Informatik I WS 07/08

Tutorium 24

20.12.07

© www.OhMyGoodness.com

Bastian Molkenthin

E-Mail: infotut@sunshine2k.de

Web: http://infotut.sunshine2k.de

Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

Rückblick Semi-Thue-Systeme

- Ein Semi-Thue-System besteht aus zwei Mengen:
  - Zeichenvorrat Σ
  - Menge von Regeln T

- Die Reihenfolge der Regeln oder die Position der zu ersetzenden Zeichen im Wort spielt keine Rolle – es kann jede Regel zu jeder Zeit angewendet werden **(!)**

```
Solange anwendbare Regel vorhanden
{
        wähle beliebige anwendbare Regel
        wähle beliebige Anwendungsstelle
        wende Regel an
}
```

Beispielaufgaben

- Schreiben Sie ein Semi Thue System, das in einem Wort, deutsche Buchstaben durch ihre griechischen ersetzt. Das Alphabet sei hierbei:

$$\sum = \{a, b, c, \alpha, \beta, \gamma\}$$

Lösung:

$$a \longrightarrow \alpha$$
$$b \longrightarrow \beta$$
$$c \longrightarrow \gamma$$

- Schreiben Sie ein Semi Thue System, das in einem Wort, deutsche Buchstaben durch ihre griechischen ersetzt *und andersrum*. Das Alphabet sei hierbei:

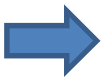
$$\sum = \{a, b, c, \alpha, \beta, \gamma\}$$

Lösung: Nicht möglich, würde nicht terminieren!

Markov - Algorithmen

- Ähnlich den Semi-Thue-Systemen, wieder zwei Grundmengen:
  - Zeichenvorrat Σ
  - Menge von Regeln T

- Größter Unterschied ist die definierte Reihenfolge bei der Regelanwendung:
  - Wende stets die erste mögliche Regel an (Nummerieren!)
  - Wende sie so weit links wie möglich an
  - Nach eine Halteregel (.) beende den Algorithmus
  - Wenn keine Regel mehr anwendbar ist, beende den Algorithmus

 Markov-System ist deterministisch!

Markov - Algorithmen (2)

Ein Textersetunzgssystem heißt **deterministisch** , wenn

wenn zu jeder Zeit immer nur eine Regel auf genau ein Teilwort anwendbar ist. Nichtdeterministisch bedeutet, dass für eine Eingabe das Verhalten des Systems unterschiedliche sein kann.

Ein Textersetunzgssystem heißt **determiniert** , wenn

dieselbe Eingabe immer dieselbe Ausgabe ergibt. Nichtdeterminiert bedeutet, dass für eine Eingabe die Ausgabe des Systems unterschiedlich sein kann.

- Wichtiger "Trick" bei Markov-Algorithmen:
  - Oft ist es nützlich, die Bearbeitungsposition zu speichern

➡ Verwendung sogenannter Schiffchen

Verwende hierzu bevorzugt grieschische Buchstaben α, β, γ, δ, ...

Markov Aufgabe

Entwickle einen Markov-Algorithmus, welcher die Länge beliebiger Wörter bestehend aus Strichen (|) verdoppelt!

➡ Hier müssen Schiffchen verwendet werden

Lösung: $\Sigma = \{ |, \alpha \}$

$T = \{$ (1) $\alpha| \rightarrow ||\alpha,$

(2) $\alpha \rightarrow .\varepsilon,$

(3) $\varepsilon \rightarrow \alpha$ $\}$

- Das Schiffchen wird durch die letzte Regel erzeugt, welche zu Beginn die einzig Anwendbare ist!

Markov Aufgabe (2)

Wende den Markov-Algorithmus auf das Wort |||| an.
Gib alle Schritte mit Hinweis auf die verwendete Regel an!

$\Sigma = \{\ |, \alpha\ \}$

$T = \{ \quad (1) \quad \alpha| \rightarrow ||\alpha,$

$\qquad\quad (2) \quad \alpha \rightarrow .\varepsilon,$

$\qquad\quad (3) \quad \varepsilon \rightarrow \alpha \quad \}$

Lösung:

$$|||| \Rightarrow^3 \alpha|||| \Rightarrow^1 ||\alpha||| \Rightarrow^1 ||||\alpha|| \Rightarrow^1 ||||||\alpha| \Rightarrow^1 ||||||||\alpha \Rightarrow^2 ||||||||$$

Markov – Aufgabe 2

Entwickle einen Markov-Algorithmus, welcher sämtliche Kleinbuchstaben in einem Wort durch Großbuchstaben und sämtliche Großbuchstaben durch Kleinbuchstaben ersetzt!

Zeichenvorrat sei gegeben mit $\Sigma = \{a, b, c, A, B, C, \alpha\}$

Lösung:

$T = \{$

(1) $\alpha a \rightarrow A\alpha,$
(2) $\alpha b \rightarrow B\alpha,$
(3) $\alpha c \rightarrow C\alpha,$
(4) $\alpha A \rightarrow a\alpha,$
(5) $\alpha B \rightarrow b\alpha,$
(6) $\alpha C \rightarrow c\alpha,$
(7) $\alpha \rightarrow .\varepsilon,$
(8) $\varepsilon \rightarrow \alpha$ $\}$

Chomsky - Grammatiken

- (Chomsky-) Grammatiken besitzen einen ähnlichen Aufbau wie schon die Semi-Thue-Systeme und Markov-Algorithmen:
  - Der Zeichenvorrat wird unterteilt in die Mengen der Terminalsymbole (Σ) und der Nichtterminalsymbole (**N**).
  - Regeln nennen wir Produktionen und sind in der Menge **P**.
  - Alle Ableitungen beginnen bei dem Startsymbol (ein *Nicht-Terminal!*) **A** (auch Axiom genannt)
- Der Viertupel aus Σ, N, P und Axiom bildet die Grammatik:

$$\mathbf{G = (\Sigma, N, P, A)}$$

- Sämtliche Worte aus Σ^*, welche sich mit den gegebenen Regeln aus A ableiten lassen, bilden die von der Grammatik erzeugte Sprache:

$$\mathbf{L(G) = \{w \in \Sigma^* \mid A \Rightarrow^* w\}}$$

Chomsky – Grammatiken (2)

- Bereits gesehen bei Semi-Thue-Systemen und Markov-Algorithmen:
  - Die verschiedenen Systeme sind unterschiedlich mächtig
    (es gibt Probleme, welche sich mit Markov-Algorithmen lösen lassen,
    jedoch nicht mit Semi-Thue-Systemen)

- In der **Chomsky-Hierarchie** werden die verschiedenen Typen von Grammatiken in 4 Klassen eingeteilt, wobei hier eine "Rangfolge" bezüglich der Mächtigkeit vorliegt

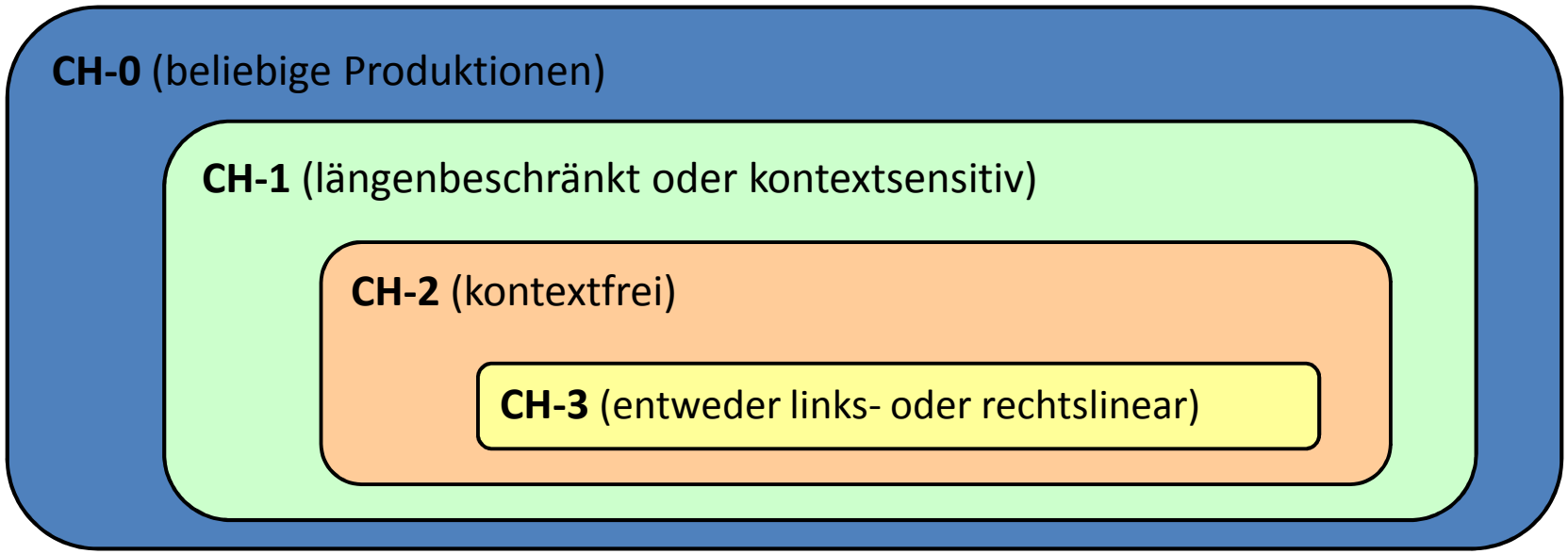
  ⇒ Die Unterschiede liegen hierbei in der Form der erlaubten Produktionen

Chomsky - Hierarchie

- **Chomsky-0-Grammatik (CH-0)**
  - allgemeiner Produktionstyp: $l \rightarrow r$ wobei $l, r \in V^*$ beliebig
  - insbesondere auch ε-Produktionen ($r = \varepsilon$)
- **Chomsky-1-Grammatik (CH-1)**
  - längenbeschränkt: $l \rightarrow r$ wobei $l, r \in V^*, 1 \leq |l| \leq |r|$
  - kontextsensitiv: $uAv \rightarrow urv$ wobei $A \in N, u, v \in V^*$, $r \in V+$, d.h. $r \neq \varepsilon$
- **Chomsky-2-Grammatik (CH-2)**
  - kontextfrei: $A \rightarrow r$ wobei $A \in N, r \in V^*$
- **Chomsky-3-Grammatik (CH-3)**
  - entweder linkslinear: $A \rightarrow Bx$ oder $A \rightarrow x$ wobei $A, B \in N, x \in \Sigma$
  - oder rechtslinear: $A \rightarrow xB$ oder $A \rightarrow x$ wobei $A, B \in N, x \in \Sigma$

Chomsky - Hierarchie

- Chomsky-Grammatiken kleinerer Nummer enthalten Grammatiken höherer Nummer
- Größere Nummern enthalten die kleineren **NICHT**
- Je kleiner die Nummer desto mächtiger
- Chomsky-0 ist so gut wie Semi Thue oder Markov
- Chomsky-2 Grammatiken sind äquivalent zur EBNF
- Unter dem Chomsky-Typ einer **Sprache** versteht man den höchsten Grammatik-Typ, welcher diese Sprache noch erzeugen kann



Chomsky-Hierarchie (Aufgabe 1)

Gib zu der folgenden Grammatik jeweils mit Begründung den (einschränkendsten) Typ und die erzeugte Sprache an!

- $G_1 = (\Sigma, N, P, A)$
- $\Sigma = \{a, b\}$
- $N = \{A, B\}$
- $P = \{ \quad A \rightarrow aA \mid aB,$
  $B \rightarrow bB \mid b \quad \}$

Typ: CH-3, da alle Produktionen rechtslinear sind

Sprache: $L(G_1) = \{ a^n b^m \mid n, m \in \mathbb{N}\backslash\{0\} \}$

Chomsky-Hierarchie (Aufgabe 2)

Gib zu der folgenden Grammatik jeweils mit Begründung den (einschränkendsten) Typ und die erzeugte Sprache an!

- $G_2 = (\Sigma, N, P, S)$
- $\Sigma = \{a, b\}$
- $N = \{A\}$
- $P = \{ \quad A \rightarrow aAb \mid ab \}$

Typ: CH-2, da z.B. $A \rightarrow ab$ kontextfrei ist, aber keine CH-1 oder CH-0-Produktionen enthalten sind (linke Seite nur Nichtterminalzeichen)

Sprache: $L(G_2) = \{ a^n b^n \mid n \in \mathbb{N} \setminus \{0\} \}$

Chomsky-Hierarchie (Aufgabe 3)

Gib zu der folgenden Grammatik jeweils mit Begründung den (einschränkendsten) Typ und die erzeugte Sprache an!

- $G_3 = (\Sigma, N, P, A)$
- $\Sigma = \{a, b, c\}$
- $N = \{A, B, C, D\}$
- $P = \{$ $A \rightarrow aA \mid aB,$
  $aB \rightarrow abC \mid aD,$
  $C \rightarrow bC \mid bD,$
  $D \rightarrow cD \mid c$ $\}$

(*) CH-3 Produktionen:

$P' = \{$ $A \rightarrow aA \mid aB,$
$B \rightarrow bC \mid cD \mid c,$
$C \rightarrow bC \mid bD,$
$D \rightarrow cD \mid c \}$

Typ: CH-1, da z.B. $aB \rightarrow abC$ nur längenbeschränkt ist, aber auch keine CH-0-Produktionen enthalten sind.

Sprache: Sprache: $L(G_3) = \{ a^i b^j c^k \mid i, j, k \in \mathbb{N} \text{ und } i, k \neq 0, j \neq 1 \}$

⇒ Die Sprache ist vom Typ CH-3, da eine CH-3-Grammatik existiert, welche die gleiche Sprache erzeugt (*)

Chomsky Aufgaben

Gegeben sei die Grammtik G = (Σ, N, P, A) mit Σ = {a, b, c} und N = {A, B, C}

Gebe jeweils den max. Chomsky-Typ (*mit Begründung!*) an!

P = {

A → AbbA | a,
b → AbbACab,
B → b,
C → aBc

Die zweite Produktion enthält eine Produktion aus einem Terminalsymbol heraus. Damit kann sie nicht CH-2 oder CH-3 sein. Da die Produktionen längenbeschränkt sind, ist sie CH-1.

P = {

A → AbbA,
B → b,
C → aBc

}

Alle Produktionen sind kontextfrei, also liegt eine CH-2-Grammatik vor. Die erste und dritte Produktion sind weder rechts noch links-linear und daher in CH-3 nicht erlaubt.

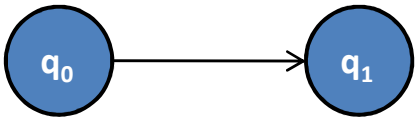
Endliche Automaten

Bestandteile:

- *endliche Menge Q von Zuständen* mit einem Anfangszustand $q_0 \in Q$
- *Zeichenvorrat Σ*
- *Zustandsübergänge*: Lesen eines Zeichens $a \in \Sigma$ führt zu einem Zustandsübergang vom aktuellen Zustand $q \in Q$ in einen neuen Zustand $q' \in Q$

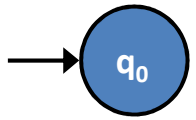
Ein endlicher Automat kann als Graph aufgefasst werden:

- Zustände $Q = \{q_0, q_1, ..., q_n\}$ des endlichen Automaten lassen sich als Ecken eines Graphen auffassen
- Zustandsübergänge $q_i a \rightarrow q_j$ mit $a \in \Sigma$ entsprechen markierte gerichtete Kanten



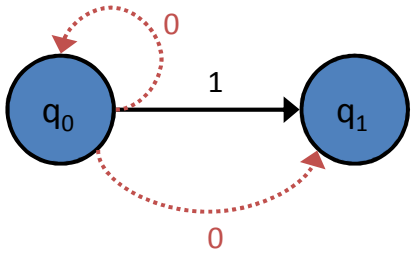
Endliche Automaten (2)

- Was ist der Startzustand?
  - Kennzeichnet die Anfangsposition im Automaten
  - Wird durch einen Pfeil gekennzeichnet (wichtig!)

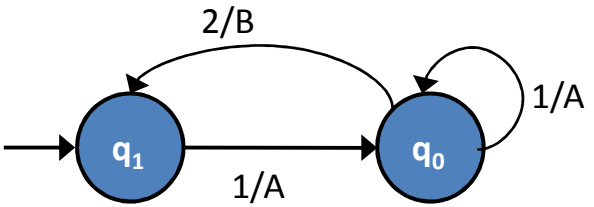


- Was bedeutet vollständig?
  - An jedem Zustand ist für jedes Zeichen aus dem Eingabealphabet ein Übergang gegeben (Nötigenfalls zu einem speziellen Fehlerzustand)

- Was bedeutet deterministisch?
  - Der Automat muss für gleiche Eingaben immer das gleiche Ergebnis liefern, also eindeutig sein.
  - Keine "Wahlmöglichkeit" bei Verzweigungen:



Hinweis SchniPoStar-Aufgabe




```
public class State {
          static char input;
          public static void stateOne() {
              Out.println("Zustand 1:");
              do {
                  Out.print("Eingabe: ");
                  input = In.readChar();
                  if (input == '1') {
                      Out.println("Ausgabe = A"); stateTwo();
                  }
              } while (input != '1');
          }

          public static void stateTwo() {
              Out.println("Zustand 2:");
              do {
                  Out.print("Eingabe: ");
                  input = In.readChar();
                  if (input == '1') {
                      Out.println("Ausgabe = A"); stateTwo();
                  }
                  else if (input == '2') {
                      Out.println("Ausgabe = B"); stateOne();
                  }
              } while (input != '1' && input != '2');
          }

          public static void main(String[] args) {
              stateOne();
          }
}
```

Automat muss im Javacode ersichtlich sein!

BigInteger

BigInteger ist eine Klasse zum Rechnen mit beliebig großen Zahlen.

Erzeugen:

BigInteger(String val) Erzeugt ein BigInteger aus einem Ziffern-String

```
BigInteger a = new BigInteger("23");
```

static BigInteger valueOf(long val) Konstruiert aus einem long ein BigInteger

```
BigInteger a = BigInteger.valueOf(23);
```

Wichtige Funktionen:

BigInteger add(BigInteger val), BigInteger and(BigInteger val),

BigInteger multiply(BigInteger val), BigInteger or(BigInteger val),

BigInteger modPow(BigInteger exponent, BigInteger m)

Übersicht: http://java.sun.com/javase/6/docs/api/java/math/BigInteger.html

Aufgabe

Schreibe ein Java Programm, welches den größten gemeinsamen Teiler zweier Zahlen berechnet. Verwende hierzu die BigInteger Klasse!

```
import java.math.BigInteger;

public class GCD {

    static BigInteger gcd(BigInteger a, BigInteger b) {
        BigInteger c;
        while (!b.equals(BigInteger.ZERO)) {
            c=a.mod(b); a=b; b=c;
        }
        return a;
    }

    public static void main(String[] args) {
        BigInteger result = gcd(new BigInteger("1002001002492"),
                                new BigInteger("12339432"));
        Out.println("GCD is : " + result);
    }
}
```

Kleine Zusatzaufgabe

Wie berechnet sich die Kantenzahl aus der Eckenzahl für einen vollständigen gerichteten Graphen mit reflexiven Kanten?

Annahme: Kantenzahl = (Eckenzahl)², also $k(x) = x^2$

(Bezeichne x die Eckenanzahl und k(x) die Kantenanzahl abh. von x)

Induktionsanfang: $x = 1$: $k(1) = 1^2 = 1$

Induktionsschritt: $x \Rightarrow x+1$:

$k(x+1) = k(x) + 1 + 2x = x^2 + 2x + 1 = (x + 1)^2$

Überlegung des Kanten-Zuwachses bei Hinzunahme eines weiteren Knotens:

+ 1 Kante, die auf den Knoten verweist (reflexive Kante)
+ je 2 Kanten auf die bestehenden x Knoten

$\Rightarrow 1 + 2x$

Fragen ???

Frohe Weihnachten & einen guten Rutsch ins Neue!