Morphologie und SFST Tools

Helmut Schmid

Centrum für Informations- und Sprachverarbeitung
Ludwig-Maximilians-Universität München

Stand: 10. April 2026

Morphologie

Morphologie untersucht die Struktur von Wörtern

- Zerlegung in Morpheme: Formel-ab-leit-ung-en
  Morpheme sind die kleinsten bedeutungstragenden Einheiten

- weitergehende Analyse:

Basislemma	leiten/V
Präfigierung	ab/PART leiten/V
Nominalisierung	ab/PART leiten/V ung/NN
Komposition	Formel/NN ab/PART leiten/V ung/NN
Flexion	Formel/NN ab/PART leiten/V ung/NN en/PI

Morphologische Prozesse

Flexion

- Flexionsmorpheme markieren syntaktische Eigenschaften
  rechn-est 2. Sg. Präsens
  ge-rechn-et Partizip Perfekt
  rechn-en Infinitiv, 1. Pl. Präsens, 3. Pl. Präsens
- Die Flexionsmorpheme werden an den Wortstamm angefügt
- Flexion von Verben: Konjugation
- Flexion von Nomen, Adjektiven: Deklination

Deutsch ist eine fusionale Sprache, d.h. ein Morphem repräsentiert mehrere syntaktische Merkmale → "-est" - 2. Sg. Präsens

Deutsch zeigt außerdem Synkretismus, d.h. eine Form hat mehrere Analysen → rechnen

Morphologische Prozesse

Flexion

- Flexionsmorpheme markieren syntaktische Eigenschaften
  rechn-est 2. Sg. Präsens
  ge-rechn-et Partizip Perfekt
  rechn-en Infinitiv, 1. Pl. Präsens, 3. Pl. Präsens
- Die Flexionsmorpheme werden an den Wortstamm angefügt
- Flexion von Verben: Konjugation
- Flexion von Nomen, Adjektiven: Deklination

Deutsch ist eine fusionale Sprache, d.h. ein Morphem repräsentiert mehrere syntaktische Merkmale → "-est" - 2. Sg. Präsens

Deutsch zeigt außerdem Synkretismus, d.h. eine Form hat mehrere Analysen → rechnen

Morphologische Prozesse

Türkisch hat eine sehr komplexe Flexion (agglutinierende Sprache)

(yemek) pişirdiler - Sie kochten (Essen).

Genauer: Sie bewirkten, dass (Essen) gekocht wurde.

piş	gekocht werden (Stamm)
ir	Kausativ
di	Vergangenheit
ler	3. Person Plural

Morphologische Prozesse

Derivation

Ableitung neuer Wortformen mit anderer Wortart oder Bedeutung aus vorhandenen Wortformen

setzen → übersetzen → übersetzbar → unübersetzbar → Unübersetzbarkeit

- über- Verbpräfix
- -bar leitet ein Adjektiv aus einem Verb ab
- un- negierendes Adjektivpräfix
- keit leitet ein Nomen aus einem Adjektiv ab

Derivationsmorpheme können auch leer sein: wohnen → das Wohnen (Konversion)

Neoklassische Wortbildung

- existiert parallel zur normalen deutschen Wortbildung
- in vielen europäischen Sprachen
- Grund: Latein war lange die Sprache der Wissenschaft. Lateinische (und griechische) Fachbegriffe wurden ins Deutsche übernommen und angepasst

Deutsch	Englisch	Französisch
Norm	norm	norme
normal	normal	normal
Normalität	normality	normalité
abnormal	abnormal	anormal

Wie könnte das Wort Inaktivität gebildet worden sein?

Inaktivität → inaktiv → aktiv → akt(ion)/ag(ieren)

analoges deutsches Wort mit ganz anderen Morphemen:

Untätigkeit → untätig → tätig → Tat/tun

Neoklassische Wortbildung

- existiert parallel zur normalen deutschen Wortbildung
- in vielen europäischen Sprachen
- Grund: Latein war lange die Sprache der Wissenschaft. Lateinische (und griechische) Fachbegriffe wurden ins Deutsche übernommen und angepasst

Deutsch	Englisch	Französisch
Norm	norm	norme
normal	normal	normal
Normalität	normality	normalité
abnormal	abnormal	anormal

Wie könnte das Wort Inaktivität gebildet worden sein?

Inaktivität → inaktiv → aktiv → akt(ion)/ag(ieren)

analoges deutsches Wort mit ganz anderen Morphemen:

Untätigkeit → untätig → tätig → Tat/tun

Neoklassische Wortbildung

- existiert parallel zur normalen deutschen Wortbildung
- in vielen europäischen Sprachen
- Grund: Latein war lange die Sprache der Wissenschaft. Lateinische (und griechische) Fachbegriffe wurden ins Deutsche übernommen und angepasst

Deutsch	Englisch	Französisch
Norm	norm	norme
normal	normal	normal
Normalität	normality	normalité
abnormal	abnormal	anormal

Wie könnte das Wort Inaktivität gebildet worden sein?

Inaktivität → inaktiv → aktiv → akt(ion)/ag(ieren)

analoges deutsches Wort mit ganz anderen Morphemen:

Untätigkeit → untätig → tätig → Tat/tun

Neoklassische Wortbildung

- existiert parallel zur normalen deutschen Wortbildung
- in vielen europäischen Sprachen
- Grund: Latein war lange die Sprache der Wissenschaft. Lateinische (und griechische) Fachbegriffe wurden ins Deutsche übernommen und angepasst

Deutsch	Englisch	Französisch
Norm	norm	norme
normal	normal	normal
Normalität	normality	normalité
abnormal	abnormal	anormal

Wie könnte das Wort Inaktivität gebildet worden sein?

Inaktivität → inaktiv → aktiv → akt(ion)/ag(ieren)

analoges deutsches Wort mit ganz anderen Morphemen:

Untätigkeit → untätig → tätig → Tat/tun

Morphologische Prozesse

Komposition

- Bildung einer neuen Wortform aus zwei vorhandenen
  Donau-dampf-schiff-fahrt-s-Kapitän
- Das '-s-' ist ein Fugenmorphem
- Deutsch ist für seine komplexen Komposita bekannt
- Typen deutscher Wortstämme
  - Derivationsstamm (oft mit Umlautung)
    bäuer-lich
  - Kompositionsstämme:

    Bauers-frau, Bauern-krieg, Bauer-beruf
    Baumes-wipfel, Bäume-fällen, baum-lang

    Wie werden diese Stämme gebildet?

  - Einige Derivationsendungen kombinieren mit Kompositionsstämmen:
    taten-los, zahlen-mäßig, bauern-haft (vs. bäuer-lich)

Morphologische Prozesse

Komposition

- Bildung einer neuen Wortform aus zwei vorhandenen
  Donau-dampf-schiff-fahrt-s-Kapitän
- Das '-s-' ist ein Fugenmorphem
- Deutsch ist für seine komplexen Komposita bekannt
- Typen deutscher Wortstämme
  - Derivationsstamm (oft mit Umlautung)
    bäuer-lich
  - Kompositionsstämme:

    Bauers-frau, Bauern-krieg, Bauer-beruf
    Baumes-wipfel, Bäume-fällen, baum-lang

    Wie werden diese Stämme gebildet?

  - Einige Derivationsendungen kombinieren mit Kompositionsstämmen:
    taten-los, zahlen-mäßig, bauern-haft (vs. bäuer-lich)

Morphologische Prozesse

Klitisierung

Ein Klitik ist ein Wort, das mit einem benachbarten Wort verschmolzen (und dabei eventuell reduziert) wurde.

I've	I have (engl.)
C'est	Ce est (franz.)
fermarla	fermar la (ital.)

Morphologische Prozesse

nicht-konkatenative Morphologie

- Hier werden Morpheme nicht einfach aneinandergereiht
- Beispiel: Root-and-Pattern-Morphologie (semitische Sprachen)
  - Wurzel lmd (lernen, Hebräisch)
  - plus das Muster CaCaC (für Aktiv)
  - ergibt das Wort lamad
  - Für die drei C's werden die drei Konsonanten eingesetzt

Mehr zu Morphologie

- Affixe schließen Präfixe, Suffixe, Infixe und Zirkumfixe ein.
  - ge-...-t in getaucht ist ein Zirkumfix (Partizip Perfekt)
  - In abgetaucht stellt ge- ein Infix dar.

- Einige morphologische Prozesse sind nicht mehr produktiv (d.h. sie werden nicht mehr zur Bildung neuer Wörter benutzt)
  - -sam in "einsam", "kleidsam", "arbeitsam"
  - -sal in "Trübsal", "Labsal", "Mühsal"

Flexionsklassen

- Flektierende Wörter kann man in Flexionsklassen einteilen

  Lateinische a-Deklination: casa, casae, casae, casam, casā, ...
  Lateinische o-Deklination: avus, avī, avō, avum, avō, ...
  Lateinische u-Deklination: portus, portūs, portuī, portum, ...

- Alle Wörter einer Flexionsklasse werden nach demselben Schema flektiert.

Tiwa-Morphologie (Sprache von Pueblo-Indianern)

temiban	I went
amiban	you went
temiwe	I am going
mimiay	he was going
tewanban	I came
tewanhi	I will come

Bedeutung des Präfixes 'te-'?

Bedeutung des Präfixes 'a-'? you

Bedeutung des Präfixes 'mi-'? he

Bedeutung des Suffixes '-ban'? past

Bedeutung des Suffixes '-we? present progr.

Bedeutung des Suffixes '-ay? past progr.

Bedeutung des Suffixes '-hi? future

Wie sagt man in Tiwa: "he went" mimiban

regulärer Ausdruck für die Flexion: (te|a|mi) (mi|wan) (ban|we|ay|hi)

Tiwa-Morphologie (Sprache von Pueblo-Indianern)

temiban	I went
amiban	you went
temiwe	I am going
mimiay	he was going
tewanban	I came
tewanhi	I will come

Bedeutung des Präfixes 'te-'?	I
Bedeutung des Präfixes 'a-'?	you
Bedeutung des Präfixes 'mi-'?	he
Bedeutung des Suffixes '-ban'?	past
Bedeutung des Suffixes '-we?	present progr.
Bedeutung des Suffixes '-ay?	past progr.
Bedeutung des Suffixes '-hi?	future

Wie sagt man in Tiwa: "he went" mimiban

regulärer Ausdruck für die Flexion: (te|a|mi) (mi|wan) (ban|we|ay|hi)

Tiwa-Morphologie (Sprache von Pueblo-Indianern)

temiban	I went
amiban	you went
temiwe	I am going
mimiay	he was going
tewanban	I came
tewanhi	I will come

Bedeutung des Präfixes 'te-'?	I
Bedeutung des Präfixes 'a-'?	you
Bedeutung des Präfixes 'mi-'?	he
Bedeutung des Suffixes '-ban'?	past
Bedeutung des Suffixes '-we?	present progr.
Bedeutung des Suffixes '-ay?	past progr.
Bedeutung des Suffixes '-hi?	future

Wie sagt man in Tiwa: "he went" mimiban

regulärer Ausdruck für die Flexion: (te|a|mi) (mi|wan) (ban|we|ay|hi)

Tiwa-Morphologie (Sprache von Pueblo-Indianern)

temiban	I went
amiban	you went
temiwe	I am going
mimiay	he was going
tewanban	I came
tewanhi	I will come

Bedeutung des Präfixes 'te-'?	I
Bedeutung des Präfixes 'a-'?	you
Bedeutung des Präfixes 'mi-'?	he
Bedeutung des Suffixes '-ban'?	past
Bedeutung des Suffixes '-we?	present progr.
Bedeutung des Suffixes '-ay?	past progr.
Bedeutung des Suffixes '-hi?	future

Wie sagt man in Tiwa: "he went" mimiban

regulärer Ausdruck für die Flexion: (te|a|mi) (mi|wan) (ban|we|ay|hi)

Tiwa-Morphologie (Sprache von Pueblo-Indianern)

temiban	I went
amiban	you went
temiwe	I am going
mimiay	he was going
tewanban	I came
tewanhi	I will come

Bedeutung des Präfixes 'te-'?	I
Bedeutung des Präfixes 'a-'?	you
Bedeutung des Präfixes 'mi-'?	he
Bedeutung des Suffixes '-ban'?	past
Bedeutung des Suffixes '-we?	present progr.
Bedeutung des Suffixes '-ay?	past progr.
Bedeutung des Suffixes '-hi?	future

Wie sagt man in Tiwa: "he went" mimiban

regulärer Ausdruck für die Flexion: (te|a|mi) (mi|wan) (ban|we|ay|hi)

Tiwa-Morphologie (Sprache von Pueblo-Indianern)

temiban	I went
amiban	you went
temiwe	I am going
mimiay	he was going
tewanban	I came
tewanhi	I will come

Bedeutung des Präfixes 'te-'?	I
Bedeutung des Präfixes 'a-'?	you
Bedeutung des Präfixes 'mi-'?	he
Bedeutung des Suffixes '-ban'?	past
Bedeutung des Suffixes '-we?	present progr.
Bedeutung des Suffixes '-ay?	past progr.
Bedeutung des Suffixes '-hi?	future

Wie sagt man in Tiwa: "he went" mimiban

regulärer Ausdruck für die Flexion: (te|a|mi) (mi|wan) (ban|we|ay|hi)

Tiwa-Morphologie (Sprache von Pueblo-Indianern)

temiban	I went
amiban	you went
temiwe	I am going
mimiay	he was going
tewanban	I came
tewanhi	I will come

Bedeutung des Präfixes 'te-'?	I
Bedeutung des Präfixes 'a-'?	you
Bedeutung des Präfixes 'mi-'?	he
Bedeutung des Suffixes '-ban'?	past
Bedeutung des Suffixes '-we?	present progr.
Bedeutung des Suffixes '-ay?	past progr.
Bedeutung des Suffixes '-hi?	future

Wie sagt man in Tiwa: "he went" mimiban

regulärer Ausdruck für die Flexion: (te|a|mi) (mi|wan) (ban|we|ay|hi)

Tiwa-Morphologie (Sprache von Pueblo-Indianern)

temiban	I went
amiban	you went
temiwe	I am going
mimiay	he was going
tewanban	I came
tewanhi	I will come

Bedeutung des Präfixes 'te-'?	I
Bedeutung des Präfixes 'a-'?	you
Bedeutung des Präfixes 'mi-'?	he
Bedeutung des Suffixes '-ban'?	past
Bedeutung des Suffixes '-we?	present progr.
Bedeutung des Suffixes '-ay?	past progr.
Bedeutung des Suffixes '-hi?	future

Wie sagt man in Tiwa: "he went" mimiban

regulärer Ausdruck für die Flexion: (te|a|mi) (mi|wan) (ban|we|ay|hi)

Tiwa-Morphologie (Sprache von Pueblo-Indianern)

temiban	I went
amiban	you went
temiwe	I am going
mimiay	he was going
tewanban	I came
tewanhi	I will come

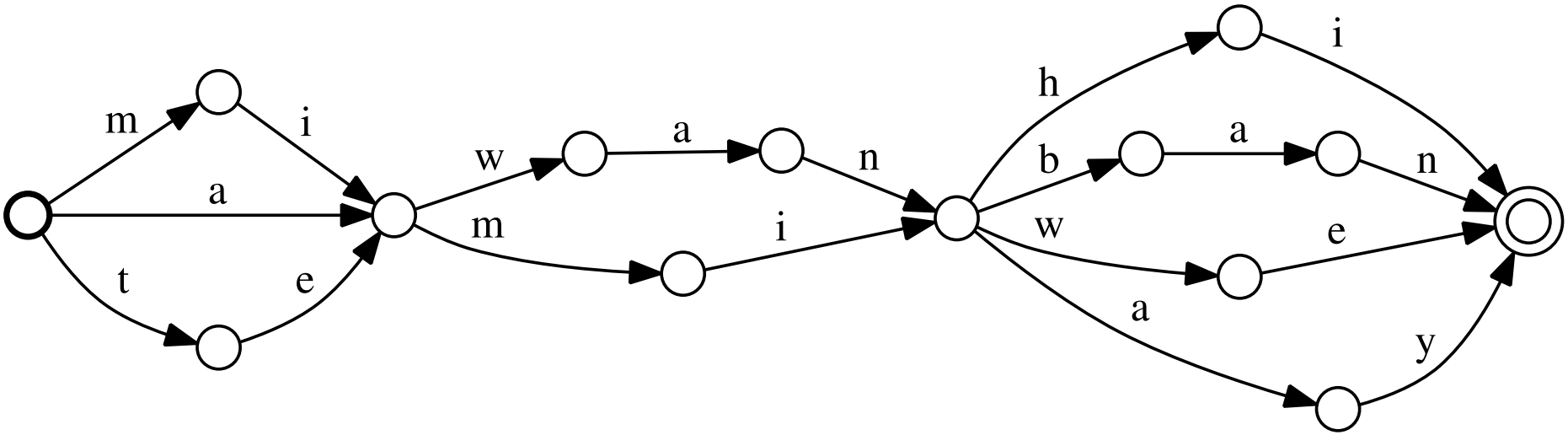
Bedeutung des Präfixes 'te-'?	I
Bedeutung des Präfixes 'a-'?	you
Bedeutung des Präfixes 'mi-'?	he
Bedeutung des Suffixes '-ban'?	past
Bedeutung des Suffixes '-we?	present progr.
Bedeutung des Suffixes '-ay?	past progr.
Bedeutung des Suffixes '-hi?	future

Wie sagt man in Tiwa: "he went" mimiban

regulärer Ausdruck für die Flexion: (te|a|mi) (mi|wan) (ban|we|ay|hi)

Endliche Automaten

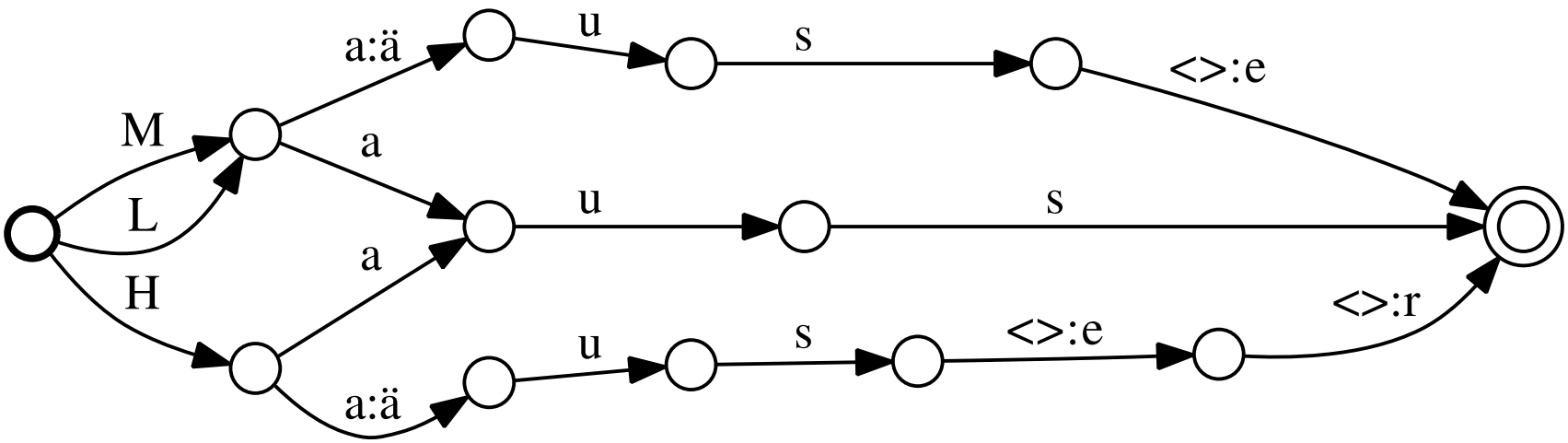
Der reguläre Ausdruck (te|a|mi) (mi|wan) (ban|we|ay|hi) kann in einen endlichen Automaten transformiert werden:



Dieser akzeptiert alle Wörter, die der reguläre Ausdruck bilden kann.

Wir wollen aber nicht nur korrekte Wörter erkennen, sondern auch eine Analyse ausgeben → endliche Transduktoren (finite-state transducer)

Finite State Transducer



Transducer mit den Abbildungen

Haus → Haus
Häuser → Haus
Maus → Maus
Mäuse → Maus
Laus → Laus
Läuse → Laus

Finite State Transducer

- endlicher Automat mit Symbolpaaren an den Übergängen.
- definiert eine Abbildung zwischen zwei regulären Sprachen (reguläre Relation)
- in beide Richtungen (Analyse, Generierung) anwendbar
- Standardmethode zur Implementierung morphologischer Analysewerkzeuge
- FSTs können durch erweiterte reguläre Ausdrücke definiert werden

  (M:M|L:L) a:ä u:u s:s <>:e bildet Mäuse/Läuse auf Maus/Laus ab

  <> ist das leere Symbol

  → Aufgabe 2
  → FST-Programmiersprache

SFST-Werkzeuge

- Programmiersprache für FSTs
- Compiler, der die Programme in Transducer übersetzt
- Programme zur Anwendung von FSTs (Interpreter)
- Beispiel:

```
> echo 'Maus | Laus | (M:M|L:L) a:ä u:u s:s <>:e' > test.fst
> fst-compiler-utf8 test.fst test.a
> fst-mor test.a
analyze> Maus
Maus
analyze> Mäuse
Maus
analyze>          # Leerzeile schaltet auf Generierung um
generate> Maus
Maus
Mäuse
```

Die SFST-Programmiersprache

Reguläre Ausdrücke

- a:b *b* wird als *a* analysiert
- a Abkürzung für a:a
- Konkatenation a b:B c
- Kleene's Operator a*
- Plus Operator b+
- Optionalität <>:e ?
- Klammern (ab)*
- Leerzeichen werden ignoriert.
- Werden die Zeichen :+*?()[ ] und Leerzeichen als solche benötigt, müssen sie mit einem \ gequotet werden.

Die SFST-Programmiersprache

weitere Operatoren

- Disjunktion a:a | a:b | c
- [a-c]:[A-C] kurz für a:A | b:B | c:C
- [a-c] kurz für [a-c]:[a-c]
- {abc}:{AB} kurz für a:A b:B c:<>
- Konjunktion [ab]*abba[ab]* & [ab]*baab[ab]*
- Negation [ab]*abba[ab]* & ![ab]*baab[ab]*
  (äquivalent zu [ab]*abba[ab]* - [ab]*baab[ab]*)

Kommentare werden mit % eingeleitet und enden am Zeilenende.

Variablen beginnen und enden mit $ (bspw. var) und speichern einen regulären Ausdruck (Transducer).

Kompositionsoperator

`$T1$ || $T2$`

- T1 und T2 werden (in Generierungsrichtung) hintereinander ausgeführt.
- Die Ausgabe von T1 bildet die Eingabe von T2
- Der Compiler erzeugt einen Transducer, der dieselbe Abbildung in einem Schritt ausführt.

Beispiel: `[a-z]:[A-Z]* || [A-Z]:[a-z]*`

Was macht dieser Transducer?

Der Transducer ist identisch zu `[a-z]*`

Kompositionsoperator

`$T1$ || $T2$`

- T1 und T2 werden (in Generierungsrichtung) hintereinander ausgeführt.
- Die Ausgabe von T1 bildet die Eingabe von T2
- Der Compiler erzeugt einen Transducer, der dieselbe Abbildung in einem Schritt ausführt.

Beispiel: `[a-z]:[A-Z]* || [A-Z]:[a-z]*`

Was macht dieser Transducer?

Der Transducer ist identisch zu `[a-z]*`

Das Alphabet

Der Befehl `ALPHABET = [a-z]:[A-Z] [a-z]:[a-z]` definiert ein Alphabet.

Auf der rechten Seite kann ein beliebiger Transducerausdruck stehen.

Das Alphabet umfasst alle Symbolpaare, die an Übergängen des Transducers auftauchen.

Das Wildcardsymbol "." steht für die Disjunktion aller Symbolpaare im Alphabet.

"a:." steht für die Disjunktion aller Symbolpaare im Alphabet mit "a" auf der linken Seite.

Was macht der Transducer ".*" bei obiger Definition des Alphabetes?

Das Alphabet

Der Befehl `ALPHABET = [a-z]:[A-Z] [a-z]:[a-z]` definiert ein Alphabet.

Auf der rechten Seite kann ein beliebiger Transducerausdruck stehen.

Das Alphabet umfasst alle Symbolpaare, die an Übergängen des Transducers auftauchen.

Das Wildcardsymbol "." steht für die Disjunktion aller Symbolpaare im Alphabet.

"a:." steht für die Disjunktion aller Symbolpaare im Alphabet mit "a" auf der linken Seite.

Was macht der Transducer ".*" bei obiger Definition des Alphabetes?

SFST-Programme

Programmstruktur

Ein SFST-Programm besteht aus einer Folge von Variablen-Definitionen, Alphabet-Definitionen und anderen Befehlen gefolgt von einem einzigen Ergebnisausdruck, dessen Wert nach der Kompilierung in der Ausgabedatei gespeichert wird.

Aus einfachen Transducern (regulären Ausdrücken) werden schrittweise immer komplexere Transducer aufgebaut, bis der gewünschte Ergebnis-Transducer fertiggestellt ist.

SFST-Beispiele

```
ALPHABET = [a-zA-Z]
```

```
[a-zA-Z]:[A-Za-z]+
```

Welche “Analyse” wird ausgegeben für “LaTeX”?

```
[ab]* - .*ba.*
```

Welche Strings akzeptiert dieser Automat?

```
.* {ll}:{LL}? .*
```

“Analyse” von “Hallo”?

“Analyse” von “HaLLo”?

```
{Hallo}:{<>Hallo}
```

“Analyse” von “Hallo”?

```
{Hallo}:{<>Hallo} & Hallo
```

Ausgabe für “Hallo”?

SFST-Beispiele

```
ALPHABET = [a-zA-Z]
```

```
[a-zA-Z]:[A-Za-z]+
```

Welche “Analyse” wird ausgegeben für “LaTeX”?

```
[ab]* - .*ba.*
```

Welche Strings akzeptiert dieser Automat?

```
.* {ll}:{LL}? .*
```

“Analyse” von “Hallo”?

“Analyse” von “HaLLo”?

```
{Hallo}:{<>Hallo}
```

“Analyse” von “Hallo”?

```
{Hallo}:{<>Hallo} & Hallo
```

Ausgabe für “Hallo”?

SFST-Beispiele

```
ALPHABET = [a-zA-Z]
```

```
[a-zA-Z]:[A-Za-z]+
```

Welche "Analyse" wird ausgegeben für "LaTeX"?

```
[ab]* - .*ba.*
```

Welche Strings akzeptiert dieser Automat?

```
.* {ll}:{LL}? .*
```

"Analyse" von "Hallo"?

"Analyse" von "HaLLo"?

```
{Hallo}:{<>Hallo}
```

"Analyse" von "Hallo"?

```
{Hallo}:{<>Hallo} & Hallo
```

Ausgabe für "Hallo"?

SFST-Beispiele

```
ALPHABET = [a-zA-Z]
```

```
[a-zA-Z]:[A-Za-z]+
```

Welche "Analyse" wird ausgegeben für "LaTeX"?

```
[ab]* - .*ba.*
```

Welche Strings akzeptiert dieser Automat?

```
.* {ll}:{LL}? .*
```

"Analyse" von "Hallo"?

"Analyse" von "HaLLo"?

```
{Hallo}:{<>Hallo}
```

"Analyse" von "Hallo"?

```
{Hallo}:{<>Hallo} & Hallo
```

Ausgabe für "Hallo"?

SFST-Beispiele

```
ALPHABET = [a-zA-Z]
```

```
[a-zA-Z]:[A-Za-z]+
```

Welche "Analyse" wird ausgegeben für "LaTeX"?

```
[ab]* - .*ba.*
```

Welche Strings akzeptiert dieser Automat?

```
.* {ll}:{LL}? .*
```

"Analyse" von "Hallo"?
"Analyse" von "HaLLo"?

```
{Hallo}:{<>Hallo}
```

"Analyse" von "Hallo"?

```
{Hallo}:{<>Hallo} & Hallo
```

Ausgabe für "Hallo"?

SFST-Beispiele

```
ALPHABET = [a-zA-Z]
```

```
[a-zA-Z]:[A-Za-z]+
```

Welche "Analyse" wird ausgegeben für "LaTeX"?

```
[ab]* - .*ba.*
```

Welche Strings akzeptiert dieser Automat?

```
.* {ll}:{LL}? .*
```

"Analyse" von "Hallo"?
"Analyse" von "HaLLo"?

```
{Hallo}:{<>Hallo}
```

"Analyse" von "Hallo"?

```
{Hallo}:{<>Hallo} & Hallo
```

Ausgabe für "Hallo"?

Tokenisierung mit FSTs

Einfacher Beispieltransducer, der Textstrings tokenisiert.

Definition von Variablen

```
$Letter$ = [A-ZÄÖÜa-zäöüß]
$leftPunc$ = [("]
$rightPunc$ = [.,;:!?")]
```

Satzzeichen und Klammern an den Worträndern abtrennen

```
$X$ = ($leftPunc$ \ :<>)* $Letter$* (\ :<> $rightPunc$)*
```

Wie sieht der Transducer in X aus?

Ergebnisausdruck

```
($X$ \ +)* $X$
```

Tokenisierung mit FSTs

Einfacher Beispieltransducer, der Textstrings tokenisiert.

Definition von Variablen

```
$Letter$ = [A-ZÄÖÜa-zäöüß]
$leftPunc$ = ["]
$rightPunc$ = [.,;:!?")]
```

Satzzeichen und Klammern an den Worträndern abtrennen

```
$X$ = ($leftPunc$ \ :<>)* $Letter$* (\ :<> $rightPunc$)*
```

Wie sieht der Transducer in X aus?

Ergebnisausdruck

```
($X$ \ +)* $X$
```

Tokenisierung mit FSTs

Anwendung des Transducers

```
> fst-compiler-utf8 tokenize.fst tokenize.a
> fst-mor tokenize.a
analyze> Er kam, sah, und siegte.
Er kam , sah , und siegte .
```

Erkennung von Abkürzungen

Definition von Variablen

```
...
$all$ = $Letter$ $leftPunc$ $rightPunc$ [\ ]
```

Satzzeichen und Klammern an den Worträndern abtrennen

```
$X$ = ($leftPunc$ \ :<>)* $Letter$* (\ :<> $rightPunc$)*
$X$ = ($X$ \ +)* $X$
```

Abkürzungen wieder zusammenfügen

```
ALPHABET = $all$
$R$ =  (<>:\ ) _-> ((Fr|Hr|Prof|Dr|etc|usw) __ \.)

$R$ || $X$
```

```
analyze> "Heureka!", sagte Hr. Müller.

" Heureka ! " , sagte Hr. Müller .
```

Ersetzungsregeln

`(b:B) ^-> (a __ c)`	Ersetze b zwischen a und c durch B in Generierungsrichtung
`(b:B) _-> (A __ C)`	Ersetze B zwischen A und C durch b in Analyserichtung

Bei leerem Kontext wird immer ersetzt:

`(b:B) ^-> ()` `(b:B) _-> ()`

allgemeine Syntax: Ersetzung `^->` (linker Kontext `__` rechter Kontext)

Ersetzung ist ein beliebiger Transducer

linker/rechter Kontext muss ein Automat sein (d.h. darf kein ':' enthalten)

Weitere Operatoren

Der Operator `^$A$` macht die Analysesymbole im Transducer `$A$` identisch zu den Oberflächensymbolen (also `^a:b = b:b`)

Der Operator `_$A$` macht die Oberflächensymbole identisch zu den Analysesymbolen (also `_a:b = a:a`)

Der Operator `^_$A$` vertauscht die Oberflächen- und Analysesymbole (also `^_{ab}:{AB} = {AB}:{ab}`)

Der Operator `"Datei"` liest den Inhalt von "Datei" und bildet einen Transducer durch Disjunktion der eingelesenen Zeilen. In der eingelesenen Datei werden nur die Symbole `< > : \` als Operatoren interpretiert.

Der Operator `"<Datei>"` liest einen bereits kompilierten Transducer aus "Datei".

Zeichenketten der Form `<...>` werden als atomare Symbole behandelt wie einzelne Buchstaben.

Morphologie

Lexikoneinträge aus Datei einlesen, ein Wort pro Zeile

```
$nouns$ = "nouns"
```

Endungen und Analysen hinzufügen

```
$MORPH$ = $nouns$ <N> <sg>:<> | $nouns$ <N> <pl>:s
```

morphophonologische Regeln definieren (story<N>s → storie<N>s)

```
ALPHABET = [A-Za-z<N>]

$R1$ = (y:{ie}) ^-> ([^aeiou]__ <N>s)
$R2$ = (<N>:<>) ^-> ()
```

Regeln anwenden

```
$MORPH$ || $R1$ || $R2$
```

Ausführen der Morphologie

```
> fst-compiler-utf8 morph.fst morph.a
> fst-mor morph.a
analyze> lobbies
lobby<N><pl>
analyze> stores
store<N><pl>
analyze> houses
house<N><pl>
analyze>
generate> house<N><pl>
houses
generate> house<N><sg>
house
generate> q
```

Einlesen von Lexikondateien

```
$nouns$ = "nouns"
$verbs$ = "verbs"
$present-verbs$ = "present-verbs"
$past-verbs$ = "past-verbs"
$part-verbs$ = "part-verbs"
$adjectives$ = "adjectives"
$gradable-adjectives$ = "gradable-adjectives"

$MORPH$ = $nouns$ <N> (<sg>:<> | <pl>:s)

...
```

Anfügen der Analysen und Flexionsendungen

```
...
$MORPH$ = $MORPH$ |\
    $verbs$ <V> {<inf>}:{} |\
    $verbs$ <V> {<pres><n3s>}:{} |\
    $verbs$ <V> {<pres><3s>}:{s} |\
    $verbs$ <V> {<gerund>}:{ing} |\
    $verbs$ <V> {<past>}:{ed} |\
    $verbs$ <V> {<part>}:{ed} |\
    $present-verbs$ <V> {<inf>}:{} |\
    $present-verbs$ <V> {<pres><n3s>}:{} |\
    $present-verbs$ <V> {<pres><3s>}:{s} |\
    $present-verbs$ <V> {<gerund>}:{ing} |\
    $past-verbs$ <V> {<past>}:{} |\
    $part-verbs$ <V> {<part>}:{}

$MORPH$ = $MORPH$ |\
    $adjectives$ <ADJ> {<pos>}:{} |\
    $gradable-adjectives$ <ADJ> {<pos>}:{} |\
    $gradable-adjectives$ <ADJ> {<comp>}:{er} |\
    $gradable-adjectives$ <ADJ> {<sup>}:{est}
...
```

Morphophonologische Regeln

Folgende Phänomene sollen behandelt werden

story+s → stories
happy+er → happier
late+est → latest

Wie könnten die Regeln dafür aussehen?

```
...

ALPHABET = ^$MORPH$

$R1$ = (y:{ie}) ^-> ([^aeiou] __ [<N><V>]s)
$R2$ = (y:i) ^-> (__ <ADJ>e)
$R3$ = (e:<>) ^-> (__ [<ADJ><V>][ei])
$R4$ = ([<N><V><ADJ>]:<>) ^-> ()

$MORPH$ || $R1$ || $R2$ || $R3$ || $R4$
```

Morphophonologische Regeln

Folgende Phänomene sollen behandelt werden

story+s → stories
happy+er → happier
late+est → latest

Wie könnten die Regeln dafür aussehen?

```
...

ALPHABET = ^$MORPH$

$R1$ = (y:{ie}) ^-> ([^aeiou] __ [<N><V>]s)
$R2$ = (y:i) ^-> (__ <ADJ>e)
$R3$ = (e:<>) ^-> (__ [<ADJ><V>][ei])
$R4$ = ([<N><V><ADJ>]:<>) ^-> ()

$MORPH$ || $R1$ || $R2$ || $R3$ || $R4$
```

Morphophonologische Regeln

Bei der Bildung des englischen Komparativs oder Superlativs wird der letzte Konsonant des Adjektivs verdoppelt, wenn

- das Adjektiv nur eine Silbe umfasst und
- auf Konsonant-Vokal-Konsonant endet und
- der letzte Konsonant nicht "y" oder "w" ist

big	bigger	biggest
fat	fatter	fattest
thin	thinner	thinnest
grey	greyer	greyest
slow	slower	slowest
plain	plainer	plainest

Wie könnte eine Regel dafür aussehen?

```
({g}:{gg}|{t}:{tt}|{n}:{nn}) ^-> (<B> $cons$+ $vowel$ __ <ADJ>e)
```

<B> ist hier eine Grenzmarkierung, die eingefügt werden muss, um den Wortanfang matchen zu können. Siehe dazu auch Übungsaufgabe 2.

Morphophonologische Regeln

Bei der Bildung des englischen Komparativs oder Superlativs wird der letzte Konsonant des Adjektivs verdoppelt, wenn

- das Adjektiv nur eine Silbe umfasst und
- auf Konsonant-Vokal-Konsonant endet und
- der letzte Konsonant nicht "y" oder "w" ist

big	bigger	biggest
fat	fatter	fattest
thin	thinner	thinnest
grey	greyer	greyest
slow	slower	slowest
plain	plainer	plainest

Wie könnte eine Regel dafür aussehen?

```
({g}:{gg}|{t}:{tt}|{n}:{nn}) ^-> (<B> $cons$+ $vowel$ __ <ADJ>e)
```

<B> ist hier eine Grenzmarkierung, die eingefügt werden muss, um den Wortanfang matchen zu können. Siehe dazu auch Übungsaufgabe 2.

Kopier-Operationen

```
({g}:{gg}|{t}:{tt}|{n}:{nn})
```

Diese Ausdruck zum Kopieren eines Buchstabens wird sehr lang, wenn er alle Konsonanten umfassen soll. Die Lösung sind Kopiervariablen.

Variablen für Symbolmengen:

```
#vowel# = aeiou
#cons# = bcdfghjklmnpqrstvx
$vowel$ = [#vowel#]
$cons$ = [#cons#]
```

Kopiervariable:

```
#=c# = #cons#

{[#=c#]}:{[#=c#][#=c#]} ^-> (<B> $cons$+ $vowel$ __ <ADJ>e)
```

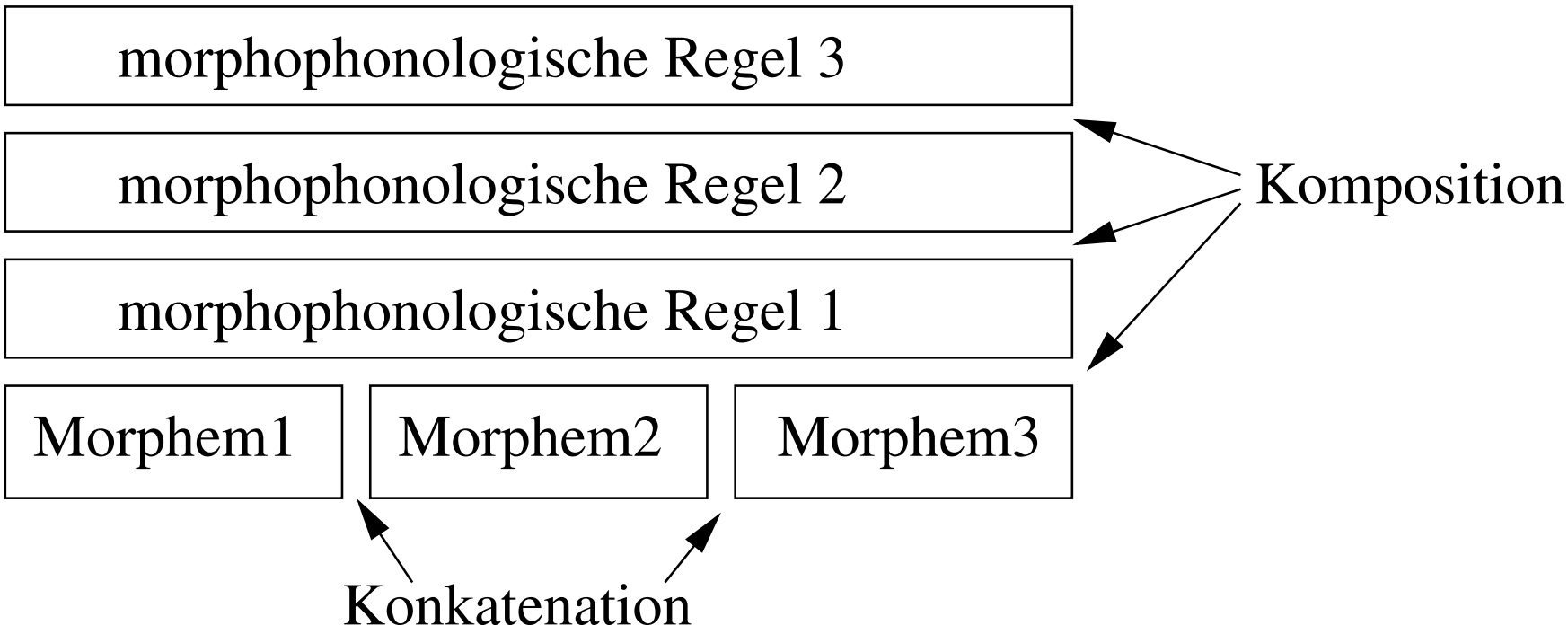
Entwicklung einer Morphologie

Vorgehensweise

- zu behandelnde Phänomene festlegen
  - Flexion, Derivation, Komposition?
  - welche Wortarten?
  - unregelmäßig gebildete Formen?
  - welche morphophonologische Regeln?
  - weitere Phänomene wie Klitika?
- Stammlexika anlegen
- Lexika einlesen
- ggf. Derivationsendungen hinzufügen
- ggf. Kompositums-Erstglieder hinzufügen
- Flexionsendungen hinzufügen
- morphophonologische Regeln anwenden

Entwicklung einer Morphologie

- Konkatenation der Morpheme
- Anwendung der morphophonologischen Regeln mit Komposition



Debugging

- Testwörter ausprobieren
- Zwischenergebnisse im Programm ausgeben mit dem Befehl
  `$var$ >> "tmp.a"`
  Dann den ausgegebenen Transducer untersuchen mit
  `> fst-generate tmp.a`
  `> fst-generate -l tmp.a` liefert die Strings der Analyseebene
  `> fst-generate -u tmp.a` Strings der Oberflächenebene
  Mit *grep* können interessante Strings herausgefiltert werden.
- Bei der Fehlersuche hilft es oft, die Lexika auf wenige zum Testen relevante Einträge zu reduzieren.
  $\Rightarrow$ schnellere Kompilierung, einfachere Fehlersuche

Beispiel

```
> cat test.fst
{Hut}:{Hüte}
> fst-compiler-utf8 test.fst test.a
test.fst: 2
> fst-print test.a
0   1   H   H
1   2   u   ü
2   3   t   t
3   4   <>  e
4
> fst-generate test.a
Hu:üt<>:e
```

Aufgabe bis nächste Woche

- SFST-Tutorial auf der Kursseite durcharbeiten
- Gedanken zur nächsten Übungsaufgabe machen