Die korrespondierenden FSTs bilden die lexikalische Form 'kaNpat' eindeutig in 'kammat' ab mit der Zwischenrepräsentation 'kampat'.

Werden die beiden FSTs in der umgekehrten Richtung auf die Eingabe 'kammat' angewandt, erhält man die drei in der Abbildung dargestellten Ergebnisse:

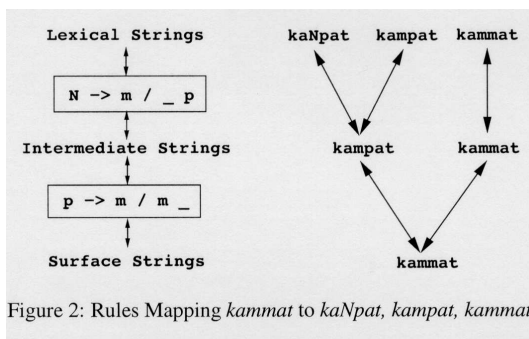


Figure 2: Rules Mapping kammat to kaNpat, kampat, kammat

Grund: Die Oberflächenform 'kammat' hat auf der Zwischenebene zwei mögliche Quellen. Die Anwendung der Regel (2) bildet 'kampat' und 'kammat' in dieselbe Oberflächenform ab. Die Zwischenform 'kampat' kann wiederum aus 'kampat' oder 'kaNpat' durch Anwendung der Regel (1) entstanden sein. D.h.:

Beide FSTs sind eindeutig in der Generierungs-, aber ambig in der Analyserichtung.

Wegen der Äquivalenz des Kompositionsautomaten zur ursprünglichen Kaskade bleibt das Ambiguitätsproblem bestehen.

Lösungsansatz:

Repräsentation des Lexikons selbst durch einen FST und Komposition mittels Regeln. Einbezug des Lexikons in die Komposition eliminiert die durch die Regeln erzeugten Ambiguitäten zur Compilerzeit — die Verzahnung vermeidet kombinatorische Explosion (Übergenerierung).

Koskenniemi's Zwei-Ebenen-Morphologie

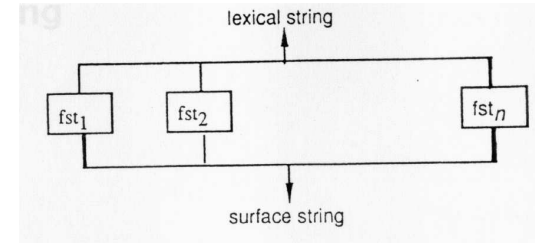
Grundidee: Parallelschaltung

- Regeln sind Constraints zwischen Symbolen, die parallel angewendet werden
— nicht sequentiell wie Ersetzungsregeln
- Die Constraints können sich auf den lexikalischen oder den Oberflächen-Kontext oder auf beide zugleich beziehen
- Lexikonzugriff und morphologische Analyse werden im Tandem durchgeführt

Die Korrespondenz zwischen lexikalischen und Oberflächen-Zeichenketten ist in jedem Fall eine reguläre Relation

⇒ Zerlegung in parallel komponierte Regeln (parallele Constraints) :

Lösung des Nichtdeterminismus für Analyse im Kay-Kaplan-Modell



Der Zwei-Ebenen-Ansatz ist

- einzelsprachunabhängig: rein deklarative Darstellung des sprachabhängigen Wissens;
- behandelt (Morpho)phonologie und Morphologie (Morphotaktik)
- nichtdirektional: ein parametrisierbarer Prozess für Generierung und Analyse.
- Effiziente Verarbeitung durch Compilation in endliche Automaten.

Zwei-Ebenen-Morphologie: Beispiel

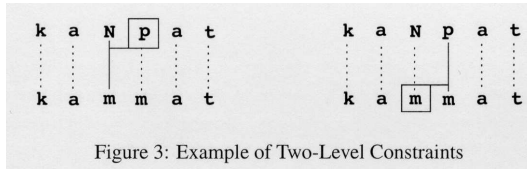


Figure 3: Example of Two-Level Constraints

Durch die Linien wird angezeigt, dass den Symbolen in der lexikalischen Zeichenkette 'kaNpat' paarweise ihre Realisierung in der Oberflächenkette 'kammāt' zugeordnet ist. Zwei der Symbolpaare werden durch den mit dem assoziierten Rahmen markierten Kontext beschränkt. Das Paar N : m wird auf die Umgebung *ingeschränkt*, in der auf der lexikalischen Seite unmittelbar p folgt; genauer: In diesem Kontext, werden alle anderen möglichen Realisierungen eines lexikalischen N *verhindert*. Ebenso erfordert das Paar p : m ein vorangehendes Oberflächen-m, und keine andere Realisierung von p ist hier erlaubt.

Regeln in Koskenniemi Notation:

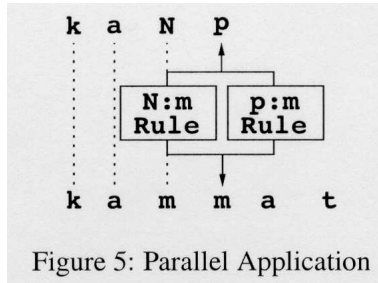
(1') N : m <=> _ p : und (2') p : m <=> : m _

Der Operator <=> kombiniert eine Kontextbeschränkung mit der Verhinderung jeder anderen Realisierung für das lexikalische Symbol des Paares.

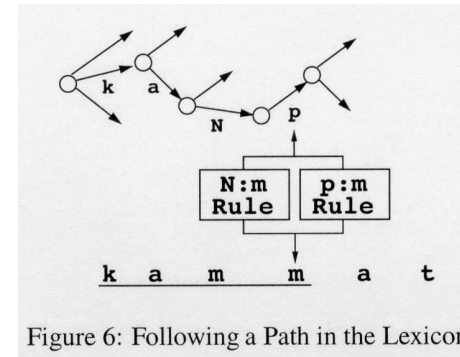
Regel (1') entspricht der obigen Ersetzungsregel (1).

Regel (2') unterscheidet sich von (2) darin, dass sie nur auf den lexikalischen Kontext referiert.

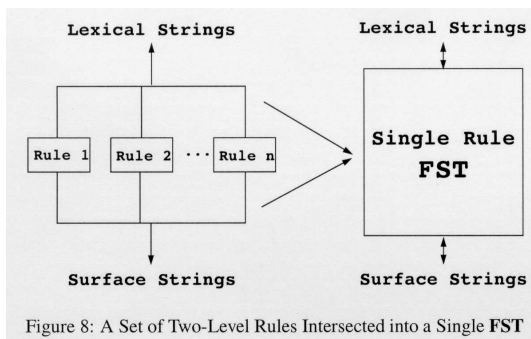
Die beiden Constraints (1') und (2') sind voneinander unabhängig und sind daher parallel anwendbar; (1) und (2) müssen geordnet werden. Wenn sie parallel arbeiten, haben sie denselben Effekt wie die o.g. Kaskade.



Verzahnung mit dem Lexikon zur Beschränkung der Übergenerierung:
Die zulässigen Symbole oben werden in jedem Schritt durch Lexikonzugriff beschränkt (Tandem). Das Lexikon wirkt kontinuierlich wie ein Filter.

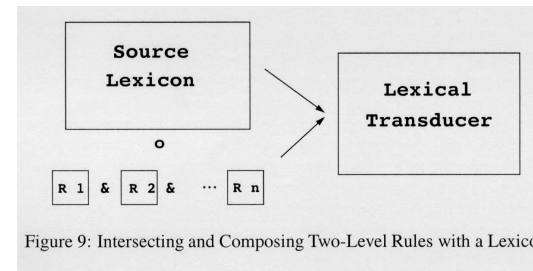


Verschmelzung von Zwei-Ebenen-Regeln



übereinstimmen.
Zusammen: $x^n \rightarrow x$; ist regulär!

Vermeidung kombinatorischer Effekte durch Verzahnung mit dem Lexikon:



Auch hier: Problem der kombinatorischen Explosion

Anmerkung:

I.a. ist der Durchschnitt zweier regulärer Relationen nicht regulär, da Epsilon-Übergänge synchronisiert werden müssen — R1 und R2 müssen in der Lokation von Epsilons

Aufbau der (morpho)phonologischen Regeln

Morphophonologische Komponente

Genauer betrachtet:

- Die lexikalische und die Oberflächen-Ebene haben je ein eigenes Alphabet
- Auf der lexikalischen Ebene gibt es diakritische Zeichen; sie repräsentieren keinen Lautwert, aber phonologisch relevante Informationen, z.B. Morphgrenze, Akzent
- Die Symbole der beiden Alphabete werden explizit einander zugeordnet (Defaults; Diakritika \Rightarrow 0)
- Morphologische Regularitäten werden als reihenfolgeunabhängige Ersetzungsregeln codiert (parallele Anwendbarkeit)

Operator	Example	Translation
\Leftarrow	$a:b \Leftarrow c_d$	a is always realized as b in the context c_d
\Rightarrow	$a:b \Rightarrow c_d$	a is realized as b only in the context c_d
\Leftrightarrow	$a:b \Leftrightarrow c_d$	a is realized as b in c_d and nowhere else
$/\Leftarrow$	$a:b /\Leftarrow c_d$	a is never realized as b in the context c_d

Table 1: Rule operators

Figure 12: Transducer for $p:m \Leftrightarrow :m _$

Figure 13: Transduction with parallel transducer:

k	a	N	p	a	n	lexical string
0	0	0	2	0	0	Transducer in figure 4
0	0	0	1	1	0	Transducer in figure 12
k	a	m	m	a	n	surface string

Figure 14: Non-allowed correspondence

k	a	N	p	a	n	lexical string
0	0	0	2	0	Transducer in figure 4	
0	0	0	1	blocks!	Transducer in figure 12	
k	a	m	p	a	n	surface string

Darstellung morphophonologischer Phänomene

- **Elision:** Ein Zeichen, das auf der lexikalischen Ebene vorhanden ist, wird an der Oberfläche nicht realisiert, d.h. Tilgung eines Phonems (Graphems) durch $x:0$ unter bestimmten durch die Regel spezifizierten Bedingungen
Beispiel: $ras+st \Rightarrow rast$
- **Epenthese:** Ein Zeichen, das auf der lexikalischen Ebene nicht vorhanden ist, wird an der Oberfläche realisiert, d.h. Einfügung eines Morphems (Graphems) durch $x:y$ oder $0:y$ unter ...
Beispiel: $badst \Rightarrow badest$
- Da sich eine Regel nur auf genau ein Paar von Zeichen beziehen kann, müssen Veränderungen, die Zeichenketten betreffen, durch "interagierende" Regeln realisiert werden

Der Transducer für Regel (2') unterscheidet sich von dem für Regel (2) in den Kanten von und zum Zustand 1 (Realisierung von p als m). Die Transducer für (1) und (1') sind identisch.

Hinweis: In Zwei-Ebenen-Regeln kann die Kontextspezifikation komplizierter sein als in den anfangs benutzten Ersetzungsregeln; sie erlaubt reguläre Ausdrücke.

Verarbeitung der Zwei-Ebenen-Regeln: Automaten

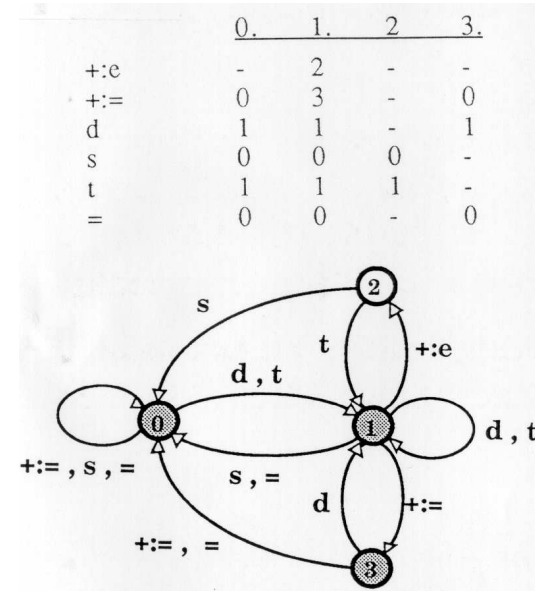
Beispielregel:

(Vereinfachte) Regel für e-Epenthese an Morphgrenze im Deutschen

$$+ : e \Leftrightarrow \{d, t\} \cdot \{s, t\}$$

(= ist 'wildcard')

Anwendungsbeispiel: badst \Rightarrow badest



Morphotaktische Komponente (1)

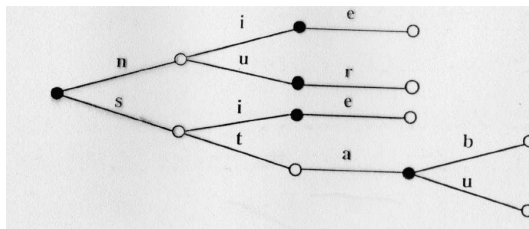
Die Morphologie arbeitet auf einem Lexikon von Morphemen, ebenfalls dargestellt durch einen endlichen Automaten.

Darstellung morphologischer Regularitäten durch "Fortsetzungsklassen"
(legale Nachfolger für jeden Morph)

Fortsetzungsklassen = reguläre Grammatiken!

Darstellung des Morphlexikons als Graphenbaum;

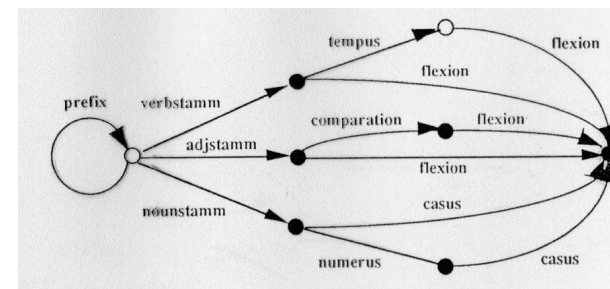
Speicherung der morphosyntaktischen Information in Knoten



Morphotaktische Komponente (2)

Fortsetzungsklassen als endlicher Automat

Beispiel: Ausschnitt der Flexionsmorphologie



Probleme der Zwei-Ebenen-Morphologie

Rechtfertigung mancher Prämissen ist umstritten

1. Voraussetzungen der Beschreibung der Morphologie durch

Fortsetzungsklassen:

Wortbildung

- ist durch reguläre Grammatik adäquat beschreibbar
z.B. Klammerungen wie Partizipialkonstruktion
ge + Verbstamm + Partizipialsuffix
- beschränkt sich auf Konkatenation
aber: \$mÜtter\$ ⇒ Omutter0, Omütter0

2. Morphologie und (Morpho)phonologie sind voneinander unabhängige

Komponenten

— somit ist nur eine indirekte Modellierung von wechselseitigen Abhängigkeiten möglich.

Fälle wie die Schwa-Epenthese

sandte, sendete; Dunkeln, dunklen

müssen durch zusätzliche diakritische Zeichen realisiert werden, die aber nicht phonologisch motiviert sind, sondern Kontrollinformation codieren (z.B. für Umlautung, s.o.)