Computerlinguistik I

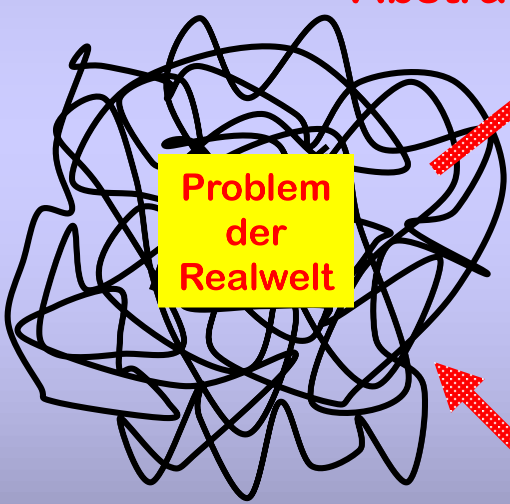
Vorlesung im WiSe 2018/19
(M-GSW-09)

Prof. Dr. Udo Hahn

Lehrstuhl für Computerlinguistik
Institut für Germanistische Sprachwissenschaft
Friedrich-Schiller-Universität Jena

http://www.julielab.de

Informatischer Problemlösungszyklus



Abstraktion

Abstraktes (computerlinguistisches) Modell

Datenstrukturen & Operationen

Algorithmus

Programmierspache(n)

Kodierung

Ausführung im Rechner

Rückkopplung (Evaluation)

Informatischer Problemlösungszyklus

- Modellbildung
  - Abstraktion von allen unwesentlichen Details der Problemstellung im Hinblick auf die algorithmische Lösung
  - Spezifikation der logischen Abhängigkeiten zwischen problemlösungsrelevanten Objekten
  - (computer)linguistisches Wissen

Informatischer Problemlösungszyklus

- Algorithmisierung
  - Übersetzung der modellbezogenen Spezifikation in
    - eine Menge von Objekten (Datenstrukturen) mit bestimmten Eigenschaften und Beziehungen zueinander
    - die erlaubten Operationen auf diesen Objekten
  - Algorithmus: (möglichst präzise) Beschreibung einer Folge zulässiger Operationen auf den Objekten, um das Problem zu lösen
  - Computerlinguistische Kernexpertise

Informatischer Problemlösungszyklus

- Kodierung (Programmierung)
  - Übersetzung der algorithmischen Spezifi-kation in Konstrukte einer (geeigneten) Programmiersprache
- Ausführung des Programms
  - Hier erst Bezug auf konkrete Maschinen (Datenstrukturen und Algorithmen sind abstrakte Konstruktionen)
  - Test-Modifikationszyklus ... Dokumentation !
  - Informatisches Know-How

Morphologische Prozesse: Flexion - Deflexion

- Kombination von Grundformen mit Flexionsaffixen (Kasus, Numerus, Tempus)
  - Deklination
    - Land: Land, Landes, Lande, Länder, Ländern
  - Konjugation
    - landen: lande, landest, landet, landeten , gelandet
- primär syntaktische, nur minimale semantische Information, keine grundlegenden Wortartwechsel

Morphologische Prozesse: Derivation - Dederivation

- Kombination von Grundformen mit Derivationsaffixen
  - Land: landen, verlanden, anlanden,
  - Land: Landung, Verlandung , Anlandung
  - Land: ländlich, verländlichen, Verländlichung
- modifizierende semantische Information, häufig mit Wortartwechsel verbunden

Morphologische Prozesse: Komposition - Dekomposition

- Kombination von Grundformen mit Grundformen (mittels Fugeninfixen)
  - Land: Landnahme, Landflucht, Landgang
  - Land: Heimatland, Ausland, Bauland
  - Land: Landesrekord, Landesverrat, Landsmann
  - Land: Inlandsflug, Landesratspräsidentengattin
- starke semantische Modifikation, fast keine Wortartwechsel
  - … aber: Rotkehlchen, Weichteile

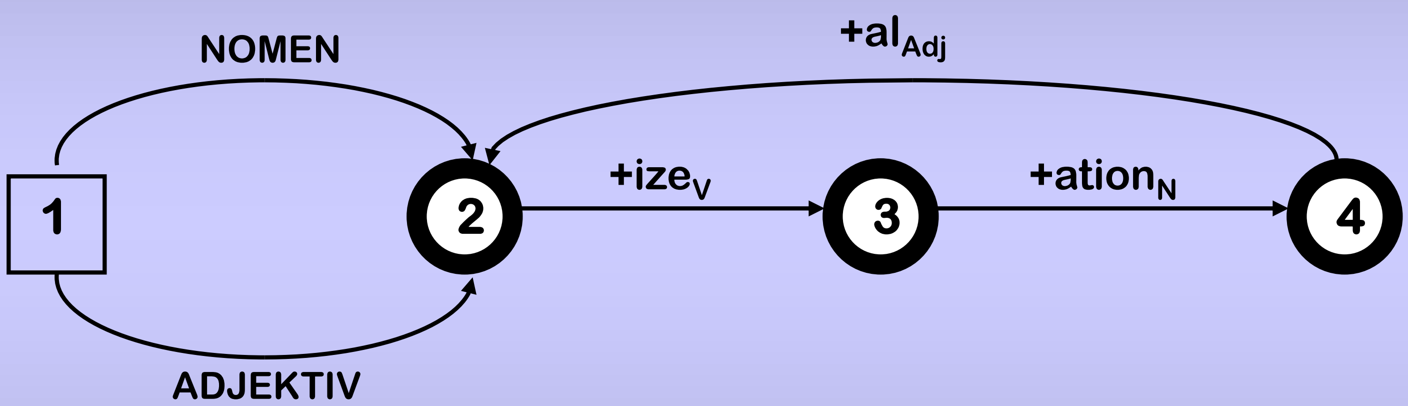
Lemmatisierung vs. Wort-Parsing

Eingabe	Lemma	Wort-Parse
Töchtern	Tochter	
Hauses	Haus	
sagte	sagen	
Spiegelungen	Spiegelung	
leichter	leicht	
verlängerte	verlängert	
	verlängern	

Lemmatisierung vs. Wort-Parsing

Eingabe	Lemma	Wort-Parse
Töchtern	Tochter	Tochter [+N, +FEM, +PL, +DAT]
Hauses	Haus	Haus [+N, +NEU, +SG, +GEN]
sagte	sagen	sagen [+V, +SG, {1P,3P}, +PAST]
Spiegelungen	Spiegelung	$[\text{Spiegel}]_{N}[\text{ung}]_{ds}$ [+N, +FEM, +PL, {NOM,GEN,DAT,AKK}]
leichter	leicht	leicht [+Adj, +POS, +MAS, +SG, +NOM] [+Adj, +KOM]
verlängerte	verlängert	$[\text{ver}]_{dp}[[\text{lang}]_{Adj}[\text{er}]_{ds}]_{Adj}[\text{t}]_{ds}$ [+Part, {MAS,FEM,NEU}, +SG, + DEF, +NOM] [+Part, {FEM,NEU}, +SG, + DEF, +AKK]
	verlängern	$[\text{ver}]_{dp}[[\text{lang}]_{Adj}[\text{er}]_{ds}]_{Adj}[\text{n}]_{ds}$ [+V, +SG, {1P,3P}, +PAST]

Automat für Dederivation

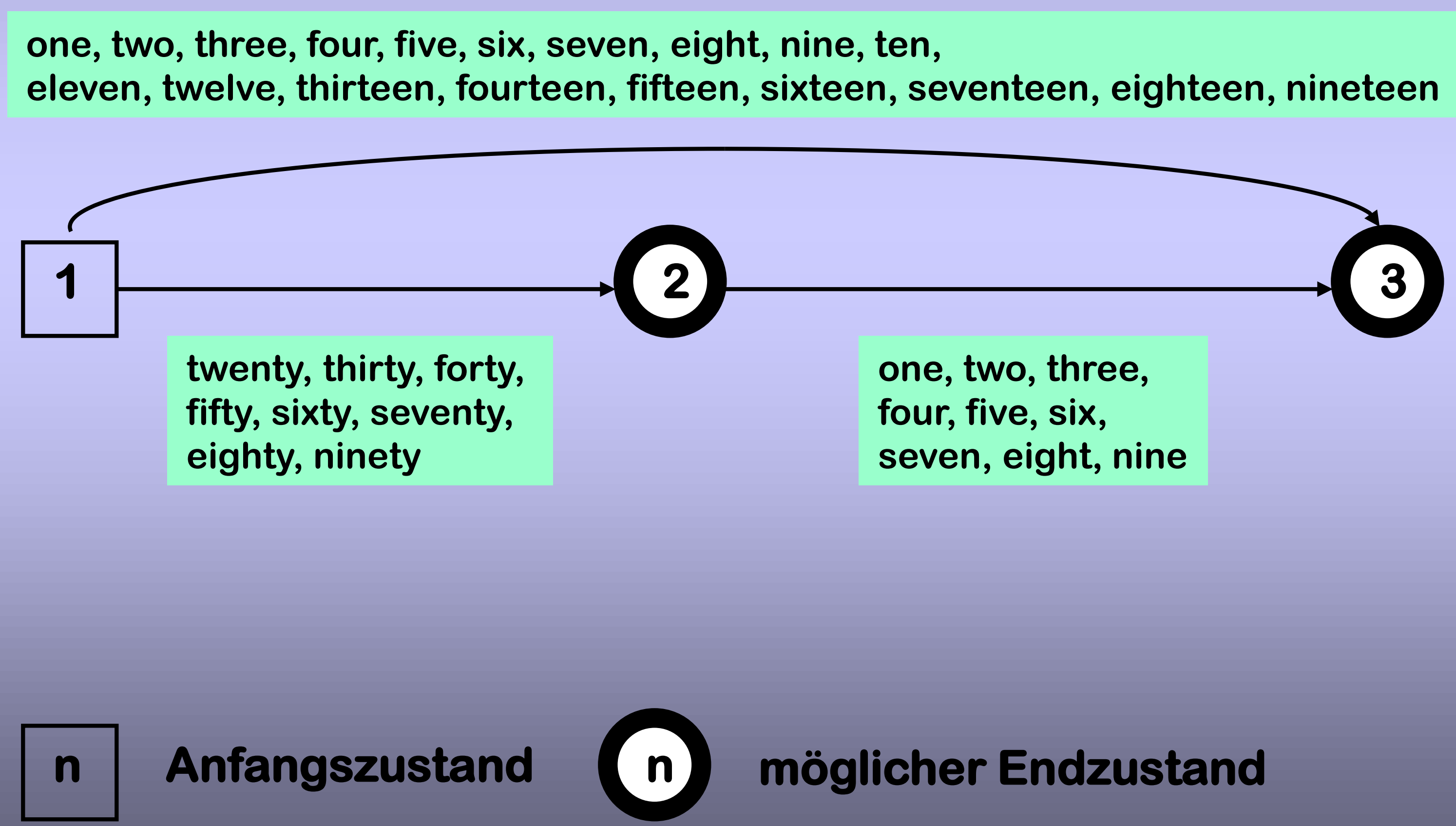


NOMEN: hospital, motor, category, ...

ADJEKTIV: moral, concrete, tender, ...

n Anfangszustand n möglicher Endzustand

Automat für englische Zahlen von 1 bis 99



Mengentheoretische Grundbegriffe

- **Die Zusammenfassung aller Elemente x, die eine Eigenschaft $\mathcal{E}$ haben, wird als Menge M bezeichnet:**

  M := { x | x hat die Eigenschaft $\mathcal{E}$ }

  Beispiele:

  LAUF := { x | x ist deutsches Lexem, das mit „LAUF“ beginnt }

  EoR := { x | x ist deutsches Lexem, das auf „E“ oder „R“ endet }

Mengentheoretische Grundbegriffe

- **Seien M_1 und M_2 Mengen. M_1 ist Teilmenge von M_2, falls aus $x \in M_1$ stets $x \in M_2$ folgt; symbolisch: $M_1 \subseteq M_2$.**
- **Gilt für zwei Mengen, M_1 und M_2 , einerseits $M_1 \subseteq M_2$ und andererseits $M_1 \neq M_2$, dann ist M_1 echte Teilmenge von M_2 ; symbolisch: $M_1 \subset M_2$**

Beispiele:

LAUF* := {Laufbahn, laufen, Lauffeuer, Laufmasche, Laufsteg} $\subseteq$ LAUF

LAUF $\subset$ LA := {x | x ist deutsches Lexem, das mit „LA“ beginnt }

R := {x | x ist deutsches Lexem, das auf „R“ endet } $\subseteq$ EoR

Mengentheoretische Grundbegriffe

- **Gilt für zwei Mengen, M_1 und M_2, sowohl $M_1 \subseteq M_2$ als auch $M_2 \subseteq M_1$, so folgt: $M_1 = M_2$ (Mengengleichheit).**
- **Die leere Menge ist die Menge, die kein Element enthält; symbolisch: {} oder Ø.**
  - **Bemerkung: Ø ist Teilmenge jeder Menge.**
- **Die Kardinalität einer endlichen Menge M ist die Anzahl ihrer Elemente; symbolisch: $|M|$**

Mengentheoretische Grundbegriffe

- **Wenn M und N Mengen sind, dann charakterisiert die Menge**

M $\cap$ N **:= {x | $x \in$ M und $x \in$ N }**

den Durchschnitt

M $\cup$ N **:= {x | $x \in$ M oder $x \in$ N }**

die Vereinigung

von M und N

Mengentheoretische Grundbegriffe

- **Beispiele:**

  LAUF*:= {Laufbahn, laufen, Lauffeuer, Laufmasche, Laufsteg}

LAUF* ∩ EoR

= { Lauffeuer, Laufmasche }

{ Lauffeuer, Laufmasche } ∪ { Lauffeuer, Laufpass }

= { Lauffeuer, Laufmasche, Laufpass }

Mengentheoretische Grundbegriffe

- **Wenn I = {1,...,n} eine nichtleere Index-menge ist und jedes $i \in$ I für M_i eine Menge charakterisiert, dann gilt als**
  - **Verallgemeinerung des Durchschnitts**

$$\bigcap_{i \in I} M_i := \{x \mid x \in M_i \text{ für alle } i \in I\} = \bigcap_{i=1}^{n} M_i$$

  - **Verallgemeinerung der Vereinigung**

$$\bigcup_{i \in I} M_i := \{x \mid x \in M_i \text{ f.mind.ein } i \in I\} = \bigcup_{i=1}^{n} M_i$$

Mengentheoretische Grundbegriffe

- **Die Menge aller Teilmengen einer Menge M heißt Potenzmenge:**

$$\wp(M) := \{ N \mid N \subseteq M \} = 2^M$$

Beispiel:

LAUFS := { Laufschritt, Laufstall, Laufsteg }

2^{LAUFS} = { Ø, {Laufschritt}, {Laufstall}, {Laufsteg},
{Laufschritt, Laufstall}, {Laufschritt, Laufsteg},
{Laufstall, Laufsteg}, LAUFS }

$| 2^{LAUFS} | = 2^3 = 8$

Mengentheoretische Grundbegriffe

- Das **Kartesische Produkt** von endlich vielen Mengen $M_1,..,M_n$, $n \geq 2$, ist die Menge aller **n-tupel**:

$$M_1 \times M_2 \times \ldots \times M_n := \{ (m_1,..,m_n) \mid m_i \in M_i ,\ 1 \leq i \leq n \}$$

Beispiel:

LAUFB := { Laufbahn, Laufbursche }

LAUFS := { Laufschritt, Laufstall, Laufsteg }

LAUFB x LAUFS = { (Laufbahn, Laufschritt), (Laufbahn, Laufstall), (Laufbahn, Laufsteg), (Laufbursche, Laufschritt), (Laufbursche, Laufstall), (Laufbursche, Laufsteg) }

Grundbegriffe zu Relationen

- Eine (zweistellige) Relation ρ zwischen zwei Mengen M_1 und M_2 ist eine Teil-menge von $M_1 \times M_2$, d.h. $\rho \subseteq M_1 \times M_2$. Man schreibt auch $m \, \rho \, n$ für $(m,n) \in \rho$.

Beispiel:

GleicheLänge $\subseteq$ DLexeme x DLexeme

GleicheLänge = { (du, da), (da, Ei), (er, es), (Dom, Bor),
(Aal, Tor), (Bild, Tier), (Tiger, Sekte),... }

Grundbegriffe zu Relationen

- **Eine Relation ρ auf einer nichtleeren Menge M heißt Äquivalenzrelation, wenn**
  - $m \,\rho\, m$ **für jedes** $m \in$ **M** (**reflexiv**)
  - **aus** $m \,\rho\, n$ **folgt** $n \,\rho\, m$ (**symmetrisch**)
  - **aus** $k \,\rho\, m$ **und** $m \,\rho\, n$ **folgt** $k \,\rho\, n$ (**transitiv**)

Beispiel:

GleicheLänge ist Äquivalenzrelation:

(1) reflexiv: (du, du), (da, da), (Ei, Ei), (Aal, Aal), (Tiger, Tiger), ...

(2) symmetrisch: (Aal, Tor) $\Rightarrow$ (Tor, Aal), (Tiger, Sekte) $\Rightarrow$ (Sekte, Tiger), ...

(3) transitiv: (du, da), (da, Ei) $\Rightarrow$ (du, Ei),
(Bild, Tier), (Tier, Rand) $\Rightarrow$ (Bild, Rand), ...

Grundbegriffe zu Relationen

- **Ist ρ eine Äquivalenzrelation auf einer Menge M, dann heißt jede Menge $[m]$:= { n | $m \rho$ n } für ein $m \in$ M die von m repräsentierte Äquivalenzklasse.**
- **Jede Äquivalenzrelation auf M bewirkt eine Einteilung von M in paarweise disjunkte (d.h. elementfreie) Äquivalenzklassen.**

Beispiele:

[da] = { Ei, er, es, du, ... }

[Aal] = { Tor, Bor, elf, vom, ... }

[Bild]= { Tier, Rand, grün, hell, ... }

Grundbegriffe zu Relationen

- **Eine Halbordnung (partielle Ordnung) auf einer Menge M ist eine Relation „<" auf M mit den Eigenschaften**
  - **aus $k < m$ und $m < n$ folgt: $k < n$ (transitiv)**
  - **für kein $m \in$ M gilt: $m < m$ (irreflexiv)**

Beispiel:

Ist_Unterbegriff $\subseteq$ DLexeme x DLexeme

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

Grundbegriffe zu Relationen

- **Eine lineare Ordnung (totale Ordnung) auf einer Menge M ist eine Halbordnung „<" auf M, bei der für beliebige $m, n \in$ M entweder m < n oder n < m oder m = n gilt.**

Beispiel:

Lexikographisch_Vor $\subseteq$ DLexeme x DLexeme
Lexikographisch_Vor
= { (Aal, Bild), (Bild, Rand), (Rand, Tiger), (Stelle, Stiel), (Stiel, Stil), (Stil, Stunde), (Klee, Zone), … }

Grundbegriffe zu Relationen

- **Das Produkt zweier Relationen, ρ und σ auf M, ist festgelegt durch**

  $\rho\,\sigma := \{\, (x,z) \mid (x,y) \in \rho$ **und** $(y,z) \in \sigma$ **f.e.** $y \in M \,\}$

Grundbegriffe zu Relationen

- Für eine beliebige Relation ρ auf M definiert
  - $\rho^0 := \{ (m,m) \mid m \in M \}$ die Diagonale,
  - $\rho^1 := \rho$ und $\rho^i := \rho^{i-1} \rho$ für $i>1$
  - $\rho^+ := \bigcup_{i \geq 1} \rho^i = \rho^1 \cup \rho^2 \cup \ldots \cup \rho^n$
    die transitive Hülle von ρ,
  - $\rho^* := \bigcup_{i \geq 0} \rho^i = \rho^0 \cup \rho^1 \cup \rho^2 \cup \ldots \cup \rho^n$
    die reflexive und transitive Hülle von ρ

Transitive Hülle von *„Ist_Unterbegriff“*

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

Transitive Hülle von *„Ist_Unterbegriff“*

Ist_Unterbegriff
= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

$\text{Ist\_Unterbegriff}^1 = \text{Ist\_Unterbegriff}$
= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

Transitive Hülle von *„Ist_Unterbegriff"*

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (<u>VW-PKW</u>, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

Ist_Unterbegriff^1 = Ist_Unterbegriff

= { (VW-Golf, <u>VW-PKW</u>), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

Ist_Unterbegriff^2 = Ist_Unterbegriff^1 Ist_Unterbegriff

= { (VW-Golf, PKW) }

Transitive Hülle von *„Ist_Unterbegriff"*

Ist_Unterbegriff
= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

$Ist\_Unterbegriff^1$ = Ist_Unterbegriff
= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

$Ist\_Unterbegriff^2 = Ist\_Unterbegriff^1$ Ist_Unterbegriff
= { (VW-Golf, PKW), (VW-PKW, KFZ) }

Transitive Hülle von *„Ist_Unterbegriff“*

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (<u>KFZ</u>, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

$\text{Ist\_Unterbegriff}^1 = \text{Ist\_Unterbegriff}$

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, <u>KFZ</u>), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

$\text{Ist\_Unterbegriff}^2 = \text{Ist\_Unterbegriff}^1\ \text{Ist\_Unterbegriff}$

= { (VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt) }

Transitive Hülle von „*Ist_Unterbegriff*“

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

Ist_Unterbegriff^1 = Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

Ist_Unterbegriff^2 = Ist_Unterbegriff^1 Ist_Unterbegriff

= { (VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt), (KFZ, Objekt) }

Transitive Hülle von *„Ist_Unterbegriff“*

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

$\text{Ist\_Unterbegriff}^1 = \text{Ist\_Unterbegriff}$

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

$\text{Ist\_Unterbegriff}^2 = \text{Ist\_Unterbegriff}^1 \ \text{Ist\_Unterbegriff}$

= { (VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt), (KFZ, Objekt), (Schreibtisch, Artefakt) }

Transitive Hülle von *„Ist_Unterbegriff“*

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

$\text{Ist\_Unterbegriff}^1 = \text{Ist\_Unterbegriff}$

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

$\text{Ist\_Unterbegriff}^2 = \text{Ist\_Unterbegriff}^1 \ \text{Ist\_Unterbegriff}$

= { (VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt), (KFZ, Objekt), (Schreibtisch, Artefakt), (Möbel, Objekt) }

Transitive Hülle von *„Ist_Unterbegriff“*

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

$\text{Ist\_Unterbegriff}^2 = \text{Ist\_Unterbegriff}^1\ \text{Ist\_Unterbegriff}$

= { (VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt), (KFZ, Objekt), (Schreibtisch, Artefakt), (Möbel, Objekt) }

$\text{Ist\_Unterbegriff}^3 = \text{Ist\_Unterbegriff}^2\ \text{Ist\_Unterbegriff}$

= { (VW-Golf, KFZ) }

Transitive Hülle von *„Ist_Unterbegriff"*

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

Ist_Unterbegriff^2 = Ist_Unterbegriff^1 Ist_Unterbegriff

= { (VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt), (KFZ, Objekt), (Schreibtisch, Artefakt), (Möbel, Objekt) }

Ist_Unterbegriff^3 = Ist_Unterbegriff^2 Ist_Unterbegriff

= { (VW-Golf, KFZ), (VW-PKW, Artefakt) }

Transitive Hülle von *„Ist_Unterbegriff“*

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

$\text{Ist\_Unterbegriff}^2 = \text{Ist\_Unterbegriff}^1\ \text{Ist\_Unterbegriff}$

= { (VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt), (KFZ, Objekt), (Schreibtisch, Artefakt), (Möbel, Objekt) }

$\text{Ist\_Unterbegriff}^3 = \text{Ist\_Unterbegriff}^2\ \text{Ist\_Unterbegriff}$

= { (VW-Golf, KFZ), (VW-PKW, Artefakt), (PKW, Objekt) }

Transitive Hülle von „*Ist_Unterbegriff*“

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

Ist_Unterbegriff^2 = Ist_Unterbegriff^1 Ist_Unterbegriff

= { (VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt), (KFZ, Objekt), (Schreibtisch, Artefakt), (Möbel, Objekt) }

Ist_Unterbegriff^3 = Ist_Unterbegriff^2 Ist_Unterbegriff

= { (VW-Golf, KFZ), (VW-PKW, Artefakt), (PKW, Objekt), (Schreibtisch, Objekt) }

Transitive Hülle von „*Ist_Unterbegriff*"

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

Ist_Unterbegriff^2 = Ist_Unterbegriff^1 Ist_Unterbegriff

= { (VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt), (KFZ, Objekt), (Schreibtisch, Artefakt), (Möbel, Objekt) }

Ist_Unterbegriff^3 = Ist_Unterbegriff^2 Ist_Unterbegriff

= { (VW-Golf, KFZ), (VW-PKW, Artefakt), (PKW, Objekt), (Schreibtisch, Objekt) }

Ist_Unterbegriff^4 = Ist_Unterbegriff^3 Ist_Unterbegriff

= { (VW-Golf, Artefakt) }

Transitive Hülle von *„Ist_Unterbegriff"*

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

Ist_Unterbegriff^2 = Ist_Unterbegriff^1 Ist_Unterbegriff

= { (VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt), (KFZ, Objekt), (Schreibtisch, Artefakt), (Möbel, Objekt) }

Ist_Unterbegriff^3 = Ist_Unterbegriff^2 Ist_Unterbegriff

= { (VW-Golf, KFZ), (VW-PKW, Artefakt), (PKW, Objekt), (Schreibtisch, Objekt) }

Ist_Unterbegriff^4 = Ist_Unterbegriff^3 Ist_Unterbegriff

= { (VW-Golf, Artefakt), (VW-PKW, Objekt) }

Transitive Hülle von *„Ist_Unterbegriff"*

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

Ist_Unterbegriff^2 = Ist_Unterbegriff^1 Ist_Unterbegriff

= { (VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt), (KFZ, Objekt), (Schreibtisch, Artefakt), (Möbel, Objekt) }

Ist_Unterbegriff^3 = Ist_Unterbegriff^2 Ist_Unterbegriff

= { (VW-Golf, KFZ), (VW-PKW, Artefakt), (PKW, Objekt), (Schreibtisch, Objekt) }

Ist_Unterbegriff^4 = Ist_Unterbegriff^3 Ist_Unterbegriff

= { (VW-Golf, Artefakt), (VW-PKW, Objekt) }

Ist_Unterbegriff^5 = Ist_Unterbegriff^4 Ist_Unterbegriff

= { (VW-Golf, Objekt) }

Transitive Hülle von *„Ist_Unterbegriff"*

Ist_Unterbegriff

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt) }

$\text{Ist\_Unterbegriff}^2 = \text{Ist\_Unterbegriff}^1\ \text{Ist\_Unterbegriff}$

= { (VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt), (KFZ, Objekt), (Schreibtisch, Artefakt), (Möbel, Objekt) }

$\text{Ist\_Unterbegriff}^3 = \text{Ist\_Unterbegriff}^2\ \text{Ist\_Unterbegriff}$

= { (VW-Golf, KFZ), (VW-PKW, Artefakt), (PKW, Objekt), (Schreibtisch, Objekt) }

$\text{Ist\_Unterbegriff}^4 = \text{Ist\_Unterbegriff}^3\ \text{Ist\_Unterbegriff}$

= { (VW-Golf, Artefakt), (VW-PKW, Objekt) }

$\text{Ist\_Unterbegriff}^5 = \text{Ist\_Unterbegriff}^4\ \text{Ist\_Unterbegriff}$

= { (VW-Golf, Objekt) }

$\text{Ist\_Unterbegriff}^6 = \text{Ist\_Unterbegriff}^5\ \text{Ist\_Unterbegriff} = \{\}$

Transitive Hülle von *„Ist_Unterbegriff“*

$$\bigcup_{i \geq 1} \text{Ist\_Unterbegriff}^i = \text{Ist\_Unterbegriff}^1 \cup \text{Ist\_Unterbegriff}^2 \cup \ldots \cup \text{Ist\_Unterbegriff}^n$$

= { (VW-Golf, VW-PKW), (VW-PKW, PKW), (PKW, KFZ), (KFZ, Artefakt), (Artefakt, Objekt), (Schreibtisch, Möbel), (Möbel, Artefakt),

(VW-Golf, PKW), (VW-PKW, KFZ), (PKW, Artefakt), (KFZ, Objekt), (Schreibtisch, Artefakt), (Möbel, Objekt),

(VW-Golf, KFZ), (VW-PKW, Artefakt), (PKW, Objekt), (Schreibtisch, Objekt),

(VW-Golf, Artefakt), (VW-PKW, Objekt),

(VW-Golf, Objekt) }

Grundlagen formaler Sprachen:
Alphabet

- Sei Σ ein beliebiges Alphabet, d.i. eine Menge von Symbolen oder Zeichen
  - Beispiele für verbreitete Alphabete:
    - {A,B,C, ..., X,Y,Z} — lateinisches Alphabet
    - {1,2,3, ..., 7,8,9, 0} — indisch-arabisches Zahlensystem
    - {0,1} — Binärzahlen
    - {  } — internat. Ampelalphabet
    - {A[denin], G[uanin], T[hymin], C[ytosin]} — Basen-Alphabet der DNA

Grundlagen formaler Sprachen:
Wörter

- Seien **Wörter** (**Sätze**, **Strings**, **Ketten**) über einem Alphabet Σ in der folgenden Weise definiert:

1. ε ist ein Wort über Σ (ε ist das Leerwort, das keine Symbole hat)
2. falls χ ein Wort über Σ und $\alpha \in \Sigma$ ist, dann ist $\chi\,\alpha$ ein Wort über Σ
3. γ ist ein Wort über Σ genau dann, wenn sein Bildung aus (1) oder (2) folgt

Grundlagen formaler Sprachen:
Konkatenation von Wörtern

- Das Wort $\omega \circ \tau := \omega\tau := \omega_1 \dots \omega_m \tau_1 \dots \tau_n$ heißt **Konkatenation** von ω und τ, falls $\omega = \omega_1 \dots \omega_m$ und $\tau = \tau_1 \dots \tau_n$ ($\omega_i, \tau_j \in \Sigma$) Wörter über Σ sind;
„o“ (sprich: „Kringel“) ist der Konkatenationsoperator.
  - Für alle Wörter ω gilt: $\omega \circ \varepsilon = \varepsilon \circ \omega = \omega$

Beispiele für Konkatenationen:

- ABC o X = ABCX
- 24 o 24 = 2424
- wort o stamm = wortstamm

Formale Grundlagen von Automaten: formale Sprache

- **Eine (formale) Sprache $\mathcal{L}$ über einem Alphabet Σ ist eine Menge von Ketten über Σ .**
- **Sei ferner Σ^* (bzw. Σ^+) die Menge *aller* Ketten über Σ unter Einschluss (bzw. Ausschluss) von ε.**
- **Dann gilt für jede Sprache $\mathcal{L}$ über Σ :**

$$\mathcal{L} \subseteq \Sigma^*$$

Beispiele für formale Sprachen

- $\Sigma = \{A,B,C, \ldots, X,Y,Z,\_\}$
  - $\mathcal{L}_1 = \{GUTEN\_TAG, GUTEN, TAG\}$
    $= \{GUTEN, GUTEN\_TAG, TAG\}$
  - $\mathcal{L}_2 = \{GTNTG, GTN, TG\}$
  - $\mathcal{L}_3 = \{TNT, TN, T\}$
  - $\mathcal{L}_4 = \{GG, GTTG, GGGG, GGTTGG, \ldots\}$
  - $\mathcal{L}_5 = \{A, C, G, T, ACCGTG, \ldots\}$

Beispiele für formale Sprachen

- $\Sigma = \{0,1,2,3, \ldots, 7,8,9,+,-,=\}$
  - $\mathcal{L}_1 = \{5+7=20-8,\ 5+7=5+8,\ 5759=57-59\}$
  - $\mathcal{L}_2 = \{5+=7-2=,\ +++,\ 1-11782---3\}$
  - $\mathcal{L}_3 = \{3,\ 33,\ 333,\ 3333,\ 33333,\ \ldots\}$

Beispiele für formale Sprachen

Σ = {NOMEN, ADJEKTIV, $+\text{ize}_V$, $+\text{ation}_N$, $+\text{al}_{Adj}$}

– $\mathcal{L}_1$ = { hospital, hospital$+\text{ize}_V$,
  hospital $+\text{ize}_V+\text{ation}_N$, ... }

– $\mathcal{L}_2$ = { hospital, moral,
  hospital$+\text{ize}_V$, moral$+\text{ize}_V$, ... }

– $\mathcal{L}_3$ = { $+\text{ize}_V$, $+\text{ize}_V+\text{ize}_V$, $+\text{ize}_V+\text{ize}_V$hospital, ... }

NOMEN: hospital, motor, category, ...

ADJEKTIV: moral, concrete, tender, ...

Grundlagen formaler Sprachen:
Wortmengen

- Eine **Wortmenge** (**Sprache**) $\mathcal{F}$ bezüglich eines Alphabets Σ ist gegeben durch

$$\mathcal{F} := \{\ \omega \mid \omega \text{ ist Wort über } \Sigma\ \}$$

Grundlagen formaler Sprachen:
Konkatenation von Wortmengen

- Die **Zusammensetzung** (**Konkatenation**) von **Wortmengen** $\mathcal{F}$ und $\mathcal{M}$ ist gegeben durch

$$\mathcal{F} \circ \mathcal{M} := \mathcal{F}\,\mathcal{M} := \{\, \omega\tau \mid \omega \in \mathcal{F}\,,\, \tau \in \mathcal{M} \,\}$$

- Dabei gilt:

$$\mathcal{F}\,\{\varepsilon\} = \{\varepsilon\}\,\mathcal{F} = \mathcal{F}$$

$$\mathcal{F}\,\emptyset = \emptyset\,\mathcal{F} = \emptyset$$

Beispiele für Konkatenationen:

- DLex := {Fisch, Fest, Fleck}; DEnd := {e, es, er}
- DLex ∘ DEnd = {Fische, Fisches, Fischer, Feste, Festes, Fester, Flecke, Fleckes, Flecker}

Grundlagen formaler Sprachen:
Potenzen von Wörtern bzw. Wortmengen

- **Potenzen** von **Wörtern** ω bzw. **Wortmengen** $\mathcal{F}$ sind gegeben durch:
  - $\omega^0 := \varepsilon$ $\quad \omega^1 := \omega$ $\quad \omega^i := \omega^{i-1}\,\omega$, für $i \geq 1$
  - $\mathcal{F}^0 := \{\varepsilon\}$ $\quad \mathcal{F}^1 := \mathcal{F}$ $\quad \mathcal{F}^i := \mathcal{F}^{i-1}\,\mathcal{F}$, für $i \geq 1$

Beispiele für Potenzen von Wortmengen:

- DSilbe := {ba, di, ko}
- $DSilbe^0 = \{\varepsilon\}$, $DSilbe^1 = \{ba, di, ko\}$
- $DSilbe^2 = DSilbe^1\ DSilbe$
  = { baba, badi, bako, diba, didi, diko , koba, kodi, koko}
- $DSilbe^3 = DSilbe^2\ DSilbe$

Grundlagen formaler Sprachen:
Plus- und Sternhülle – formale Sprache

- Die **Plushülle** bzw. **Sternhülle** einer Wortmenge $\mathcal{F}$ werden definiert durch:

$$\mathcal{F}^{+} := \bigcup_{i \geq 1} \mathcal{F}^{i} \qquad \mathcal{F}^{*} := \bigcup_{i \geq 0} \mathcal{F}^{i}$$

- Ist Σ ein Alphabet, dann ist Σ^* die Gesamtheit aller Wörter über Σ. Jede Teilmenge dieser Sternhülle, $\mathcal{L} \subseteq \Sigma^*$, heißt **formale Sprache** über Σ .

Endlicher Automat (1/2)

- Ein endlicher Automat (*finite-state auto-maton, FSA*) ist ein formales System zur Erkennung von ([einer beschränkten Menge! von]) formalen Sprachen.
- Ein FSA beginnt den Erkennungsprozess in seinem ausgezeichneten Startzustand. Falls der FSA die Eingabe komplett gelesen hat und er sich in einem der ausgezeichne-ten Endzustände befindet, hat er die Ein-gabe akzeptiert, sonst nicht.

Endlicher Automat (2/2)

- Ein FSA besteht i.A. aus einem Eingabeband, auf dem der zu akzeptierende Ausdruck steht, und einem endlichen Steuerungs-mechanismus, der die Form der erlaubten Zustandsänderungen (Bewegungen) determiniert.
- Die Zustandsänderungsfunktion gibt an, welche möglichen Folgezustände aus dem aktuellen Zustand und dem aktuellen Symbol auf dem Eingabeband erreichbar sind.

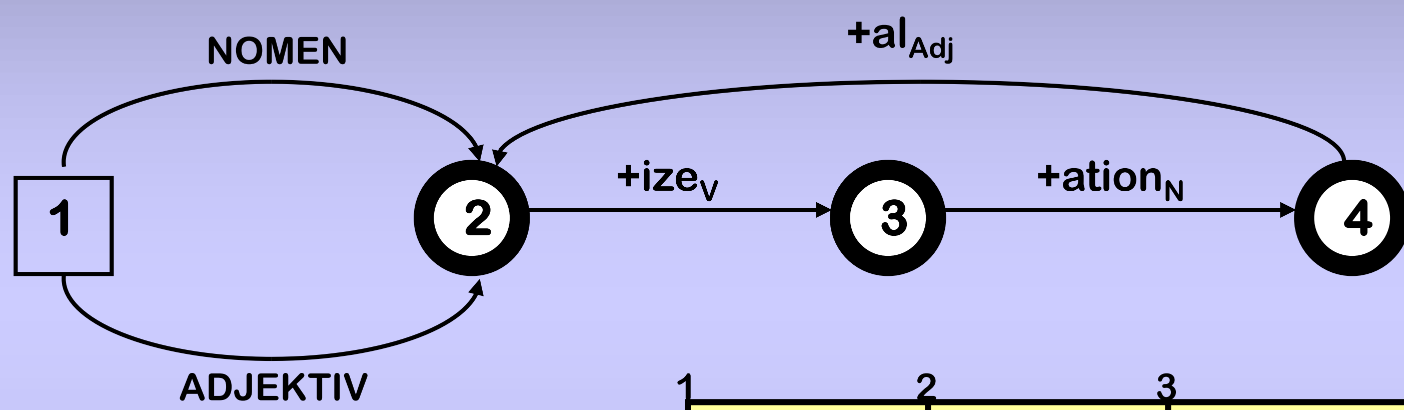
Formale Grundlagen von Automaten: endlicher Automat

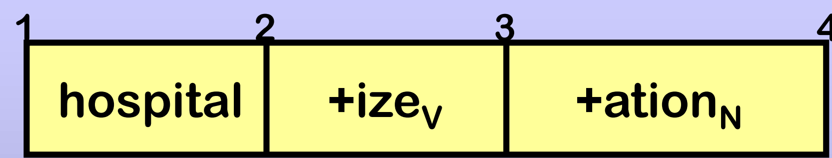
- Ein (nicht-deterministischer) endlicher Automat ist ein 5-Tupel (Q, Σ , δ, q_o, F) mit
  - Q: endliche Menge von (Steuerungs-)Zuständen $q_o, q_1, q_2, \ldots, q_n$
  - Σ: endliches Eingabealphabet
  - δ: Zustandsübergangsfunktion, die das Steuerungsverhalten des FSA determiniert

$$\delta : Q \times \Sigma \mapsto \wp(Q)$$

  - $q_o \in Q$: der ausgezeichnete Startzustand
  - $F \subseteq Q$: Menge der ausgezeichneten Endzustände

Graph- und Funktionsnotation für Dederivationsautomaten (1/2)





NOMEN: hospital, motor, category, ...

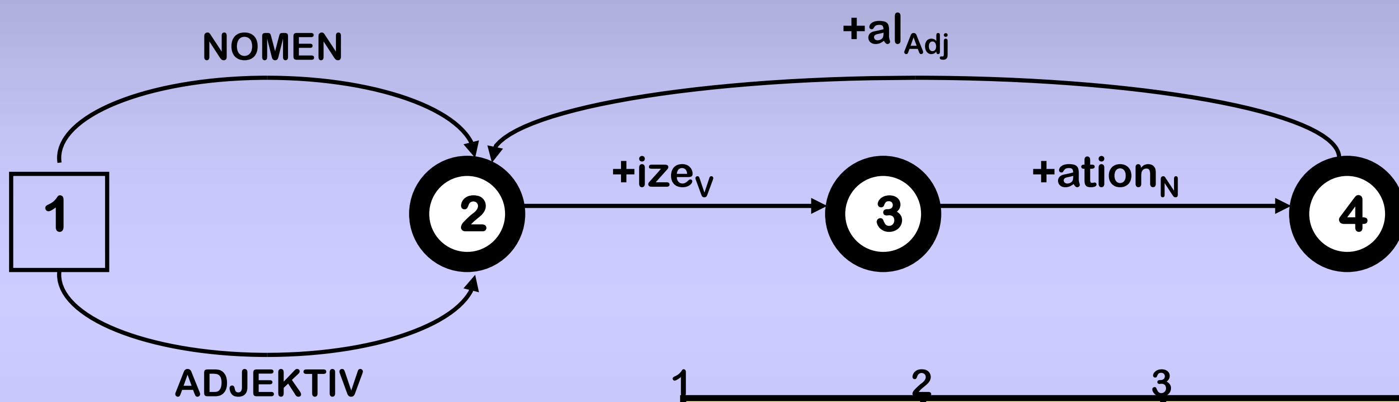
ADJEKTIV: moral, concrete, tender, ...

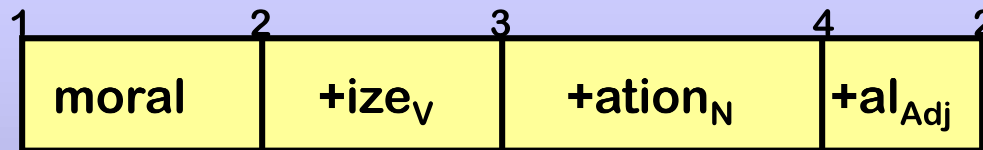
$\delta(1,\text{hospital}) = 2$
$\delta(2,+ize_V) = 3$
$\delta(3,+ation_N) = 4$

n Anfangszustand

n möglicher Endzustand

Graph- und Funktionsnotation für Dederivationsautomaten (2/2)





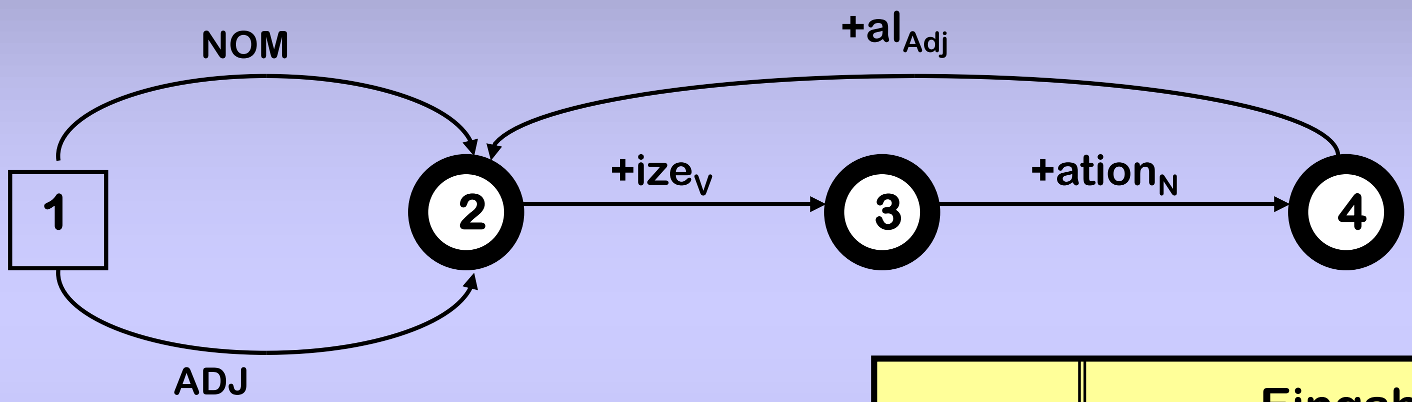
NOMEN: hospital, motor, category, ...

ADJEKTIV: moral, concrete, tender, ...

$\delta(1,\text{hospital}) = 2$	$\delta(1,\text{moral}) = 2$
$\delta(2,+\text{ize}_V) = 3$	$\delta(2,+\text{ize}_V) = 3$
$\delta(3,+\text{ation}_N) = 4$	$\delta(3,+\text{ation}_N) = 4$
	$\delta(4,+\text{al}_{Adj}) = 2$

n Anfangszustand n möglicher Endzustand

Graph- und Tabellennotation für Dederivationsautomaten

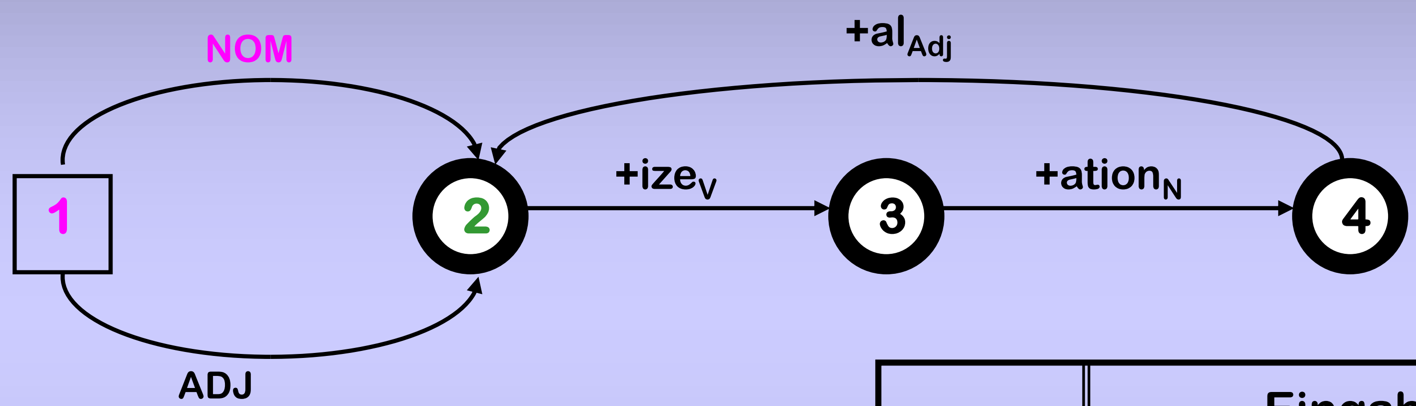


Anfangszustand

möglicher Endzustand

Zustand	Eingabesymbol			
	NOM|ADJ	+ize$_V$	+ation$_N$	+al$_{Adj}$
1	2	0	0	0
<u>2</u>	0	3	0	0
<u>3</u>	0	0	4	0
<u>4</u>	0	0	0	2

Graph- und Tabellennotation für Dederivationsautomaten

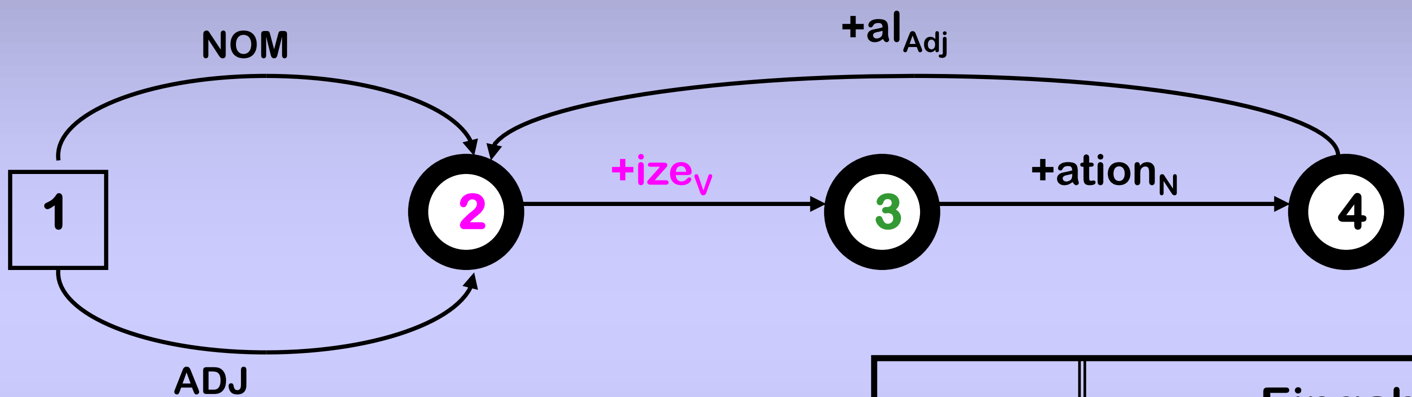


Anfangszustand

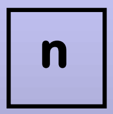
möglicher Endzustand

Zustand	Eingabesymbol			
	NOM|ADJ	$+ize_V$	$+ation_N$	$+al_{Adj}$
1	2	0	0	0
2	0	3	0	0
3	0	0	4	0
4	0	0	0	2

Graph- und Tabellennotation für Dederivationsautomaten



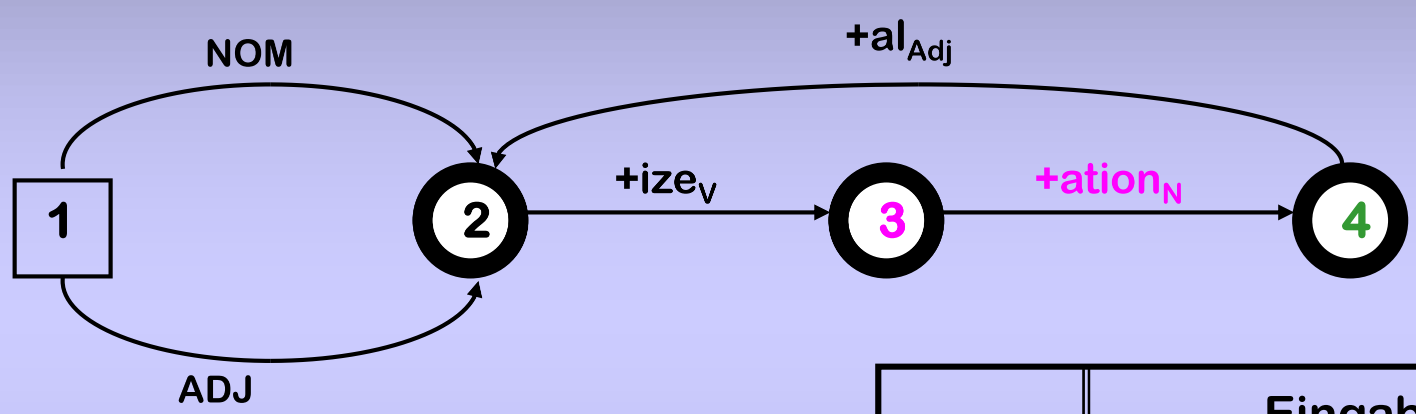
Zustand	Eingabesymbol			
	NOM|ADJ	+ize$_V$	+ation$_N$	+al$_{Adj}$
1	2	0	0	0
<u>2</u>	0	3	0	0
<u>3</u>	0	0	4	0
<u>4</u>	0	0	0	2



Anfangszustand

möglicher Endzustand

Graph- und Tabellennotation für Dederivationsautomaten

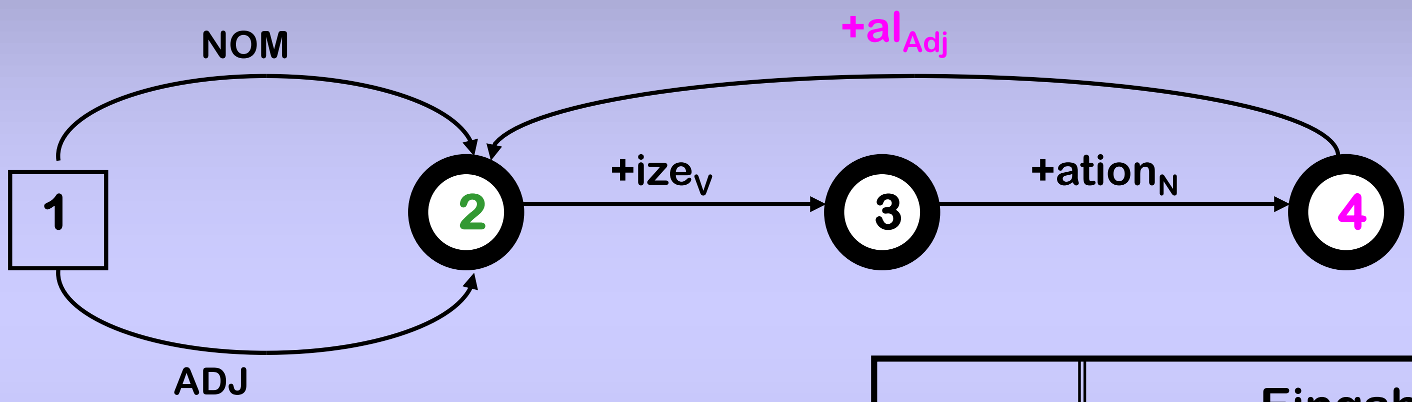


n Anfangszustand

n möglicher Endzustand

Zustand	Eingabesymbol			
	NOM|ADJ	+ize$_{V}$	+ation$_{N}$	+al$_{Adj}$
1	2	0	0	0
<u>2</u>	0	3	0	0
<u>3</u>	0	0	4	0
<u>4</u>	0	0	0	2

Graph- und Tabellennotation für Dederivationsautomaten

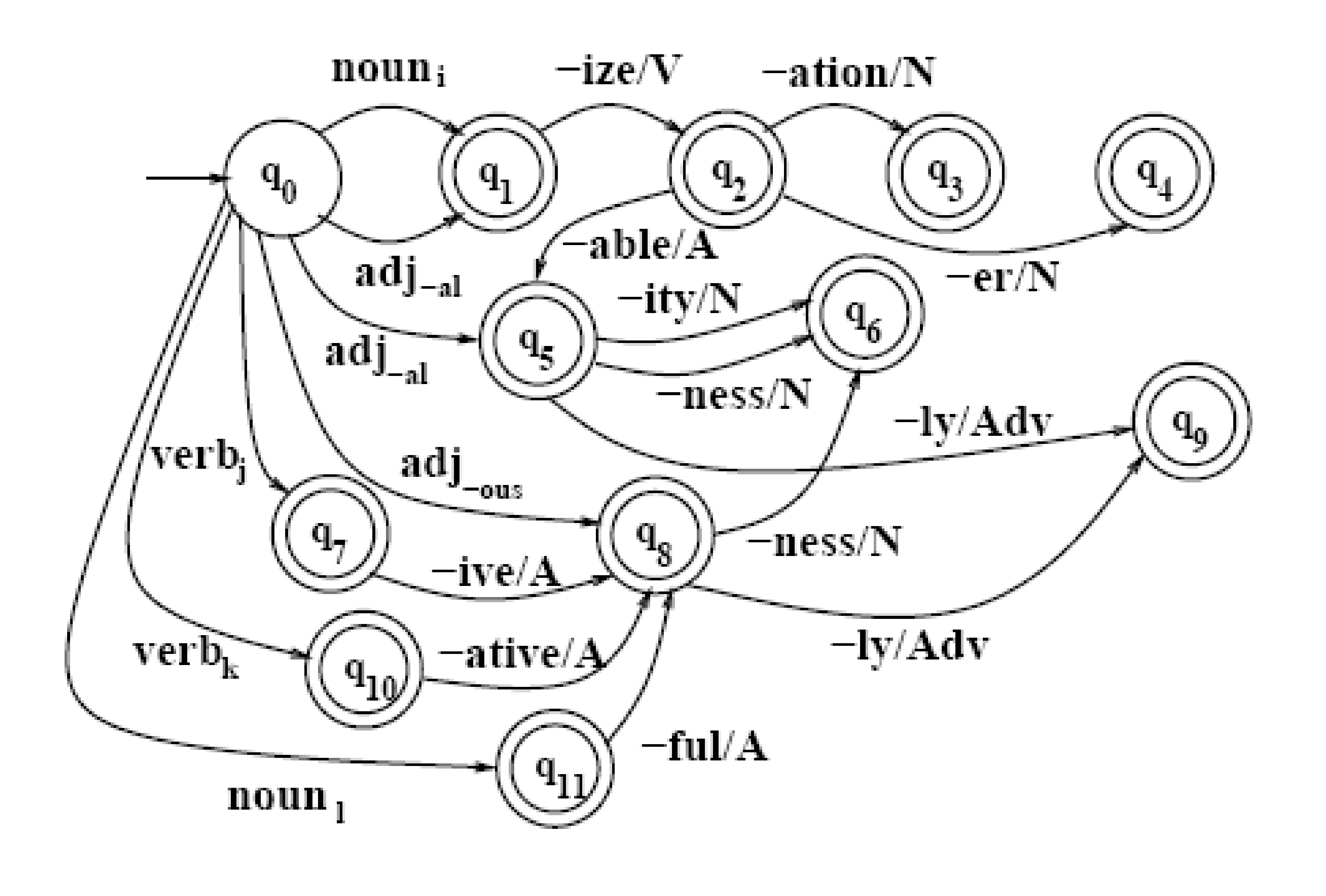


Anfangszustand

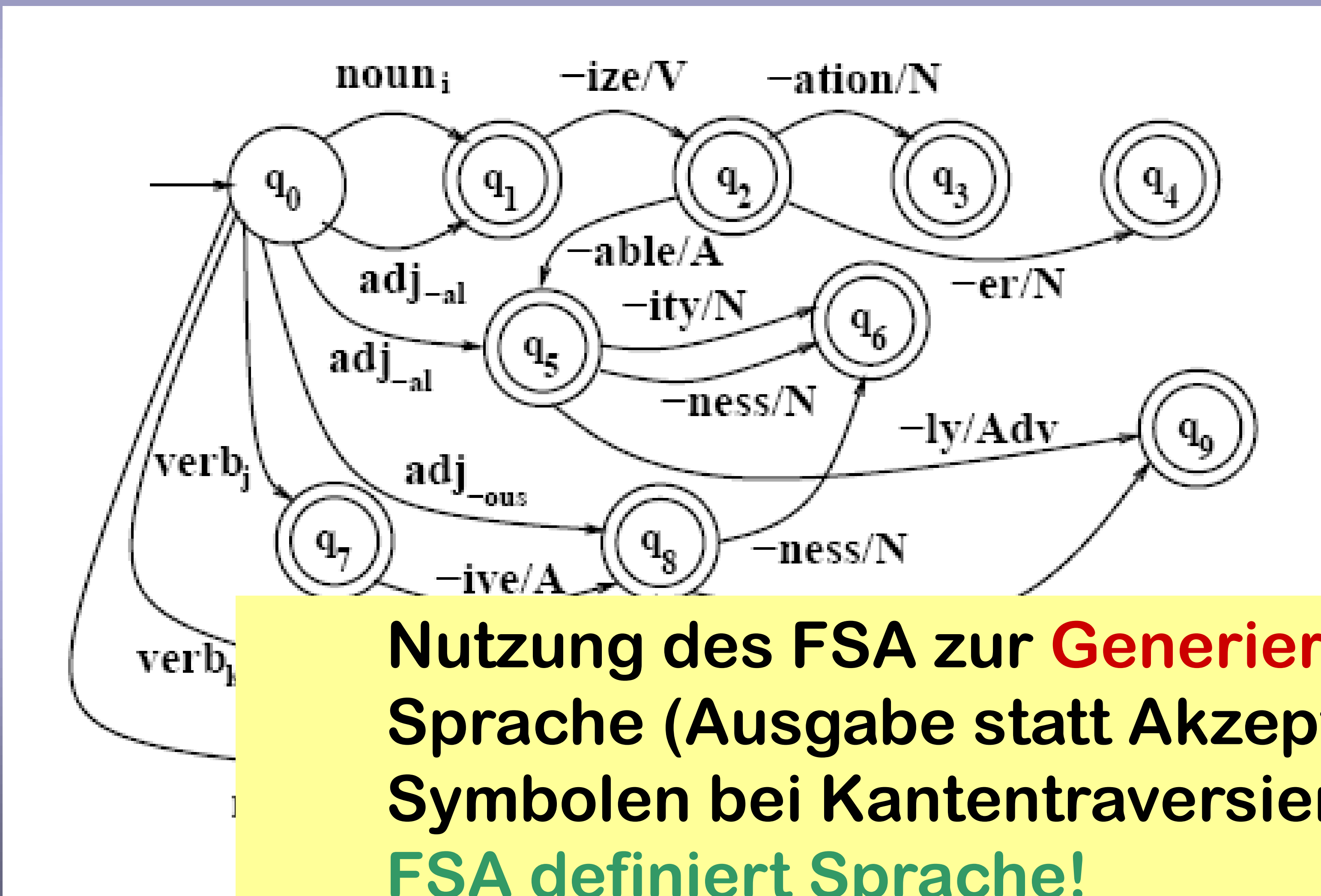
möglicher Endzustand

Zustand	Eingabesymbol			
	NOM|ADJ	$+ize_V$	$+ation_N$	$+al_{Adj}$
1	2	0	0	0
2	0	3	0	0
3	0	0	4	0
4	0	0	0	2

Komplexerer Dederivationsautomat



Komplexerer Dederivationsautomat



Nutzung des FSA zur Generierung von Sprache (Ausgabe statt Akzeptanz von Symbolen bei Kantentraversierung); FSA definiert Sprache!

Formale Grundlagen von Automaten: endlicher Automat

- Ein (nicht-deterministischer) endlicher Automat ist ein 5-Tupel $(Q, \Sigma, \delta, q_o, F)$ mit
  - Q: endliche Menge von (Steuerungs-)Zuständen $q_o, q_1, q_2, \ldots, q_n$
  - Σ: endliches Eingabealphabet
  - δ: Zustandsübergangsfunktion, die das Steuerungsverhalten des FSA determiniert

$$\delta : Q \times \Sigma \mapsto \wp(Q)$$

  - $q_o \in Q$: der ausgezeichnete Startzustand
  - $F \subseteq Q$: Menge der ausgezeichneten Endzustände

Formale Grundlagen von Automaten: Konfiguration

- Sei FSA = $(Q, \Sigma, \delta, q_o, F)$ ein (nicht-deterministischer) endlicher Automat.

  Dann heiße ein Paar $(q,\omega) \in Q \times \Sigma^*$ eine Konfiguration von FSA.

  Eine Konfiguration der Form (q_o,ω) heiße initiale Konfiguration, eine der Form (q,ε), wobei $q \in F$, heiße akzeptierende bzw. Endkonfiguration.

Formale Grundlagen von Automaten: Bewegung

- Sei FSA = $(Q, \Sigma, \delta, q_o, F)$ ein (nicht-deterministischer) endlicher Automat.

  Eine Bewegung in FSA wird durch eine binäre Relation $\vdash_{FSA}$ *(„geht unter FSA nach")* über Konfigurationen repräsentiert:

$$\vdash_{FSA} \subseteq Q \times \Sigma^*$$

Formale Grundlagen von Automaten: Bewegung

- Sei FSA = $(Q, \Sigma, \delta, q_o, F)$ ein (nicht-deterministischer) endlicher Automat.

  Eine Bewegung in FSA wird durch eine binäre Relation $\vdash_{FSA}$ *(„geht unter FSA nach“)* über Konfigurationen repräsentiert.

  Falls $q' \in \delta(q,\tau)$ (q' ist also ein möglicher Folgezustand von q, $\tau \in \Sigma$), dann gilt:

  $$(q,\tau\gamma) \vdash_{FSA} (q',\gamma) \text{ für alle } \gamma \in \Sigma^*$$

Formale Grundlagen von Automaten: Akzeptanz

- **Sei FSA = $(Q, \Sigma, \delta, q_o, F)$ ein (nicht-deterministischer) endlicher Automat und $\vdash^*_{FSA}$ die reflexive und transitive Hülle von $\vdash_{FSA}$.**
  - **reflexive Hülle:**
    - **$(q, \tau) \vdash_{FSA} (q, \tau)$ für alle $(q, \tau) \in Q \times \Sigma^*$**
  - **transitive Hülle:**
    - **$(q', \tau_1) \vdash_{FSA} (q'', \tau_2)$ und $(q'', \tau_2) \vdash_{FSA} (q''', \tau_3)$ $\Rightarrow (q', \tau_1) \vdash_{FSA} (q''', \tau_3)$ für alle $(q, \tau) \in Q \times \Sigma^*$**

Formale Grundlagen von Automaten: Akzeptanz

- Sei FSA = $(Q, \Sigma, \delta, q_o, F)$ ein (nicht-deterministischer) endlicher Automat und $\vdash^*_{FSA}$ die reflexive und transitive Hülle von $\vdash_{FSA}$.
  Eine Eingabekette ω wird durch den FSA akzeptiert, wenn

$$(q_o, \omega) \vdash^*_{FSA} (q,\varepsilon) \text{ für ein } q \in F.$$

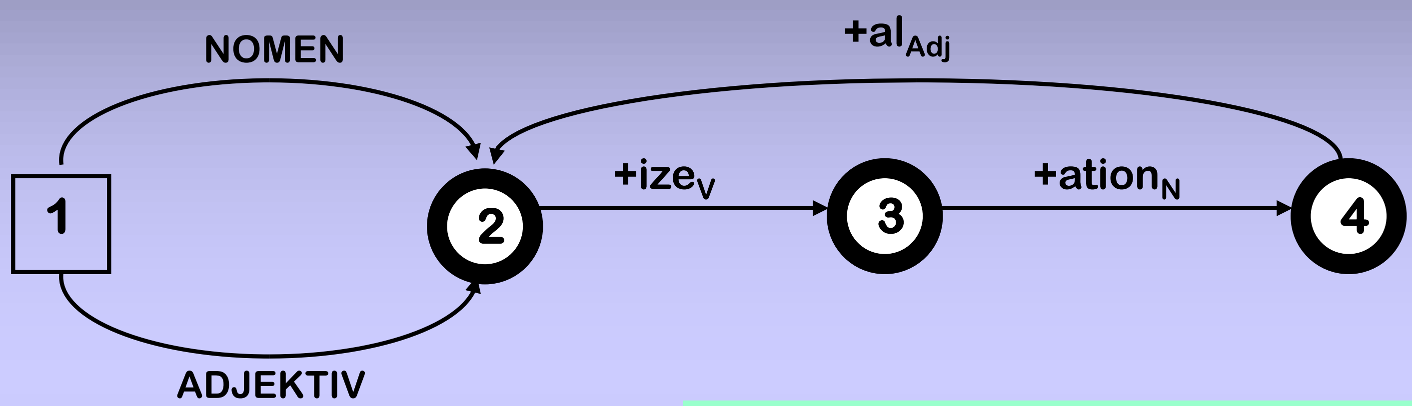
Formale Grundlagen von Automaten: akzeptierte Sprache

- **Sei FSA = $(Q, \Sigma, \delta, q_o, F)$ ein (nicht-deterministischer) endlicher Automat und $\vdash^*_{FSA}$ die reflexive und transitive Hülle von $\vdash_{FSA}$. Die (formale) Sprache $\mathcal{L}_{FSA}$, die durch FSA definiert wird, ist die Menge von Eingabeketten, die vom FSA akzeptiert wird:**

$$\mathcal{L}_{FSA} := \{\, \omega \mid \omega \in \Sigma^*, (q_o,\omega) \vdash^*_{FSA} (q,\varepsilon) \text{ für ein } q \in F \,\}$$

$$\subseteq \Sigma^*$$

Automat für Dederivation



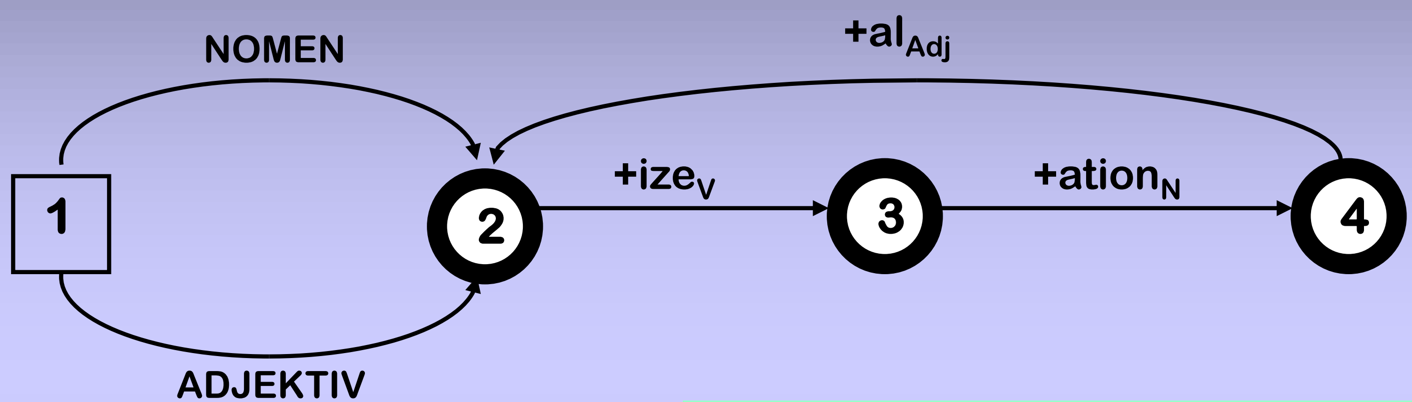
NOMEN: hospital, motor, category, ...

ADJEKTIV: moral, concrete, tender, ...

n **Anfangszustand** n **möglicher Endzustand**

$\boxed{\text{hospital+ize}_V\text{+ation}_N \in \mathcal{L}_{FSA}}$?

Automat für Dederivation



NOMEN: hospital, motor, category, ...

ADJEKTIV: moral, concrete, tender, ...

n Anfangszustand

n möglicher Endzustand

$\boxed{\text{hospital+ize}_V\text{+ation}_N \in \mathcal{L}_{FSA}}$!

$\{ ((1, \text{hospital+ize}_V\text{+ation}_N), (4, \varepsilon)) \} \subseteq \vdash^*_{FSA}$;

$((1, \text{hospital+ize}_V\text{+ation}_N), (4, \varepsilon)) \in \vdash^*_{FSA}$

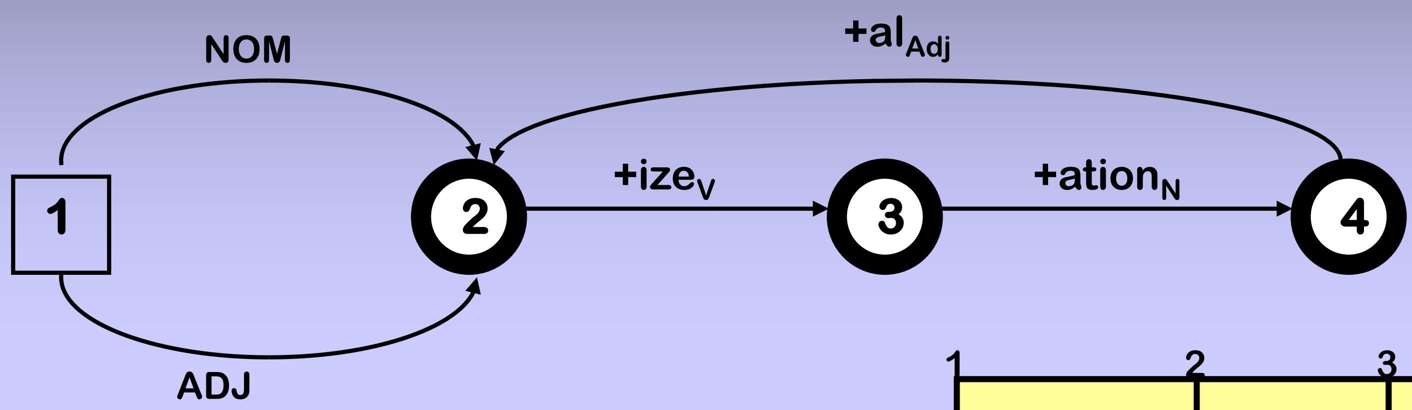
Endlicher Automat

- Das Verhalten eines FSAs hängt von zwei Arten von Informationen ab:
  - dem aktuellen Zustand des endlichen Steuerungssystems,
  - der Symbolkette auf dem Eingabeband, die aus dem aktuellen Symbol unter dem Lesekopf und allen folgenden Symbolen rechts vom aktuellen Symbol besteht

Deterministischer FSA-Erkennungsalgorithmus

- Band (für die zu testende Kette)
  - in n Zellen aufgeteilt
  - jede Zelle hält ein Symbol der Kette
- Zustandstransitionstabelle (2-D Matrix)
  - Zeilen: Zustandsmarken des FSA
  - Spalten: Symbole des Alphabets
  - Zelle: Folgezustand

Automat für Dederivation



1 2 3 4

hospital	+ize$_V$	+ation$_N$	ε

NOM: hospital, motor, category, ...

ADJ: moral, concrete, tender, ...

Zustand	Eingabe			
	NOM|ADJ	+ize$_V$	+ation$_N$	+al$_{Adj}$
1	2	0	0	0
<u>2</u>	0	3	0	0
<u>3</u>	0	0	4	0
<u>4</u>	0	0	0	2

n Anfangszustand

n möglicher Endzustand

Deterministischer FSA-Erkennungsalgorithmus

Funktion D-Erkenner(↓Band,↓FSA) = „accept" oder **„reject"**

```
Index ⇚ Bandanfang
AktualZustand ⇚ Anfangszustand des FSA
LOOP
IF   Ende der Eingabekette ist erreicht   THEN
     IF      AktualZustand ist ein Endzustand   THEN return „accept"
     ELSE  return „reject"
ELSE-IF   Zustandstransitionstabelle[ AktualZustand, Band(Index) ] = 0   THEN
          return „reject"
       ELSE    AktualZustand ⇚ Zustandstransitionstabelle[ AktualZustand, Band(Index) ]
               Index ⇚ Index + 1
LOOPEND
```


Funktion Band(Index) ermittelt das Symbol, das auf der Position **Index** im **Band** steht

Morphologisches Parsing

- Lexikon
  - Liste von Stämmen und Affixen sowie geeigneten morphologischen Merkmalen
- Morphotaktik
  - Modell der Ordnung von Morphemklassen in flektierten Wortformen
    - Beispiel: Pluralsuffix nach Stamm
- Orthographieregeln
  - Regeln zur Beschreibung von Kombinations-effekten bei Morphemen
    - Beispiel aus dem Englischen: y $\mapsto$ ie

Zwei-Ebenen-Morphologie

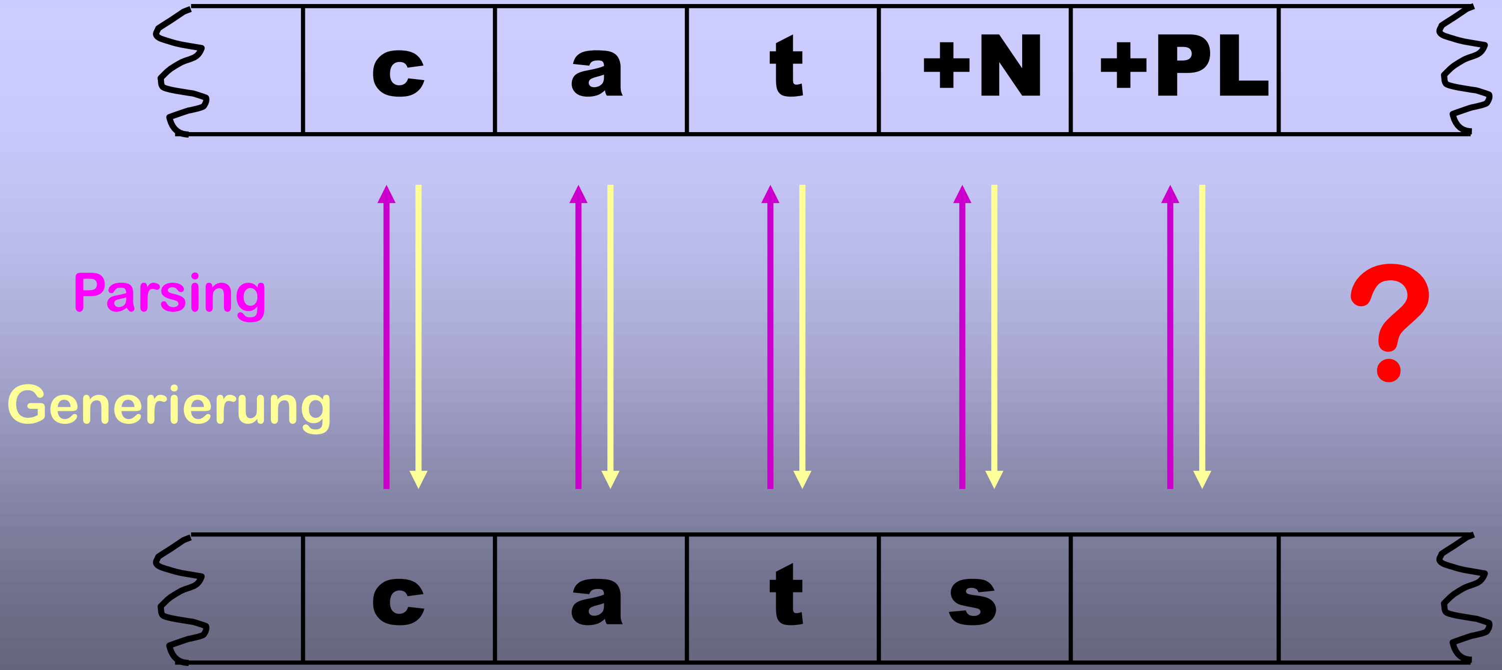
- **Lexikalische Ebene**
  - Wortrepräsentation als Konkatenation von Morphemen

	c	a	t	+N	+PL	

- **Oberflächenebene**
  - Wortrepräsentation als Abfolge von Buchstaben

	c	a	t	s		

Zwei-Ebenen-Morphologie



Erinnerung: endlicher Automat

- Ein (nicht-deterministischer) endlicher Automat ist ein 5-Tupel $(Q, \Sigma , \delta, q_o, F)$ mit
  - Q: endliche Menge von (Steuerungs-)Zuständen $q_o, q_1, q_2, \ldots, q_n$
  - Σ: endliches Eingabealphabet
  - δ: Zustandsübergangsfunktion, die das Steuerungsverhalten des FSA determiniert

$$\delta : Q \times \Sigma \mapsto \wp(Q)$$

  - $q_o \in Q$: der ausgezeichnete Startzustand
  - $F \subseteq Q$: Menge der ausgezeichneten Endzustände

Formale Grundlagen von Automaten: endlicher Transduktor

- Ein (nicht-deterministischer) endlicher Transduktor ist ein 5-Tupel $(Q, \Sigma, \delta, q_o, F)$ mit
  - Q: endliche Menge von (Steuerungs-)Zuständen
  - $\Sigma \subseteq I \times O$: endliches Alphabet mit komplexen Symbolen, die aus Eingabe-Ausgabe-Paaren *i:o* bestehen, $i \in I$ (Eingabealphabet), $o \in O$ (Ausgabealphabet), beide inkl. ε
  - δ: Zustandsübergangsfunktion $\delta(q, i{:}o)$, die das Steuerungsverhalten des FST determiniert

$$\delta : Q \times \Sigma \mapsto \wp(Q)$$

  - $q_o \in Q$: der ausgezeichnete Startzustand
  - $F \subseteq Q$: Menge der ausgezeichneten Endzustände

Formale Grundlagen von Automaten: endlicher Transduktor

- Ein (nicht-deterministischer) endlicher Transduktor ist ein 6-Tupel (Q, Σ, Δ, δ, q_o, F) mit
  - Q: endliche Menge von (Steuerungs-)Zuständen
  - Σ: endliches Eingabealphabet (inkl. ε)
  - Δ: endliches Ausgabealphabet (inkl. ε)
  - δ: Zustandsübergangsfunktion, die das Steuerungsverhalten des FST determiniert

$$\delta : Q \times \Sigma \mapsto \wp(Q \times \Delta^*)$$

  - $q_o \in Q$: der ausgezeichnete Startzustand
  - $F \subseteq Q$: Menge der ausgezeichneten Endzustände

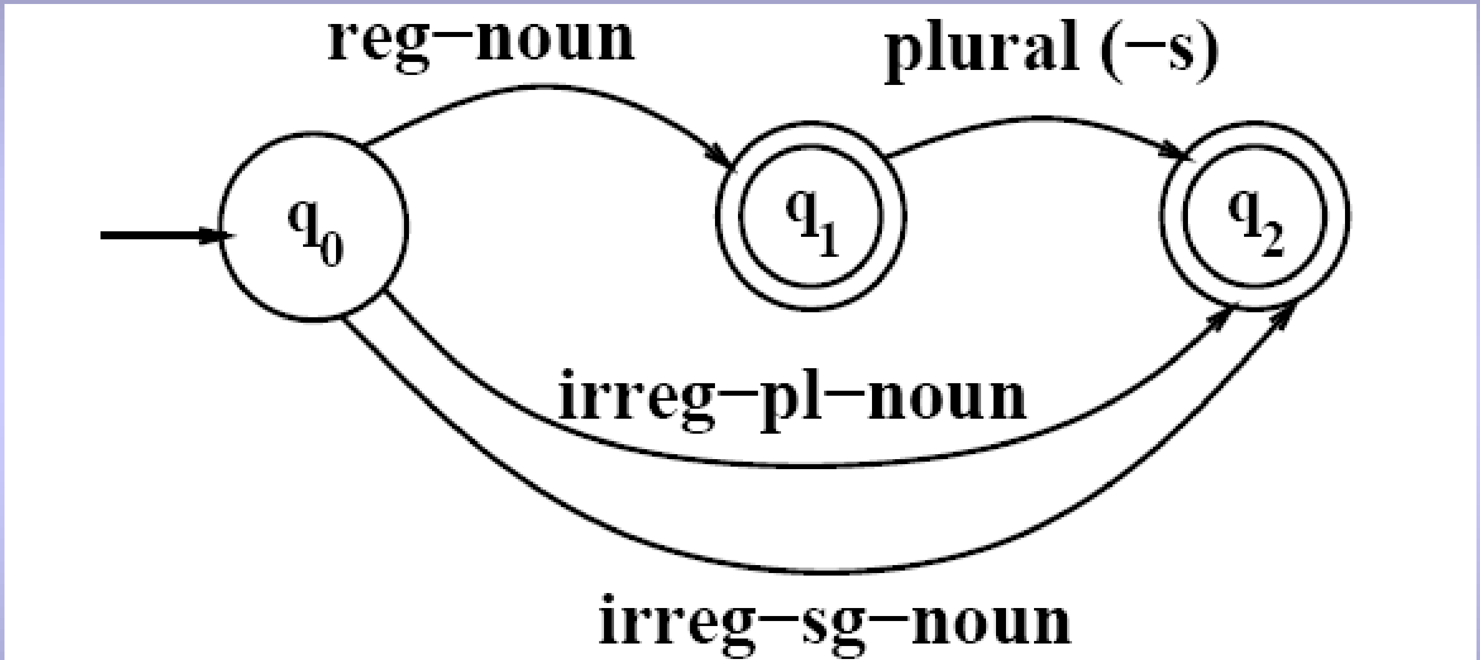
Akzeptierte Sprache: FSA *vs.* FST

- Ein FSA akzeptiert eine Sprache, die über einem Alphabet *einzelner* Symbole spezifiziert ist
- Ein FST akzeptiert eine Sprache, die über *Paaren* von Symbolen spezifiziert ist. Diese werden auch *zulässige Paare (feasible pairs)* genannt.

Konventionen zur Zwei-Ebenen-Morphologie

- Zeichenpaare für FSTs ($\alpha{:}\gamma$) werden so kodiert, dass α das Zeichen auf dem Lexikalischen Band und γ das auf dem Oberflächenband bezeichnet.
- Identische Zeichenpaare für FSTs ($\alpha{:}\alpha$) heißen Default-Paare und werden auch kurz als α notiert.
- @ (oder =) steht für ein beliebiges Symbol
- ^ ist das Morphembegrenzungssymbol
- # ist das Wortbegrenzungssymbol

FSA für Flexionsmorphologie: Englische Nomen



reg-noun	irreg-pl-noun	irreg-sg-noun	plural
aardvark cat dog fox	geese sheep mice	goose sheep mouse	-s