SS 2005

Einführung in die Informatik IV

Ernst W. Mayr

Fakultät für Informatik
TU München

http://www14.in.tum.de/lehre/2005SS/info4/index.html.de

11. April 2005

Kapitel 0 Organisatorisches

- Vorlesungen:
  - Mo 10:30–12:00, Fr 08:30–10:00 HS I (00.02.001)
- Übung:
  - 3SWS Tutorübung: bitte anmelden unter
    `https://grundstudium.in.tum.de/`
- Umfang:
  - 4V+3TÜ, 9 ECTS-Punkte
- Sprechstunde:
  - Fr 11:00–12:00 oder nach Vereinbarung

Kapitel 0 Organisatorisches

- Vorlesungen:
  - Mo 10:30–12:00, Fr 08:30–10:00 HS I (00.02.001)
- Übung:
  - 3SWS Tutorübung: bitte anmelden unter https://grundstudium.in.tum.de/
- Umfang:
  - 4V+3TÜ, 9 ECTS-Punkte
- Sprechstunde:
  - Fr 11:00–12:00 oder nach Vereinbarung

Kapitel 0 Organisatorisches

- Vorlesungen:
  - Mo 10:30–12:00, Fr 08:30–10:00 HS I (00.02.001)
- Übung:
  - 3SWS Tutorübung: bitte anmelden unter https://grundstudium.in.tum.de/
- Umfang:
  - 4V+3TÜ, 9 ECTS-Punkte
- Sprechstunde:
  - Fr 11:00–12:00 oder nach Vereinbarung

Kapitel 0 Organisatorisches

- Vorlesungen:
  - Mo 10:30–12:00, Fr 08:30–10:00 HS I (00.02.001)
- Übung:
  - 3SWS Tutorübung: bitte anmelden unter https://grundstudium.in.tum.de/
- Umfang:
  - 4V+3TÜ, 9 ECTS-Punkte
- Sprechstunde:
  - Fr 11:00–12:00 oder nach Vereinbarung

Kapitel 0 Organisatorisches

- Vorlesungen:
  - Mo 10:30–12:00, Fr 08:30–10:00 HS I (00.02.001)
- Übung:
  - 3SWS Tutorübung: bitte anmelden unter https://grundstudium.in.tum.de/
- Umfang:
  - 4V+3TÜ, 9 ECTS-Punkte
- Sprechstunde:
  - Fr 11:00–12:00 oder nach Vereinbarung

- Vorkenntnisse:
  - Einführung in die Informatik I/II/(III)
  - Diskrete Strukturen I
- Weiterführende Vorlesungen:
  - Effiziente Algorithmen und Datenstrukturen
  - Automaten, Formale Sprachen, Berechenbarkeit und Entscheidbarkeit
  - Komplexitätstheorie
  - ...
- Webseite:

  http://wwwmayr.in.tum.de/lehre/2005SS/info4/

- Vorkenntnisse:
  - Einführung in die Informatik I/II/(III)
  - Diskrete Strukturen I
- Weiterführende Vorlesungen:
  - Effiziente Algorithmen und Datenstrukturen
  - Automaten, Formale Sprachen, Berechenbarkeit und Entscheidbarkeit
  - Komplexitätstheorie
  - ...
- Webseite:

  http://wwwmayr.in.tum.de/lehre/2005SS/info4/

- Vorkenntnisse:
  - Einführung in die Informatik I/II/(III)
  - Diskrete Strukturen I
- Weiterführende Vorlesungen:
  - Effiziente Algorithmen und Datenstrukturen
  - Automaten, Formale Sprachen, Berechenbarkeit und Entscheidbarkeit
  - Komplexitätstheorie
  - ...
- Webseite:

  http://wwwmayr.in.tum.de/lehre/2005SS/info4/

- Übungsleitung:
  - Moritz Maaß, MI 03.09.037 (maass@in.tum.de)
  - Hanjo Täubig, MI 03.09.039 (taeubig@in.tum.de)
- Sekretariat:
  - Frau Schmidt, MI 03.09.052 (schmiann@in.tum.de)

- Übungsleitung:
  - Moritz Maaß, MI 03.09.037 (maass@in.tum.de)
  - Hanjo Täubig, MI 03.09.039 (taeubig@in.tum.de)
- Sekretariat:
  - Frau Schmidt, MI 03.09.052 (schmiann@in.tum.de)

- Übungsaufgaben und Klausur:
  - Ausgabe jeweils am Freitag in der Vorlesung
  - Abgabe eine Woche später
  - Besprechung in der Tutorübung
- Klausur:

- Übungsaufgaben und Klausur:
  - Ausgabe jeweils am Freitag in der Vorlesung
  - Abgabe eine Woche später
  - Besprechung in der Tutorübung
- Klausur:

- Übungsaufgaben und Klausur:
  - Ausgabe jeweils am Freitag in der Vorlesung
  - Abgabe eine Woche später
  - Besprechung in der Tutorübung
- Klausur:
  - Zwischenklausur (50% Gewicht) am [illegible]
  - Endklausur (50% Gewicht) am [illegible]
  - Wiederholungsklausur am 12. Oktober 2015
  - bei den Klausuren sind keine Hilfsmittel außer einem handbeschriebenen DIN A4-Blatt zugelassen

- Übungsaufgaben und Klausur:
  - Ausgabe jeweils am Freitag in der Vorlesung
  - Abgabe eine Woche später
  - Besprechung in der Tutorübung
- Klausur:
  - Zwischenklausur (50% Gewicht) am 21. Mai 2005, 9–12 Uhr
  - Endklausur (50% Gewicht) am 13. Juli 2005, 16–19 Uhr
  - Wiederholungsklausur am 12. Oktober 2005
  - bei den Klausuren sind keine Hilfsmittel außer einem handbeschriebenen DIN A4-Blatt zugelassen

- Übungsaufgaben und Klausur:
  - Ausgabe jeweils am Freitag in der Vorlesung
  - Abgabe eine Woche später
  - Besprechung in der Tutorübung
- Klausur:
  - Zwischenklausur (50% Gewicht) am 21. Mai 2005, 9–12 Uhr
  - Endklausur (50% Gewicht) am 13. Juli 2005, 16–19 Uhr
  - Wiederholungsklausur am 13. Oktober 2005
  - bei den Klausuren sind *keine* Hilfsmittel außer einem handbeschriebenen DIN-A4-Blatt zugelassen

- Übungsaufgaben und Klausur:
  - Ausgabe jeweils am Freitag in der Vorlesung
  - Abgabe eine Woche später
  - Besprechung in der Tutorübung
- Klausur:
  - Zwischenklausur (50% Gewicht) am 21. Mai 2005, 9–12 Uhr
  - Endklausur (50% Gewicht) am 13. Juli 2005, 16–19 Uhr
  - Wiederholungsklausur am 13. Oktober 2005
  - bei den Klausuren sind *keine* Hilfsmittel außer einem handbeschriebenen DIN-A4-Blatt zugelassen

- Übungsaufgaben und Klausur:
  - Ausgabe jeweils am Freitag in der Vorlesung
  - Abgabe eine Woche später
  - Besprechung in der Tutorübung
- Klausur:
  - Zwischenklausur (50% Gewicht) am 21. Mai 2005, 9–12 Uhr
  - Endklausur (50% Gewicht) am 13. Juli 2005, 16–19 Uhr
  - Wiederholungsklausur am 13. Oktober 2005
  - bei den Klausuren sind *keine* Hilfsmittel außer einem handbeschriebenen DIN-A4-Blatt zugelassen

- Übungsaufgaben und Klausur:
  - Ausgabe jeweils am Freitag in der Vorlesung
  - Abgabe eine Woche später
  - Besprechung in der Tutorübung
- Klausur:
  - Zwischenklausur (50% Gewicht) am 21. Mai 2005, 9–12 Uhr
  - Endklausur (50% Gewicht) am 13. Juli 2005, 16–19 Uhr
  - Wiederholungsklausur am 13. Oktober 2005
  - bei den Klausuren sind *keine* Hilfsmittel außer einem handbeschriebenen DIN-A4-Blatt zugelassen

- Übungsaufgaben und Klausur:
  - Ausgabe jeweils am Freitag in der Vorlesung
  - Abgabe eine Woche später
  - Besprechung in der Tutorübung
- Klausur:
  - Zwischenklausur (50% Gewicht) am 21. Mai 2005, 9–12 Uhr
  - Endklausur (50% Gewicht) am 13. Juli 2005, 16–19 Uhr
  - Wiederholungsklausur am 13. Oktober 2005
  - bei den Klausuren sind *keine* Hilfsmittel außer einem handbeschriebenen DIN-A4-Blatt zugelassen

1. Ziel der Vorlesung

Der Zweck dieser Vorlesung ist das Studium fundamentaler Konzepte in der Theoretischen Informatik. Dies umfasst das Studium der Grundlagen formaler Sprachen und Automaten, von Berechnungsmodellen und Fragen der Entscheidbarkeit, die Diskussion elementarer Algorithmen und Datenstrukturen sowie einiger grundlegender Konzepte der Komplexitätstheorie.

Themengebiete werden also sein:

- Berechenbarkeitstheorie
  - Betrachtung und Untersuchung der Grenzen, was Rechner überhaupt können
- Komplexitätstheorie
  - Studium der Grenzen, was Rechner mit begrenzten Ressourcen leisten können
  - Herleitung *oberer* und *unterer* Schranken
- Automatentheorie
- Grammatiken
- Algorithmen

Themengebiete werden also sein:

- Berechenbarkeitstheorie
  - Betrachtung und Untersuchung der Grenzen, was Rechner überhaupt können
- Komplexitätstheorie
  - Studium der Grenzen, was Rechner mit begrenzten Ressourcen leisten können
  - Herleitung *oberer* und *unterer* Schranken
- Automatentheorie
- Grammatiken
- Algorithmen

Themengebiete werden also sein:

- Berechenbarkeitstheorie
  - Betrachtung und Untersuchung der Grenzen, was Rechner überhaupt können
- Komplexitätstheorie
  - Studium der Grenzen, was Rechner mit begrenzten Ressourcen leisten können
  - Herleitung *oberer* und *unterer* Schranken
- Automatentheorie
  - Rechner als endliche Systeme mit endlichem oder unendlichem Speicher
- Grammatiken
- Algorithmen

Themengebiete werden also sein:

- Berechenbarkeitstheorie
  - Betrachtung und Untersuchung der Grenzen, was Rechner überhaupt können
- Komplexitätstheorie
  - Studium der Grenzen, was Rechner mit begrenzten Ressourcen leisten können
  - Herleitung *oberer* und *unterer* Schranken
- Automatentheorie
  - Rechner als endliche Systeme mit endlichem oder unendlichem Speicher
- Grammatiken
- Algorithmen

Themengebiete werden also sein:

- Berechenbarkeitstheorie
  - Betrachtung und Untersuchung der Grenzen, was Rechner überhaupt können
- Komplexitätstheorie
  - Studium der Grenzen, was Rechner mit begrenzten Ressourcen leisten können
  - Herleitung *oberer* und *unterer* Schranken
- Automatentheorie
  - Rechner als endliche Systeme mit endlichem oder unendlichem Speicher
- Grammatiken
  - Aufbau von Programmiersprachen, Ausdruckskraft, Effizienz der Syntaxanalyse
- Algorithmen

Themengebiete werden also sein:

- Berechenbarkeitstheorie
  - Betrachtung und Untersuchung der Grenzen, was Rechner überhaupt können
- Komplexitätstheorie
  - Studium der Grenzen, was Rechner mit begrenzten Ressourcen leisten können
  - Herleitung *oberer* und *unterer* Schranken
- Automatentheorie
  - Rechner als endliche Systeme mit endlichem oder unendlichem Speicher
- Grammatiken
  - Aufbau von Programmiersprachen, Ausdruckskraft, Effizienz der Syntaxanalyse
- Algorithmen

Themengebiete werden also sein:

- Berechenbarkeitstheorie
  - Betrachtung und Untersuchung der Grenzen, was Rechner überhaupt können
- Komplexitätstheorie
  - Studium der Grenzen, was Rechner mit begrenzten Ressourcen leisten können
  - Herleitung *oberer* und *unterer* Schranken
- Automatentheorie
  - Rechner als endliche Systeme mit endlichem oder unendlichem Speicher
- Grammatiken
  - Aufbau von Programmiersprachen, Ausdruckskraft, Effizienz der Syntaxanalyse
- Algorithmen
  - grundlegende effiziente Verfahren und Datenstrukturen

Themengebiete werden also sein:

- Berechenbarkeitstheorie
  - Betrachtung und Untersuchung der Grenzen, was Rechner überhaupt können
- Komplexitätstheorie
  - Studium der Grenzen, was Rechner mit begrenzten Ressourcen leisten können
  - Herleitung *oberer* und *unterer* Schranken
- Automatentheorie
  - Rechner als endliche Systeme mit endlichem oder unendlichem Speicher
- Grammatiken
  - Aufbau von Programmiersprachen, Ausdruckskraft, Effizienz der Syntaxanalyse
- Algorithmen
  - grundlegende (effiziente) Verfahren und Datenstrukturen

Themengebiete werden also sein:

- Berechenbarkeitstheorie
  - Betrachtung und Untersuchung der Grenzen, was Rechner überhaupt können
- Komplexitätstheorie
  - Studium der Grenzen, was Rechner mit begrenzten Ressourcen leisten können
  - Herleitung *oberer* und *unterer* Schranken
- Automatentheorie
  - Rechner als endliche Systeme mit endlichem oder unendlichem Speicher
- Grammatiken
  - Aufbau von Programmiersprachen, Ausdruckskraft, Effizienz der Syntaxanalyse
- Algorithmen
  - grundlegende (effiziente) Verfahren und Datenstrukturen

Themengebiete werden also sein:

- Berechenbarkeitstheorie
  - Betrachtung und Untersuchung der Grenzen, was Rechner überhaupt können
- Komplexitätstheorie
  - Studium der Grenzen, was Rechner mit begrenzten Ressourcen leisten können
  - Herleitung *oberer* und *unterer* Schranken
- Automatentheorie
  - Rechner als endliche Systeme mit endlichem oder unendlichem Speicher
- Grammatiken
  - Aufbau von Programmiersprachen, Ausdruckskraft, Effizienz der Syntaxanalyse
- Algorithmen
  - grundlegende (effiziente) Verfahren und Datenstrukturen

2. Wesentliche Techniken und Konzepte

2. Wesentliche Techniken und Konzepte

- Formalisierung
  - Rechner werden durch mathematische Objekte nachgebildet
  - zu lösende Aufgaben werden mengentheoretisch als Problem definiert
  - die Abfolge von Berechnungsschritten wird formalisiert
  - die quantitative Bestimmung der Komplexität eines Verfahrens bzw. eines Problems wird festgelegt
- Simulation

2. Wesentliche Techniken und Konzepte

- Formalisierung
  - Rechner werden durch mathematische Objekte nachgebildet
  - zu lösende Aufgaben werden mengentheoretisch als Problem definiert
  - die Abfolge von Berechnungsschritten wird formalisiert
  - die quantitative Bestimmung der Komplexität eines Verfahrens bzw. eines Problems wird festgelegt
- Simulation

2. Wesentliche Techniken und Konzepte

- Formalisierung
  - Rechner werden durch mathematische Objekte nachgebildet
  - zu lösende Aufgaben werden mengentheoretisch als Problem definiert
  - die Abfolge von Berechnungsschritten wird formalisiert
  - die quantitative Bestimmung der Komplexität eines Verfahrens bzw. eines Problems wird festgelegt
- Simulation
  - Verfahren zur Ersetzung eines Programms in einem Formalismus A durch ein Programm in einem anderen Formalismus B bei unveränderten Ein-/Ausgabeverhalten

2. Wesentliche Techniken und Konzepte

- Formalisierung
  - Rechner werden durch mathematische Objekte nachgebildet
  - zu lösende Aufgaben werden mengentheoretisch als Problem definiert
  - die Abfolge von Berechnungsschritten wird formalisiert
  - die quantitative Bestimmung der Komplexität eines Verfahrens bzw. eines Problems wird festgelegt
- Simulation
  - Verfahren zur Ersetzung eines Programms in einem Formalismus A durch ein Programm in einem anderen Formalismus B bei unverändertem Ein-/Ausgabeverhalten

2. Wesentliche Techniken und Konzepte

- Formalisierung
  - Rechner werden durch mathematische Objekte nachgebildet
  - zu lösende Aufgaben werden mengentheoretisch als Problem definiert
  - die Abfolge von Berechnungsschritten wird formalisiert
  - die quantitative Bestimmung der Komplexität eines Verfahrens bzw. eines Problems wird festgelegt
- Simulation
  - Verfahren zur Ersetzung eines Programms in einem Formalismus A durch ein Programm in einem anderen Formalismus B bei unverändertem Ein-/Ausgabeverhalten

2. Wesentliche Techniken und Konzepte

- Formalisierung
  - Rechner werden durch mathematische Objekte nachgebildet
  - zu lösende Aufgaben werden mengentheoretisch als Problem definiert
  - die Abfolge von Berechnungsschritten wird formalisiert
  - die quantitative Bestimmung der Komplexität eines Verfahrens bzw. eines Problems wird festgelegt
- Simulation
  - Verfahren zur Ersetzung eines Programms in einem Formalismus A durch ein Programm in einem anderen Formalismus B bei unverändertem Ein-/Ausgabeverhalten

2. Wesentliche Techniken und Konzepte

- Reduktion
  - formale Beschreibung für
    "Problem A ist nicht (wesentlich) schwerer als Problem B"
- Diagonalisierung
  - Auflistung aller Algorithmen einer bestimmten Klasse
  - Beweis durch Widerspruch
  - enger Bezug zur Paradoxie

2. Wesentliche Techniken und Konzepte

- Reduktion
  - formale Beschreibung für
    "Problem A ist nicht (wesentlich) schwerer als Problem B"
- Diagonalisierung
  - Auflistung aller Algorithmen einer bestimmten Klasse
  - Beweis durch Widerspruch
  - enger Bezug zur Paradoxie

2. Wesentliche Techniken und Konzepte

- Reduktion
  - formale Beschreibung für
    "Problem A ist nicht (wesentlich) schwerer als Problem B"
- Diagonalisierung
  - Auflistung aller Algorithmen einer bestimmten Klasse
  - Beweis durch Widerspruch
  - enger Bezug zu Paradoxa

2. Wesentliche Techniken und Konzepte

- Reduktion
  - formale Beschreibung für
    "Problem A ist nicht (wesentlich) schwerer als Problem B"
- Diagonalisierung
  - Auflistung aller Algorithmen einer bestimmten Klasse
  - Beweis durch Widerspruch
  - enger Bezug zu Paradoxa

2. Wesentliche Techniken und Konzepte

- Reduktion
  - formale Beschreibung für
    "Problem A ist nicht (wesentlich) schwerer als Problem B"
- Diagonalisierung
  - Auflistung aller Algorithmen einer bestimmten Klasse
  - Beweis durch Widerspruch
  - enger Bezug zu Paradoxa

2. Wesentliche Techniken und Konzepte

- Reduktion
  - formale Beschreibung für
    "Problem A ist nicht (wesentlich) schwerer als Problem B"
- Diagonalisierung
  - Auflistung aller Algorithmen einer bestimmten Klasse
  - Beweis durch Widerspruch
  - enger Bezug zu Paradoxa

3. Literatur

1. Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman: The design and analysis of computer algorithms. Addison-Wesley Publishing Company, Reading (MA), 1976
2. Manfred Broy: Informatik: eine grundlegende Einführung - Teil 4: Theoretische Informatik, Algorithmen und Datenstrukturen, Logikprogrammierung, Objektorientierung. Springer-Verlag, Berlin-Heidelberg-New York, 1996
3. Gerhard Goos: Vorlesungen über Informatik, Bd. 3, Berechenbarkeit, formale Sprachen, Spezifikationen. Springer-Verlag, Berlin-Heidelberg-New York, 1997
4. John E. Hopcroft, Jeffrey D. Ullman: Introduction to automata theory, languages, and computation. Addison-Wesley Publishing Company, Reading (MA), 1979

5. Uwe Schöning: Theoretische Informatik — kurzgefasst. Spektrum Akademischer Verlag GmbH, Heidelberg-Berlin, 1997
6. Karin Erk, Lutz Priese: Theoretische Informatik: Eine umfassende Einführung. Springer-Verlag, Berlin-Heidelberg-New York, 2000
7. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest: Introduction to algorithms. McGraw-Hill Book Company, New York-St. Louis-San Francisco-Montreal-Toronto, 1990
8. Thomas Ottmann, Peter Widmayer: Algorithmen und Datenstrukturen. B.I., Mannheim-Leipzig-Wien-Zürich, 1993
9. Volker Heun: Grundlegende Algorithmen. Vieweg, 2000
10. Ingo Wegener: Theoretische Informatik. B.G. Teubner, Stuttgart, 1993

Kapitel I Formale Sprachen und Automaten

1. Beispiele

Sei Σ ein (endliches) Alphabet. Dann

Definition 1

1. ist Σ^* das Monoid über Σ, d.h. die Menge aller endlichen Wörter über Σ;
2. ist Σ^+ die Menge aller nichtleeren endlichen Wörter über Σ^*;
3. bezeichnet $|w|$ für $w \in \Sigma^*$ die Länge von w;
4. ist Σ^n für $n \in \mathbb{N}_0$ die Menge aller Wörter der Länge n in Σ^*;
5. eine Teilmenge $L \subseteq \Sigma^*$ eine formale Sprache.

Kapitel I Formale Sprachen und Automaten

1. Beispiele

Sei Σ ein (endliches) Alphabet. Dann

Definition 1

1. ist Σ^* das Monoid über Σ, d.h. die Menge aller endlichen Wörter über Σ;
2. ist Σ^+ die Menge aller nichtleeren endlichen Wörter über Σ^*;
3. bezeichnet $|w|$ für $w \in \Sigma^*$ die Länge von w;
4. ist Σ^n für $n \in \mathbb{N}_0$ die Menge aller Wörter der Länge n in Σ^*;
5. eine Teilmenge $L \subseteq \Sigma^*$ eine formale Sprache.

Kapitel I Formale Sprachen und Automaten

1. Beispiele

Sei Σ ein (endliches) Alphabet. Dann

Definition 1

1. ist Σ^* das Monoid über Σ, d.h. die Menge aller endlichen Wörter über Σ;
2. ist Σ^+ die Menge aller nichtleeren endlichen Wörter über Σ^*;
3. bezeichnet $|w|$ für $w \in \Sigma^*$ die Länge von w;
4. ist Σ^n für $n \in \mathbb{N}_0$ die Menge aller Wörter der Länge n in Σ^*;
5. eine Teilmenge $L \subseteq \Sigma^*$ eine formale Sprache.

Kapitel I Formale Sprachen und Automaten

1. Beispiele

Sei Σ ein (endliches) Alphabet. Dann

Definition 1

1. ist Σ^* das Monoid über Σ, d.h. die Menge aller endlichen Wörter über Σ;
2. ist Σ^+ die Menge aller nichtleeren endlichen Wörter über Σ^*;
3. bezeichnet $|w|$ für $w \in \Sigma^*$ die Länge von w;
4. ist Σ^n für $n \in \mathbb{N}_0$ die Menge aller Wörter der Länge n in Σ^*;
5. eine Teilmenge $L \subseteq \Sigma^*$ eine formale Sprache.

Kapitel I Formale Sprachen und Automaten

1. Beispiele

Sei Σ ein (endliches) Alphabet. Dann

Definition 1

1. ist Σ^* das Monoid über Σ, d.h. die Menge aller endlichen Wörter über Σ;
2. ist Σ^+ die Menge aller nichtleeren endlichen Wörter über Σ^*;
3. bezeichnet $|w|$ für $w \in \Sigma^*$ die Länge von w;
4. ist Σ^n für $n \in \mathbb{N}_0$ die Menge aller Wörter der Länge n in Σ^*;
5. eine Teilmenge $L \subseteq \Sigma^*$ eine formale Sprache.

Beispiel 2

Wir betrachten folgende Grammatik:

$$
\begin{aligned}
\langle \text{Satz} \rangle &\rightarrow \langle \text{Subjekt} \rangle \langle \text{Prädikat} \rangle \langle \text{Objekt} \rangle \\
\langle \text{Subjekt} \rangle &\rightarrow \langle \text{Artikel} \rangle \langle \text{Attribut} \rangle \langle \text{Substantiv} \rangle \\
\langle \text{Artikel} \rangle &\rightarrow \epsilon \\
\langle \text{Artikel} \rangle &\rightarrow \text{der} | \text{die} | \text{das} | \text{ein} | \ldots \\
\langle \text{Attribut} \rangle &\rightarrow \epsilon | \langle \text{Adjektiv} \rangle | \langle \text{Adjektiv} \rangle \langle \text{Attribut} \rangle \\
\langle \text{Adjektiv} \rangle &\rightarrow \text{gross} | \text{klein} | \text{schön} | \ldots
\end{aligned}
$$

Die vorletzte Ersetzungsregel ist rekursiv, die durch diese Grammatik definierte Sprache deshalb unendlich.

Beispiel 2

Wir betrachten folgende Grammatik:

$$
\begin{aligned}
\langle \text{Satz} \rangle &\rightarrow \langle \text{Subjekt} \rangle \langle \text{Prädikat} \rangle \langle \text{Objekt} \rangle \\
\langle \text{Subjekt} \rangle &\rightarrow \langle \text{Artikel} \rangle \langle \text{Attribut} \rangle \langle \text{Substantiv} \rangle \\
\langle \text{Artikel} \rangle &\rightarrow \epsilon \\
\langle \text{Artikel} \rangle &\rightarrow \text{der}|\text{die}|\text{das}|\text{ein}|\ldots \\
\langle \text{Attribut} \rangle &\rightarrow \epsilon|\langle \text{Adjektiv} \rangle|\langle \text{Adjektiv} \rangle \langle \text{Attribut} \rangle \\
\langle \text{Adjektiv} \rangle &\rightarrow \text{gross}|\text{klein}|\text{schön}|\ldots
\end{aligned}
$$

Die vorletzte Ersetzungsregel ist rekursiv, die durch diese Grammatik definierte Sprache deshalb unendlich.

Beispiel 3

- $L_1 = \{aa, aaaa, aaaaaa, \ldots\} = \{(aa)^n, n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a\})$
- $L_2 = \{ab, abab, ababab, \ldots\} = \{(ab)^n, n \in \mathbb{N}\}$
  $(\Sigma_2 = \{a, b\})$
- $L_3 = \{ab, aabb, aaabbb, \ldots\} = \{a^n b^n, n \in \mathbb{N}\}$
  $(\Sigma_3 = \{a, b\})$
- $L_4 = \{a, b, aa, ab, bb, aaa, aab, abb, bbb \ldots\}$
  $= \{a^m b^n, m, n \in \mathbb{N}_0, m + n > 0\}$ $(\Sigma_4 = \{a, b\})$

Beispiel 3

- $L_1 = \{aa, aaaa, aaaaaa, \ldots\} = \{(aa)^n, n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a\})$
- $L_2 = \{ab, abab, ababab, \ldots\} = \{(ab)^n, n \in \mathbb{N}\}$
  $(\Sigma_2 = \{a, b\})$
- $L_3 = \{ab, aabb, aaabbb, \ldots\} = \{a^n b^n, n \in \mathbb{N}\}$
  $(\Sigma_3 = \{a, b\})$
- $L_4 = \{a, b, aa, ab, bb, aaa, aab, abb, bbb \ldots\}$
  $= \{a^m b^n, m, n \in \mathbb{N}_0, m + n > 0\}$ $(\Sigma_4 = \{a, b\})$

Beispiel 3

- $L_1 = \{aa, aaaa, aaaaaa, \ldots\} = \{(aa)^n, n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a\})$
- $L_2 = \{ab, abab, ababab, \ldots\} = \{(ab)^n, n \in \mathbb{N}\}$
  $(\Sigma_2 = \{a, b\})$
- $L_3 = \{ab, aabb, aaabbb, \ldots\} = \{a^n b^n, n \in \mathbb{N}\}$
  $(\Sigma_3 = \{a, b\})$
- $L_4 = \{a, b, aa, ab, bb, aaa, aab, abb, bbb \ldots\}$
  $= \{a^m b^n, m, n \in \mathbb{N}_0, m + n > 0\}$ $(\Sigma_4 = \{a, b\})$

Beispiel 3

- $L_1 = \{aa, aaaa, aaaaaa, \ldots\} = \{(aa)^n, n \in \mathbb{N}\}$
  $(\Sigma_1 = \{a\})$
- $L_2 = \{ab, abab, ababab, \ldots\} = \{(ab)^n, n \in \mathbb{N}\}$
  $(\Sigma_2 = \{a, b\})$
- $L_3 = \{ab, aabb, aaabbb, \ldots\} = \{a^n b^n, n \in \mathbb{N}\}$
  $(\Sigma_3 = \{a, b\})$
- $L_4 = \{a, b, aa, ab, bb, aaa, aab, abb, bbb \ldots\}$
  $= \{a^m b^n, m, n \in \mathbb{N}_0, m + n > 0\} \qquad (\Sigma_4 = \{a, b\})$

2. Die Chomsky-Hierarchie

Diese Sprachenhierarchie ist nach Noam Chomsky [MIT, 1976] benannt.

2.1 Phrasenstrukturgrammatik, Chomsky-Grammatik

Grammatiken bestehen aus

1. einem Terminalalphabet Σ (manchmal auch T), $|\Sigma| < \infty$
2. einem endlichen Vorrat von Nichtterminalzeichen (Variablen) V, $V \cap \Sigma = \emptyset$
3. einem Startsymbol (Axiom) $s \in V$
4. einer endlichen Menge P von Produktionen (Ableitungsregeln) der Form $\ell \to r$ mit $\ell \in (V \cup \Sigma)^+ \wedge r \in (V \cup \Sigma)^*$

2. Die Chomsky-Hierarchie

Diese Sprachenhierarchie ist nach Noam Chomsky [MIT, 1976] benannt.

2.1 Phrasenstrukturgrammatik, Chomsky-Grammatik

Grammatiken bestehen aus

1. einem Terminalalphabet Σ (manchmal auch T), $|\Sigma| < \infty$
2. einem endlichen Vorrat von Nichtterminalzeichen (Variablen) V, $V \cap \Sigma = \emptyset$
3. einem Startsymbol (Axiom) $s \in V$
4. einer endliche Menge P von Produktionen (Ableitungsregeln) der Form $l \to r$, mit $l \in (V \cup \Sigma)^+$, $r \in (V \cup \Sigma)^*$

2. Die Chomsky-Hierarchie

Diese Sprachenhierarchie ist nach Noam Chomsky [MIT, 1976] benannt.

2.1 Phrasenstrukturgrammatik, Chomsky-Grammatik

Grammatiken bestehen aus

1. einem Terminalalphabet Σ (manchmal auch T), $|\Sigma| < \infty$
2. einem endlichen Vorrat von Nichtterminalzeichen (Variablen) V, $V \cap \Sigma = \emptyset$
3. einem Startsymbol (Axiom) $s \in V$
4. einer endliche Menge P von Produktionen (Ableitungsregeln) der Form $l \to r$, mit $l \in (V \cup \Sigma)^+$, $r \in (V \cup \Sigma)^*$

2. Die Chomsky-Hierarchie

Diese Sprachenhierarchie ist nach Noam Chomsky [MIT, 1976] benannt.

2.1 Phrasenstrukturgrammatik, Chomsky-Grammatik

Grammatiken bestehen aus

1. einem Terminalalphabet Σ (manchmal auch T), $|\Sigma| < \infty$
2. einem endlichen Vorrat von Nichtterminalzeichen (Variablen) V, $V \cap \Sigma = \emptyset$
3. einem Startsymbol (Axiom) $s \in V$
4. einer endliche Menge P von Produktionen (Ableitungsregeln) der Form $l \rightarrow r$, mit $l \in (V \cup \Sigma)^+$, $r \in (V \cup \Sigma)^*$

2. Die Chomsky-Hierarchie

Diese Sprachenhierarchie ist nach Noam Chomsky [MIT, 1976] benannt.

2.1 Phrasenstrukturgrammatik, Chomsky-Grammatik

Grammatiken bestehen aus

1. einem Terminalalphabet Σ (manchmal auch T), $|\Sigma| < \infty$
2. einem endlichen Vorrat von Nichtterminalzeichen (Variablen) V, $V \cap \Sigma = \emptyset$
3. einem Startsymbol (Axiom) $s \in V$
4. einer endliche Menge P von Produktionen (Ableitungsregeln) der Form $l \rightarrow r$, mit $l \in (V \cup \Sigma)^+$, $r \in (V \cup \Sigma)^*$

2. Die Chomsky-Hierarchie

Diese Sprachenhierarchie ist nach Noam Chomsky [MIT, 1976] benannt.

2.1 Phrasenstrukturgrammatik, Chomsky-Grammatik

Grammatiken bestehen aus

1. einem Terminalalphabet Σ (manchmal auch T), $|\Sigma| < \infty$
2. einem endlichen Vorrat von Nichtterminalzeichen (Variablen) V, $V \cap \Sigma = \emptyset$
3. einem Startsymbol (Axiom) $s \in V$
4. einer endliche Menge P von Produktionen (Ableitungsregeln) der Form $l \rightarrow r$, mit $l \in (V \cup \Sigma)^+$, $r \in (V \cup \Sigma)^*$

Eine Phrasenstrukturgrammatik (Grammatik) ist ein Quadrupel $G = (V, \Sigma, P, S)$.