

Einführung in die Computerlinguistik und Sprachtechnologie

Vorlesung im WiSe 2018/19
(B-GSW-12)

Prof. Dr. Udo Hahn

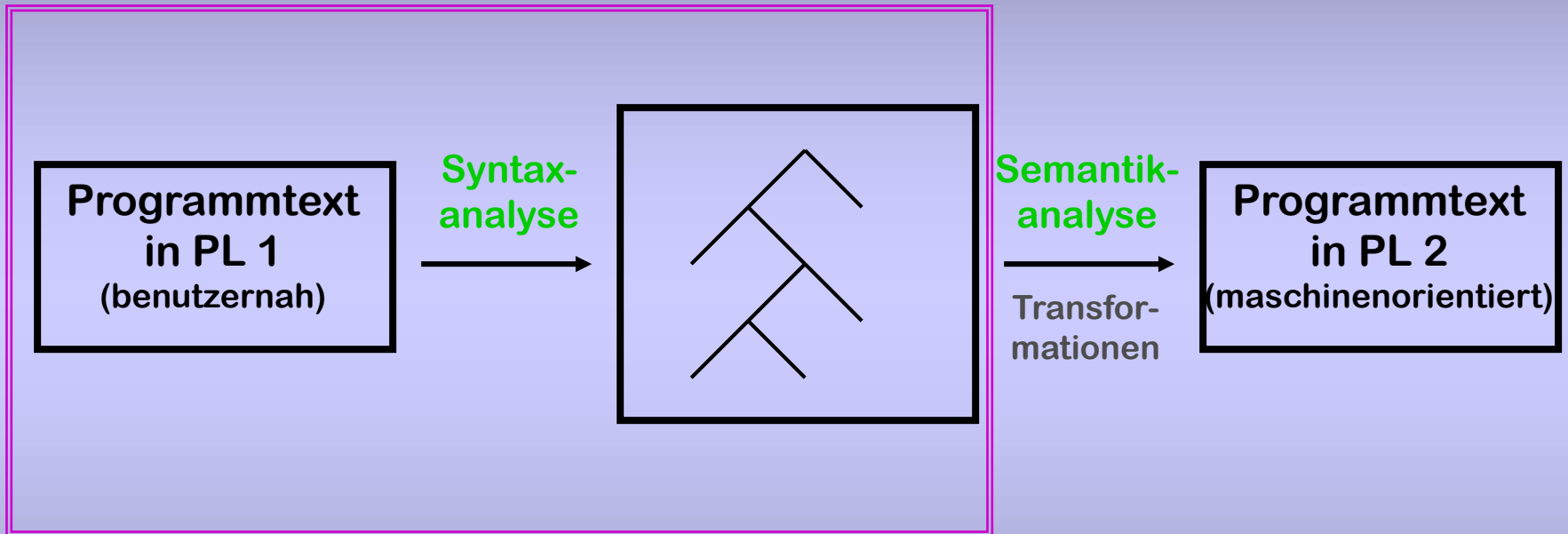
Lehrstuhl für Computerlinguistik
Institut für Germanistische Sprachwissenschaft
Friedrich-Schiller-Universität Jena

<http://www.julielab.de>

Syntaxanalyse

- **Formale** Analyse von Ausdrücken einer Sprache
 - Computerlinguistik
 - Formale Analyse von Wörtern oder Sätzen einer **natürlichen** Sprache (z.B. des Deutschen)
 - Informatik
 - Formale Analyse von Ausdrücken einer **formalen** Sprache (z.B. einer Programmiersprache)

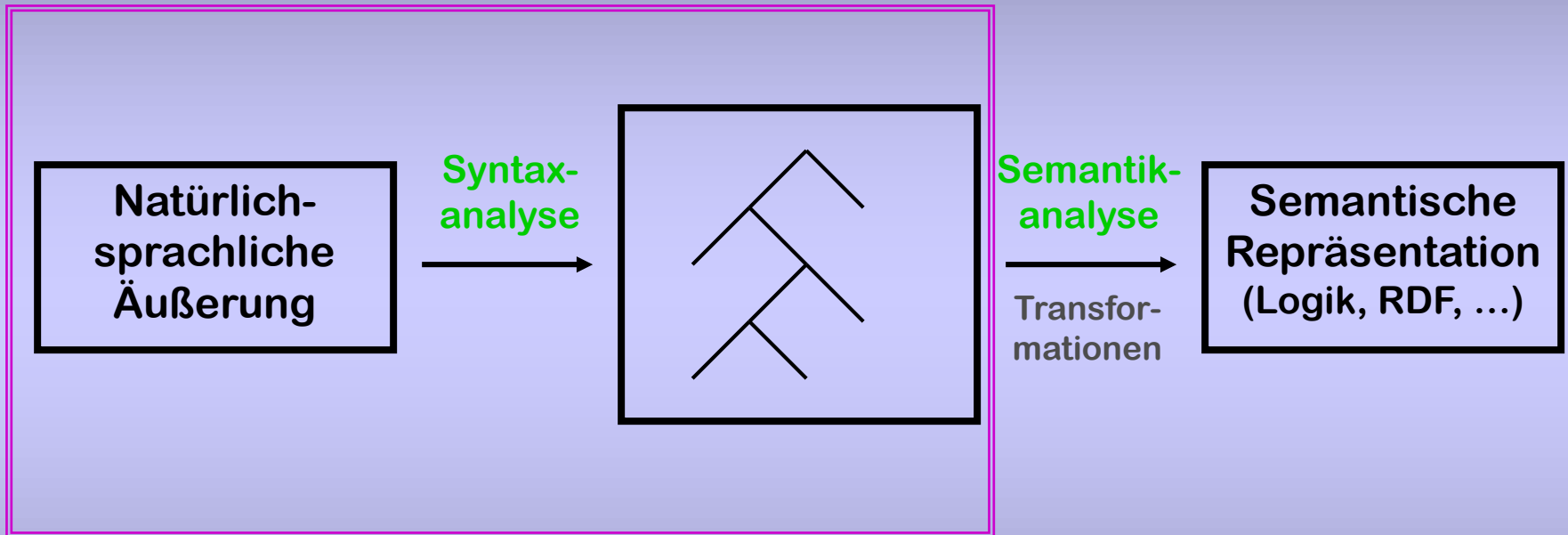
Analyse von Programmen



Aufgaben der Syntaxanalyse:

1. Syntaktisch korrekte Programme werden als korrekt erkannt
2. Syntaktisch unkorrekte Programme werden zurück gewiesen:
Fehlererkennung und -diagnose

Analyse von natürlichsprachlichen Äußerungen



Aufgaben der Syntaxanalyse:

1. Syntaktisch korrekte Äußerungen werden als korrekt erkannt
2. Syntaktisch unkorrekte Äußerungen werden zurück gewiesen:

Aber: Robustheit im Umgang mit paragrammatischen Äußerungen ist wünschenswert !

Beziehung zwischen Informatik und Computerlinguistik

- Informatik besitzt umfangreichen Methodenfundus
 - präzise beschriebene Analyseverfahren
 - Charakterisierung der formalen Eigenschaften dieser Verfahren (Entscheidbarkeit, Berechnungskomplexität)
 - mathematische Beschreibung der „Hintergrundtheorie“ (formale Grammatiken, formale Sprachen, Automaten)
- Übernahme und Adaption an NL in CL

Mengentheoretische Grundbegriffe

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

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

Beispiele:

LAUF := $\{x \mid x \text{ ist deutsches Lexem, das mit „LAUF“ beginnt} \}$

EoR := $\{x \mid x \text{ ist deutsches Lexem, das auf „E“ oder „R“ endet} \}$

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:

$\text{LAUF}^* := \{\text{Laufbahn, laufen, Lauffeuer, Laufmaschine, Laufsteg}\} \subseteq \text{LAUF}$

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

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

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 \emptyset .
 - Bemerkung: \emptyset 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

$$M \cap N := \{x \mid x \in M \text{ und } x \in N\}$$

den **Durchschnitt**

$$M \cup N := \{x \mid x \in M \text{ oder } x \in N\}$$

die **Vereinigung**

von M und N

Mengentheoretische Grundbegriffe

- **Beispiele:**

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

LAUF* \cap EoR

= { Lauffeuer, Laufmaschine }

{ Lauffeuer, Laufmaschine } \cup { Lauffeuer,
Laufpass }

= { Lauffeuer, Laufmaschine, Laufpass }

Mengentheoretische Grundbegriffe

- Wenn $I = \{1, \dots, n\}$ eine nichtleere Indexmenge 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

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

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

Beispiel:

LAUFB := { Laufbahn, Laufbursche }

LAUFS := { Laufschrift, Laufstall, Laufsteg }

LAUFB \times LAUFS = { (Laufbahn, Laufschrift), (Laufbahn, Laufstall),
(Laufbahn, Laufsteg), (Laufbursche, Laufschrift),
(Laufbursche, Laufstall), (Laufbursche, Laufsteg) } ¹²

Grundbegriffe zu Relationen

- Eine (zweistellige) **Relation** ρ zwischen zwei Mengen M_1 und M_2 ist eine Teilmenge 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

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

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

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 $i > 1$
 - $\rho^+ := \bigcup_{i \geq 1} \rho^i = \rho^1 \cup \rho^2 \cup \dots \cup \rho^n$
die **transitive Hülle** von ρ ,
 - $\rho^* := \bigcup_{i \geq 0} \rho^i = \rho^0 \cup \rho^1 \cup \rho^2 \cup \dots \cup \rho^n$
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) }

*Ist_Unterbegriff*¹ = *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) }

Ist_Unterbegriff¹ = Ist_Unterbegriff

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

Ist_Unterbegriff² = Ist_Unterbegriff¹ 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_Unterbegriff¹ = Ist_Unterbegriff

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

Ist_Unterbegriff² = Ist_Unterbegriff¹ Ist_Unterbegriff

= { (VW-Golf, PKW), (VW-PKW, 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_Unterbegriff¹ = Ist_Unterbegriff

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

Ist_Unterbegriff² = Ist_Unterbegriff¹ 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_Unterbegriff¹ = Ist_Unterbegriff

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

Ist_Unterbegriff² = Ist_Unterbegriff¹ 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) }

Ist_Unterbegriff¹ = Ist_Unterbegriff

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

Ist_Unterbegriff² = Ist_Unterbegriff¹ 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) }

Ist_Unterbegriff¹ = Ist_Unterbegriff

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

Ist_Unterbegriff² = Ist_Unterbegriff¹ 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) }

Ist_Unterbegriff² = Ist_Unterbegriff¹ Ist_Unterbegriff

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

Ist_Unterbegriff³ = Ist_Unterbegriff² 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_Unterbegriff² = Ist_Unterbegriff¹ Ist_Unterbegriff

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

Ist_Unterbegriff³ = Ist_Unterbegriff² 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) }

Ist_Unterbegriff² = Ist_Unterbegriff¹ Ist_Unterbegriff

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

Ist_Unterbegriff³ = Ist_Unterbegriff² 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_Unterbegriff² = Ist_Unterbegriff¹ Ist_Unterbegriff

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

Ist_Unterbegriff³ = Ist_Unterbegriff² 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_Unterbegriff² = Ist_Unterbegriff¹ Ist_Unterbegriff

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

Ist_Unterbegriff³ = Ist_Unterbegriff² Ist_Unterbegriff

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

Ist_Unterbegriff⁴ = Ist_Unterbegriff³ 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_Unterbegriff² = Ist_Unterbegriff¹ Ist_Unterbegriff

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

Ist_Unterbegriff³ = Ist_Unterbegriff² Ist_Unterbegriff

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

Ist_Unterbegriff⁴ = Ist_Unterbegriff³ 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) }

Ist_Unterbegriff² = Ist_Unterbegriff¹ Ist_Unterbegriff

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

Ist_Unterbegriff³ = Ist_Unterbegriff² Ist_Unterbegriff

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

Ist_Unterbegriff⁴ = Ist_Unterbegriff³ Ist_Unterbegriff

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

Ist_Unterbegriff⁵ = Ist_Unterbegriff⁴ Ist_Unterbegriff

= { (VW-Golf, Objekt) }




Ist_Unterbegriff⁶ = Ist_Unterbegriff⁵ Ist_Unterbegriff = ³¹{ }

Transitive Hülle von „Ist_Unterbegriff“

$$\bigcup_{i \geq 1} \text{Ist_Unterbegriff}^i = \text{Ist_Unterbegriff}^1 \cup \text{Ist_Unterbegriff}^2 \cup \dots \cup \text{Ist_Unterbegriff}^n$$

= { (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:

- $ABC \circ X = ABCX$
- $24 \circ 24 = 2424$
- $wort \circ stamm = wortstamm$

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:

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

Grundlagen formaler Sprachen:

Potenzen von Wörtern bzw. Wortmengen

- **Potenzen** von **Wörtern** ω bzw. **Wortmengen** \mathcal{F} sind gegeben durch:

$$- \omega^0 := \varepsilon \quad \omega^1 := \omega \quad \omega^i := \omega^{i-1} \omega, \text{ für } i \geq 1$$

$$- \mathcal{F}^0 := \{\varepsilon\} \quad \mathcal{F}^1 := \mathcal{F} \quad \mathcal{F}^i := \mathcal{F}^{i-1} \mathcal{F}, \text{ für } i \geq 1$$

Beispiele für Potenzen von Wortmengen:

- **DSilbe** := {ba, di, ko}

- **DSilbe**⁰ = { ε }, **DSilbe**¹ = {ba, di, ko}

- **DSilbe**² = **DSilbe**¹ **DSilbe**

$$= \{ \text{baba, badi, bako, diba, didi, diko, koba, kodi, koko} \}$$

- **DSilbe**³ = **DSilbe**² **DSilbe**

Grundlagen formaler Sprachen: Plus- und Sternhülle – formale Sprache

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

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

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

- 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 .

Grundbegriffe zu formalen Grammatiken

- Eine **formale Grammatik** G ist ein 4-tupel

$$G = (N, T, P, S)$$

mit

- N: das Alphabet der **Nicht-Terminalsymbole**
- T: das Alphabet der **Terminalsymbole**
- P: eine endliche Menge von **Produktionen** der Form

$\alpha \rightarrow \gamma$ (gesprochen: „ α produziert γ “) mit

$\alpha \in (N \cup T)^* N (N \cup T)^*$ und

$\gamma \in (N \cup T)^*$

- S: das **Startsymbol**, $S \in N$

$\mathcal{V} = N \cup T$, bezeichnet das **Gesamtalphabet** ($N \cap T = \emptyset$)

Beziehung zwischen formalen Grammatiken & formalen Sprachen

- Eine formale Grammatik G **erzeugt** eine formale Sprache. Der Erzeugungsprozess ist festgelegt durch eine auf \mathcal{V}^* definierte Relation „ \Rightarrow “ (gesprochen: „*ist direkt ableitbar nach*“).
Für $u, v, \gamma \in \mathcal{V}^*$ und $\alpha \in \mathcal{V}^* N \mathcal{V}^*$ gilt:
 $u \alpha v \Rightarrow u \gamma v$ genau dann, wenn $\alpha \rightarrow \gamma \in P$

Grundbegriffe zu formalen Grammatiken

- Die **transitive Hülle** der Relation „ \Rightarrow “ schreibt man
 $^+\Rightarrow$ (gesprochen: „ist nichttrivial ableitbar nach“)
- Die **reflexive und transitive Hülle** der Relation „ \Rightarrow “ schreibt man
 $^*\Rightarrow$ (gesprochen: „ist ableitbar nach“)

Grundbegriffe zu formalen Grammatiken

- Man schreibt $s \stackrel{n}{\Rightarrow} z$, um auszudrücken, dass Ketten s_0, s_1, \dots, s_n existieren mit
$$s = s_0, s_i \Rightarrow s_{i+1} \text{ für } 0 \leq i < n \text{ und } s_n = z$$
- Damit ist also
$$s \stackrel{+}{\Rightarrow} z \text{ g.d.w. } s \stackrel{n}{\Rightarrow} z \text{ für ein } n \geq 1 \quad \text{und}$$
$$s \stackrel{*}{\Rightarrow} z \text{ g.d.w. entweder } s = z \text{ oder } s \stackrel{+}{\Rightarrow} z$$

Grundbegriffe zu formalen Grammatiken

- Die von der formalen Grammatik $G = (N, T, P, S)$ erzeugte **formale Sprache** $\mathcal{L}(G)$ ist wie folgt definiert:

$$\mathcal{L}(G) := \{ \tau \mid \tau \in T^*, S \stackrel{*}{\Rightarrow} \tau,$$

$S \text{ ist Startsymbol von } G \}$

τ heißt auch **Wort** (oder **Satz**) der Sprache $\mathcal{L}(G)$.

Grundbegriffe zu formalen Grammatiken

- Abhängig von der Form der zugelassenen Produktionen definiert man vier Typen von formalen Grammatiken:
 - Eine Grammatik G heißt **Typ-0-Grammatik**, wenn die Gestalt der Produktionen nicht weiter eingeschränkt ist. D.h., sie haben die Form

$$\alpha \rightarrow \gamma$$

mit

$$\alpha \in (N \cup T)^* N (N \cup T)^* \text{ und}$$

$$\gamma \in (N \cup T)^*$$

Grundbegriffe zu formalen Grammatiken

- Eine Grammatik G heißt **Typ-1-Grammatik** (**kontextsensitive Grammatik**), wenn P nur Produktionen der Gestalt

$$\alpha \rightarrow \gamma$$

mit

$$\alpha \in (N \cup T)^* N (N \cup T)^* \text{ und}$$

$$\gamma \in (N \cup T)^*$$

$$|\alpha| \leq |\gamma|$$

(sog. *non-shrinking rules*) und eventuell die Produktion $S \rightarrow \varepsilon$ enthält (wobei letztere nur zugelassen ist, wenn das Startsymbol S in keiner Produktion auf der rechten Seite auftritt).₄₆

Grundbegriffe zu formalen Grammatiken

- Eine Grammatik G heißt **Typ-2-Grammatik** (**kontextfreie Grammatik**), wenn P nur Produktionen enthält der Gestalt

$$A \rightarrow \gamma \quad \text{mit } A \in N \text{ und } \gamma \in (N \cup T)^*$$

Grundbegriffe zu formalen Grammatiken

- Eine Grammatik G heißt **Typ-3-Grammatik** (**reguläre Grammatik**), wenn P nur Produktionen der Gestalt

$$A \rightarrow \gamma \quad \text{mit } A \in N \text{ und } \gamma \in N T^* \cup T^*$$

(sog. **links**lineare Produktionen) oder nur Produktionen der Gestalt

$$A \rightarrow \gamma \quad \text{mit } A \in N \text{ und } \gamma \in T^* N \cup T^*$$

(sog. **rechts**lineare Produktionen) enthält.

- Man spricht dann auch entsprechend von **linkslinaren** bzw. **rechtslinaren Grammatiken**.
- Eine reguläre Grammatik darf nicht Regeln nach beiden Produktionsregelmustern mischen.

Beispiel einer rechtslinearen Grammatik

$G-3 = (N, T, P, S)$ mit

$N = \{ S, A, B \}$

$T = \{ a, b \}$

$P = \{ S \rightarrow aA,$

$A \rightarrow aA,$

$A \rightarrow bbB,$

$B \rightarrow bB,$

$B \rightarrow b \}$

$S \Rightarrow aA$	mit	$S \rightarrow aA \in P$
$aA \Rightarrow abbB$	mit	$A \rightarrow bbB \in P$
$abbB \Rightarrow abbb$	mit	$B \rightarrow b \in P$

$\mathcal{L}(G-3) = \{ abbb, aabbb, aaabbb, aaaabbb, abbbb, \dots \}$
 $= a^n b^m, n \geq 1, m \geq 3$

Beispiel einer kontextfreien Grammatik

$G-2 = (N, T, P, S)$ mit

$$N = \{ S \}$$

$$T = \{ a, b \}$$

$$P = \{ S \rightarrow aSb, \\ S \rightarrow ab \}$$

$$\mathcal{L}(G-2) = \{ ab, aabb, aaabbb, aaaabbbb, \dots \} \\ = a^n b^n, n \geq 1$$

Beispiel einer kontextfreien Grammatik

$G-2 = (N, T, P, S)$ mit

$$N = \{ S \}$$

$$T = \{ a, b \}$$

$$P = \{ S \rightarrow aSb, \\ S \rightarrow ab \}$$

$$\mathcal{L}(G-2) = \{ ab, aabb, aaabbb, aaaabbbb, \dots \} \\ = a^n b^n, n \geq 1$$

$S \Rightarrow aSb$ mit $S \rightarrow aSb \in P$

$aSb \Rightarrow aabb$ mit $S \rightarrow ab \in P$

Beispiel einer kontextfreien Grammatik

$G-2 = (N, T, P, S)$ mit

$$N = \{ S \}$$

$$T = \{ a, b \}$$

$$P = \{ S \rightarrow aSb, \\ S \rightarrow ab \}$$

$$\mathcal{L}(G-2) = \{ ab, aabb, aaabbb, aaaabbbb, \dots \} \\ = a^n b^n, n \geq 1$$

$$S \stackrel{4}{\Rightarrow} aaaabbbb$$

$$S^* \Rightarrow aaaabbbb$$

$$\boxed{S} \Rightarrow \boxed{aSb}$$

s_0 s_1

Beispiel einer kontextfreien Grammatik

$G-2 = (N, T, P, S)$ mit

$$N = \{ S \}$$

$$T = \{ a, b \}$$

$$P = \{ S \rightarrow aSb, \\ S \rightarrow ab \}$$

$$\mathcal{L}(G-2) = \{ ab, aabb, aaabbb, aaaabbbb, \dots \} \\ = a^n b^n, n \geq 1$$

$$S \stackrel{4}{\Rightarrow} aaaabbbb$$

$$S^* \Rightarrow aaaabbbb$$

$$\boxed{S} \Rightarrow \boxed{aSb} \Rightarrow \boxed{aaSbb}$$

s_0 s_1 s_2

Beispiel einer kontextfreien Grammatik

$G-2 = (N, T, P, S)$ mit

$$N = \{ S \}$$

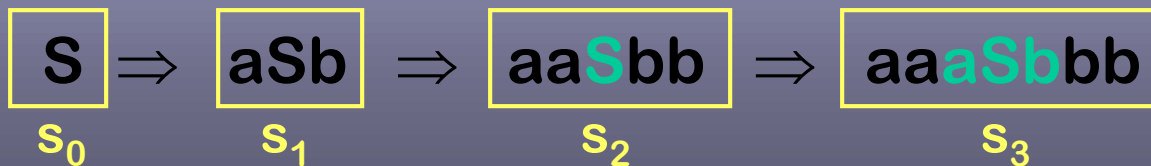
$$T = \{ a, b \}$$

$$P = \{ S \rightarrow aSb, \\ S \rightarrow ab \}$$

$$\mathcal{L}(G-2) = \{ ab, aabb, aaabbb, aaaabbbb, \dots \} \\ = a^n b^n, n \geq 1$$

$S \xRightarrow{4} aaaabbbb$

$S^* \Rightarrow aaaabbbb$



Beispiel einer kontextfreien Grammatik

$G-2 = (N, T, P, S)$ mit

$$N = \{ S \}$$

$$T = \{ a, b \}$$

$$P = \{ S \rightarrow aSb,$$

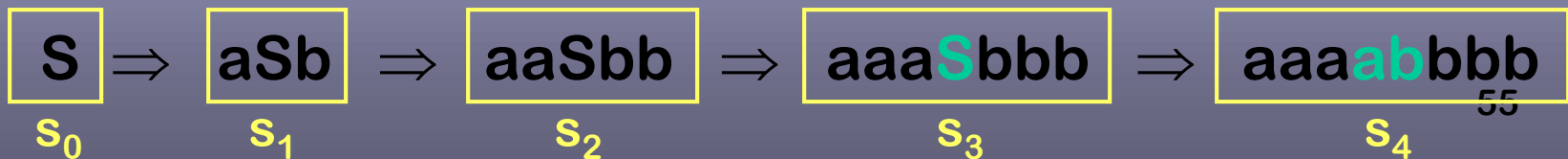
$$S \rightarrow ab \}$$

$$\mathcal{L}(G-2) = \{ ab, aabb, aaabbb, aaaabbbb, \dots \}$$

$$= a^n b^n, n \geq 1$$

$$S \stackrel{4}{\Rightarrow} aaaabbbb$$

$$S^* \Rightarrow aaaabbbb$$



Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC,$
 $CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC,$
 $aB \rightarrow ab,$
 $bB \rightarrow bb,$
 $C \rightarrow c \}$

$\mathcal{L}(G-1) = \{ abc, aabbcc, aaabbbccc, \dots \}$
 $= a^n b^n c^n, n \geq 1$

Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC,$
 $CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC,$
 $aB \rightarrow ab, bB \rightarrow bb, C \rightarrow c \}$

$S^* \Rightarrow aaabbbccc$

Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC,$
 $CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC,$
 $aB \rightarrow ab, bB \rightarrow bb, C \rightarrow c \}$

$S \Rightarrow aSBC$

Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC,$
 $CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC,$
 $aB \rightarrow ab, bB \rightarrow bb, C \rightarrow c \}$

$S \Rightarrow aSBC \Rightarrow aaSBCBC$

Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC, \\ CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC, \\ aB \rightarrow ab, bB \rightarrow bb, C \rightarrow c \}$

$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaBCBCBC$

Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC,$

$CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC,$

$aB \rightarrow ab, bB \rightarrow bb, C \rightarrow c \}$

$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaB**CB**CBC \Rightarrow$

$\dots \Rightarrow aaaB**BC**CBC$

Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC,$

$CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC,$

$aB \rightarrow ab, bB \rightarrow bb, C \rightarrow c \}$

$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC \Rightarrow$
 $\dots \Rightarrow aaaBBCBC \Rightarrow \dots \Rightarrow aaaBBCBC$

Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC,$

$CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC,$

$aB \rightarrow ab, bB \rightarrow bb, C \rightarrow c \}$

$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC \Rightarrow$

$\dots \Rightarrow aaaBBCBC \Rightarrow \dots \Rightarrow aaaBBBCBC \Rightarrow$

$\dots \Rightarrow aaaBBBCCC$

Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC,$

$CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC,$

$aB \rightarrow ab, bB \rightarrow bb, C \rightarrow c \}$

$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC \Rightarrow$

$\dots \Rightarrow aaaBBCCBC \Rightarrow \dots \Rightarrow aaaBBCCBC \Rightarrow$

$\dots \Rightarrow aa**aB**BBCCC \Rightarrow aa**ab**BBCCC$

Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC,$
 $CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC,$
 $aB \rightarrow ab, bB \rightarrow bb, C \rightarrow c \}$

$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC \Rightarrow$
 $\dots \Rightarrow aaaBBCCBC \Rightarrow \dots \Rightarrow aaaBBCCBC \Rightarrow$
 $\dots \Rightarrow aaaBBBCCC \Rightarrow aaa**b**BCCC \Rightarrow$
 $aaa**bb**BCCC$

Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC,$
 $CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC,$
 $aB \rightarrow ab, bB \rightarrow bb, C \rightarrow c \}$

$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC \Rightarrow$
 $\dots \Rightarrow aaaBBCCBC \Rightarrow \dots \Rightarrow aaaBBCCBC \Rightarrow$
 $\dots \Rightarrow aaaBBBCCC \Rightarrow aaabBBCCC \Rightarrow$
 $aaab**b**CCC \Rightarrow aaab**bb**CCC$

Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC,$
 $CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC,$
 $aB \rightarrow ab, bB \rightarrow bb, C \rightarrow c \}$

$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC \Rightarrow$
 $\dots \Rightarrow aaaBBCCBC \Rightarrow \dots \Rightarrow aaaBBCCBC \Rightarrow$
 $\dots \Rightarrow aaaBBBCCC \Rightarrow aaabBBCCC \Rightarrow$
 $aaabbBCCC \Rightarrow aaabbbCCC \Rightarrow \dots \Rightarrow aaabbbccc$

Beispiel einer kontextsensitiven Grammatik

$G-1 = (N, T, P, S)$ mit

$N = \{ S, B, C, X \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSBC, S \rightarrow aBC,$
 $CB \rightarrow XB, XB \rightarrow XC, XC \rightarrow BC,$
 $aB \rightarrow ab, bB \rightarrow bb, C \rightarrow c \}$

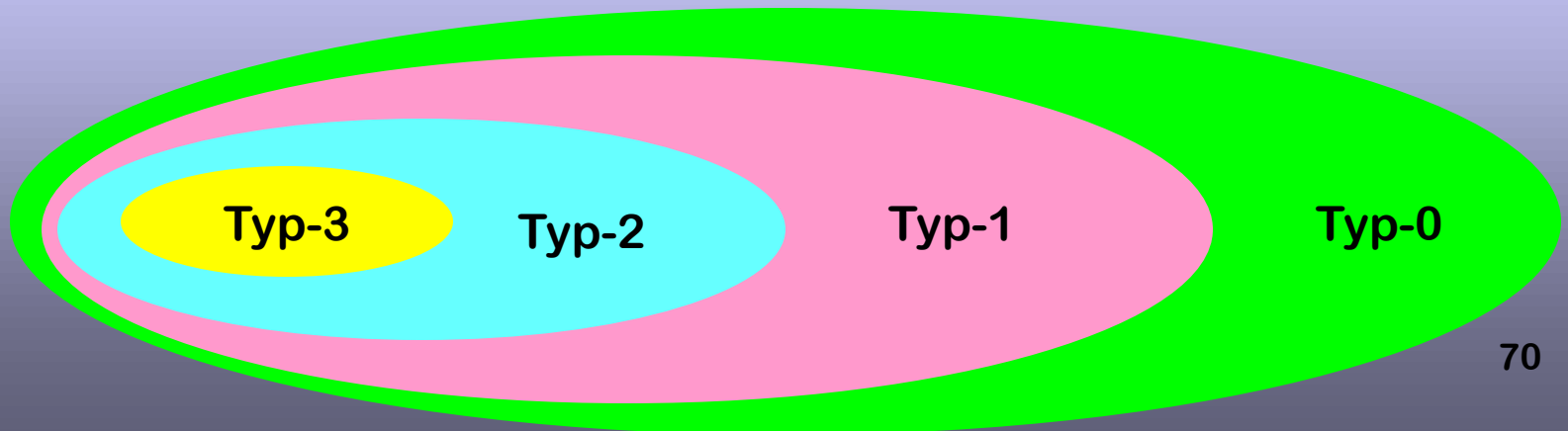
$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC \Rightarrow$
 $\dots \Rightarrow aaaBBCBC \Rightarrow \dots \Rightarrow aaaBBCBC \Rightarrow$
 $\dots \Rightarrow aaaBBBCCC \Rightarrow aaabBBCCC \Rightarrow$
 $aaabbBCCC \Rightarrow aaabbbCCC \Rightarrow \dots \text{aaabbbccc}$

Typen formaler Sprachen (und ihr Bezug zu Grammatik-Typen)

- Eine **formale Sprache** heißt vom **Typ 0, 1, 2** oder **3**, wenn sie von einer Grammatik des entsprechenden Typs erzeugt werden kann.
 - Eine Typ-1-Sprache heißt auch **kontextsensitive Sprache**.
 - Eine Typ-2-Sprache heißt auch **kontextfreie Sprache**.
 - Eine Typ-3-Sprache heißt auch **reguläre Sprache**.

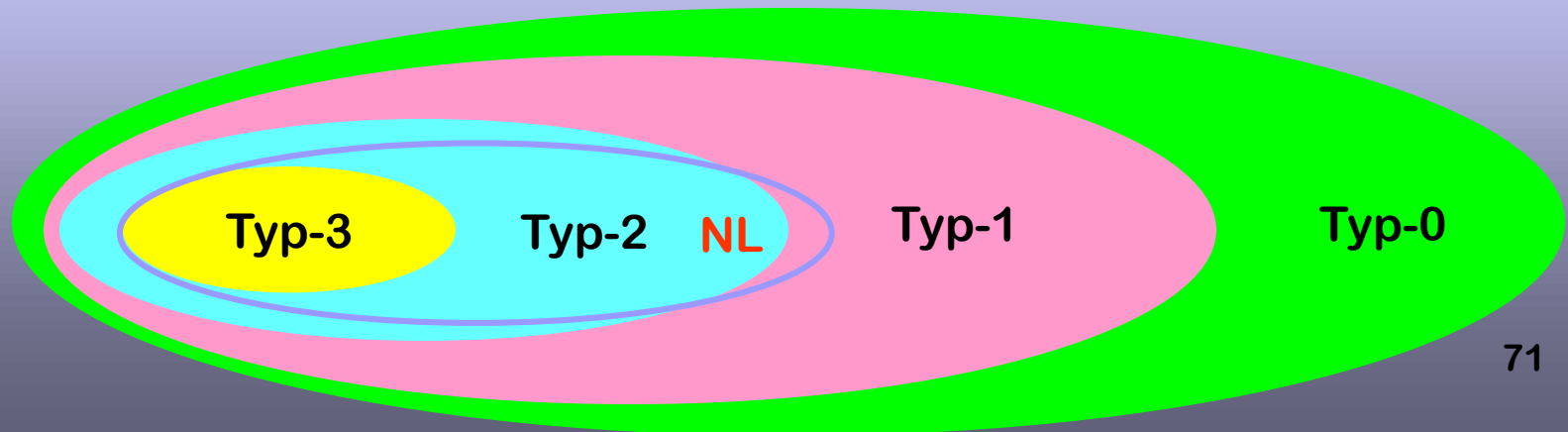
Chomsky-Hierarchie formaler Sprachen

- Für jedes Alphabet Σ (mit mindestens zwei Symbolen) ist die Menge der Typ- i -Sprachen über Σ für $i = 0, 1, 2$ jeweils (echte) Obermenge der Typ- $[i+1]$ -Sprachen über Σ . Die damit gegebene Hierarchie von formalen Sprachen heißt **Chomsky-Hierarchie**.



Natürliche Sprachen als formale Sprachen

- NLs sind keine Typ-3-Sprachen
- NLs sind überwiegend (Englisch, Deutsch, Französisch, Spanisch, ...) Typ-2-Sprachen
- Einige wenige NLs sind sicher (Schweizer Deutsch) bzw. vermutlich (Niederländisch, Bambara [Mali]) milde Typ-1-Sprachen



Endlicher Automat (1/2)

- Ein endlicher Automat (*finite-state automaton, 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 ausgezeichneten **Endzustände** befindet, hat er die Eingabe **akzeptiert**, sonst nicht.

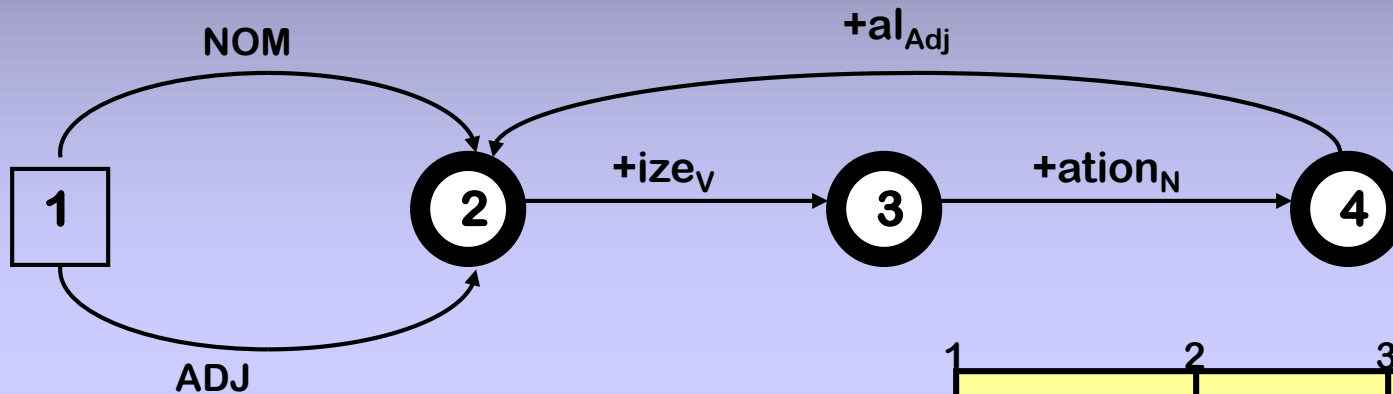
Endlicher Automat (2/2)

- Ein FSA besteht i.A. aus einem **Eingabeband**, auf dem der zu akzeptierende Ausdruck steht, und einem endlichen **Steuerungsmechanismus**, 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 $(Q, \Sigma, \delta, q_0, F)$ mit
 - **Q**: endliche Menge von (Steuerungs-)Zuständen
 $q_0, q_1, q_2, \dots, q_n$
 - **Σ** : endliches Eingabealphabet
 - **δ** : Zustandsübergangsfunktion, die das Steuerungsverhalten des FSA determiniert
$$\delta : Q \times \Sigma \mapsto \wp(Q)$$
 - **$q_0 \in Q$** : der ausgezeichnete Startzustand
 - **$F \subseteq Q$** : Menge der ausgezeichneten Endzustände

Automat für Dederivation



1	2	3	4
hospital	+ize _v	+ation _N	ε

NOM: hospital, motor, category, ...

ADJ: moral, concrete, tender, ...

n Anfangszustand

n möglicher Endzustand

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

Deterministischer FSA- Erkennungsalgorithmus

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

Erkennungsalgorithmus für deterministischen FSA

Funktion D-Erkennen(\downarrow Band, \downarrow FSA) = „accept“ oder „reject“

Index \Leftarrow Bandanfang

AktualZustand \Leftarrow 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 \Leftarrow Zustandstransitionstabelle[AktualZustand, Band(Index)]
 Index \Leftarrow Index + 1

LOOPEND

Kostenrechnung für det. FSA- Erkennungsalgorithmus

Funktion D-Erkennen(\downarrow Band, \downarrow FSA) = „accept“ oder „reject“

Index \Leftarrow Bandanfang

AktualZustand \Leftarrow 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 \Leftarrow Zustandstransitionstabelle[AktualZustand, Band(Index)]

Index \Leftarrow Index + 1

LOOPEND

1

1

n

1

1

1







1

1

Charakteristische Problemgrößen beim Parsing

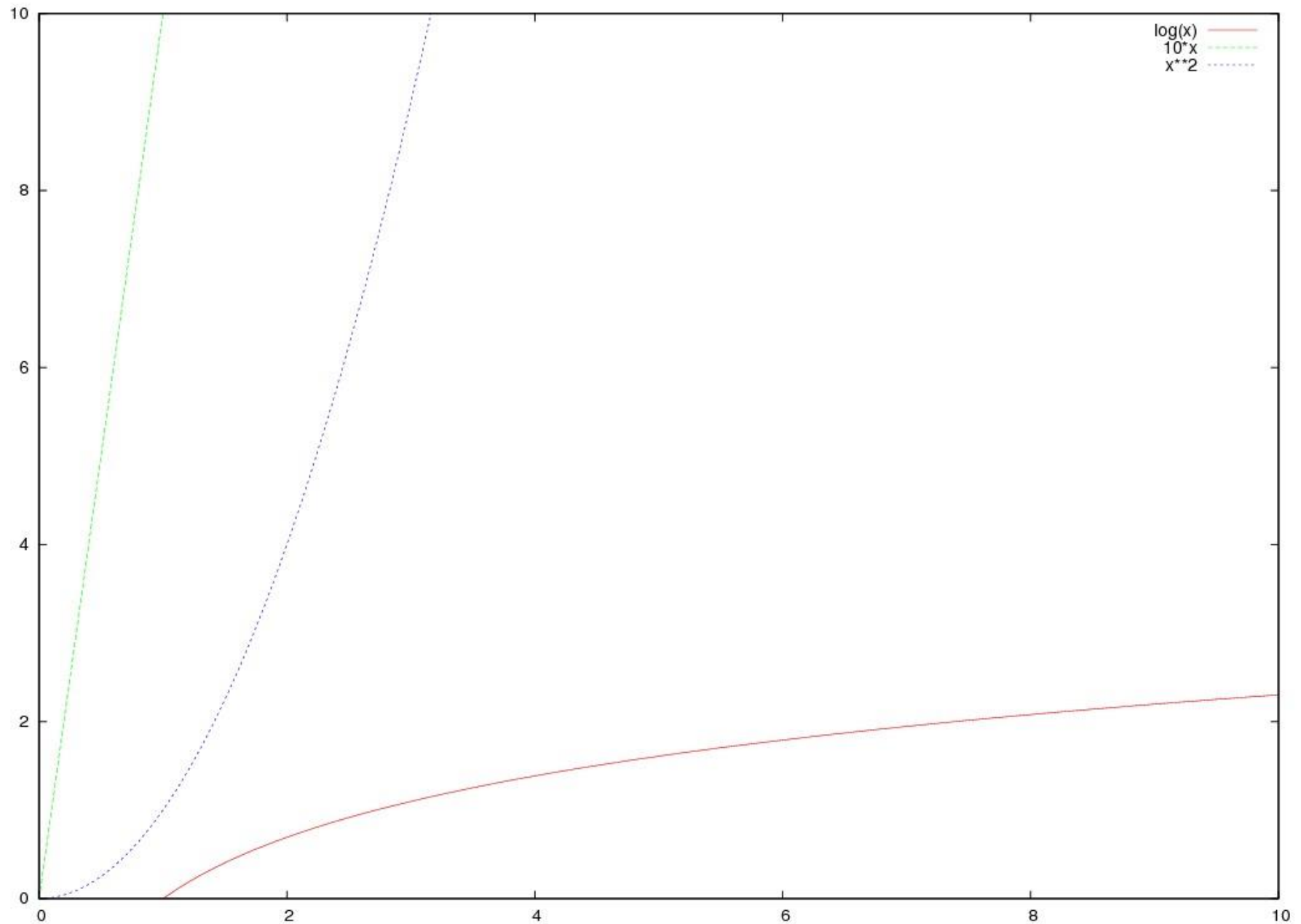
- n ist die Länge der Eingabe (Wort, Satz)
- n ist die Kardinalität der Produktionsregelmenge
- Hinweis: n ist vom (Parsing-)Algorithmus abhängig

Komplexitätsklassen

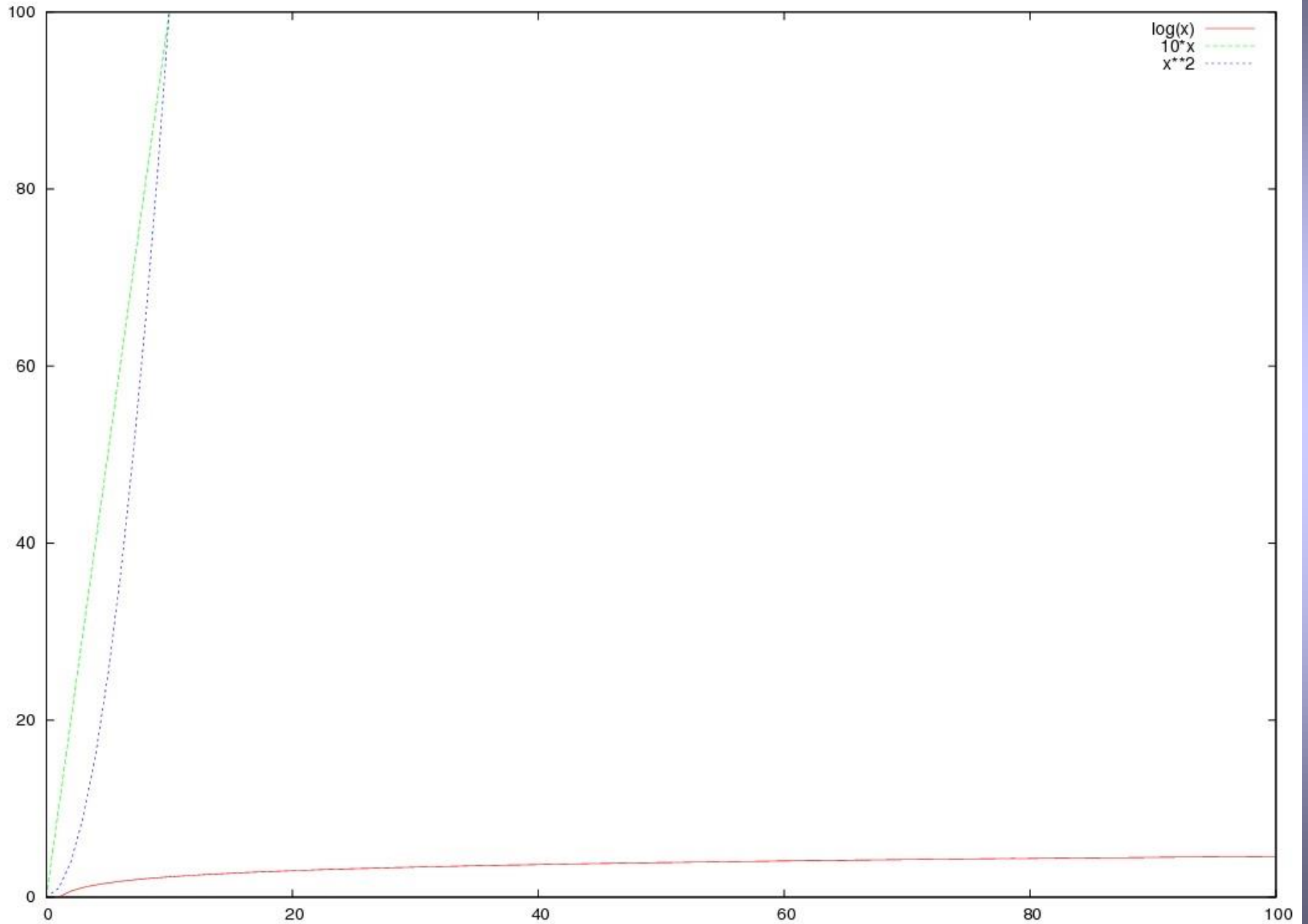
- $O(1)$ konstant 
- $O(\log n)$ logarithmisch 
- $O(n)$ linear 
- $O(n^k)$ polynomial ($k \in [2,4]$) 
- $O(n^k)$ polynomial ($k > 4$) 
- $O(k^n)$ exponentiell 

– wobei n die Problemgröße ist

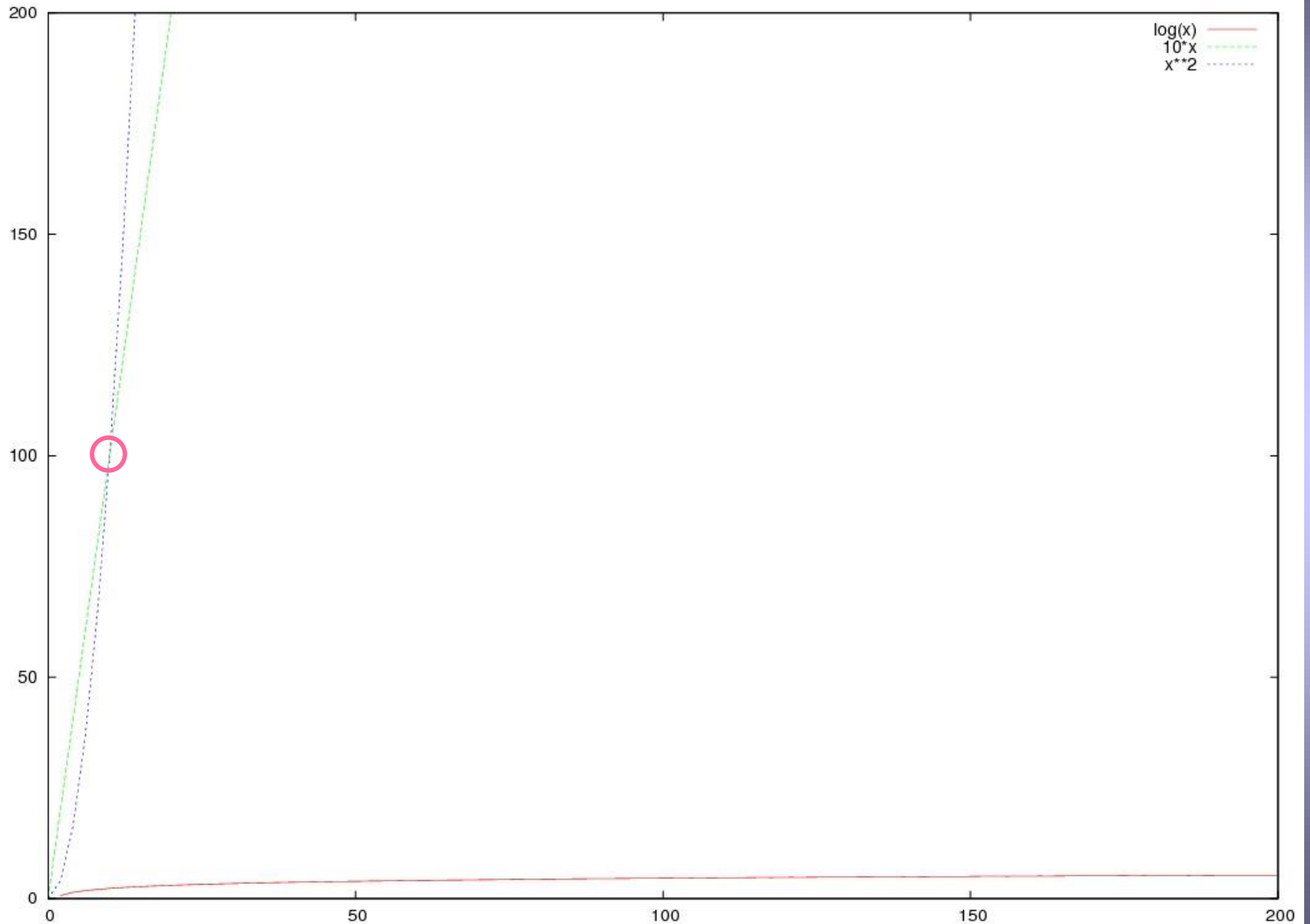
(Lauf-)Zeit-Klassen



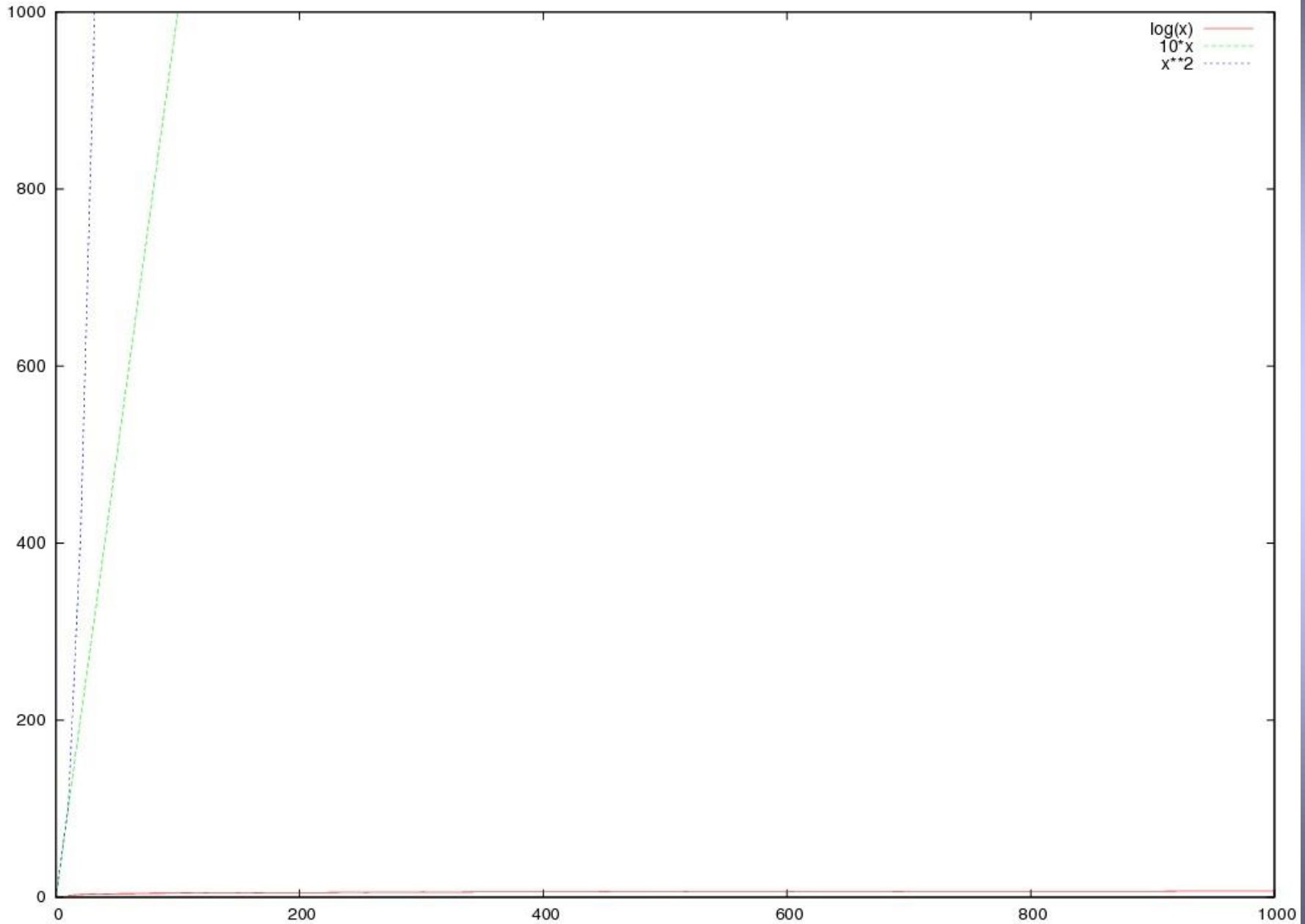
(Lauf-)Zeit-Klassen



(Lauf-)Zeit-Klassen



(Lauf-)Zeit-Klassen



Implikationen für die automatische Sprachanalyse

- NLS sind keine **Typ-3-Sprachen**
 - Trotzdem werden **endliche Automaten (FSA)** für NLP-Analytik eingesetzt
 - **Lineare** Laufzeit ($O(n)$)
- NLS sind überwiegend (Englisch, Deutsch, Französisch, Spanisch, ...) **Typ-2-Sprachen**
 - **Kellerautomaten** als Basismodell
 - Syntaxanalyse in max. **kubischer** Laufzeitkomplexität ($O(n^3)$)
- Einige wenige NLS sind sicher (Schweizer Deutsch) bzw. vermutlich (Niederländisch, Bambara [Mali]) milde **Typ-1-Sprachen**
 - Syntaxanalyse in max. $O(n^6)$ Laufzeitkomplexität
 - Grammatikmodell: Tree Adjoining Grammar (TAG)