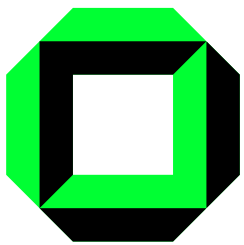
Info I – Übungsblatt 8

Joachim Breitner

mit Aufgaben von Martin Kiefel

http://www.joachim-breitner.de/wiki/Infotut

9. Januar 2006



Unser Programm heute

1. Übungsblatt 7
2. Zufallszahlen in Java
3. Semi-Thue-Systeme
4. Markov-Algorithmus
5. Grammatiken
6. Endliche Automaten

Übungsblatt-Rückblick

Statistik

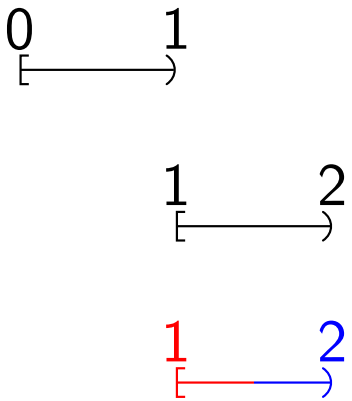
- Wenige Abgaben
- Schnitt: 28 von 30 Punkten

Der naive Ansatz

Folgender Code soll zufällig 1 oder 2 zurückgeben:

```
(int)(Math.round(Math.random() + 1));
```

Veranschaulichung:



`Math.random()`

`+1`

`Math.round`

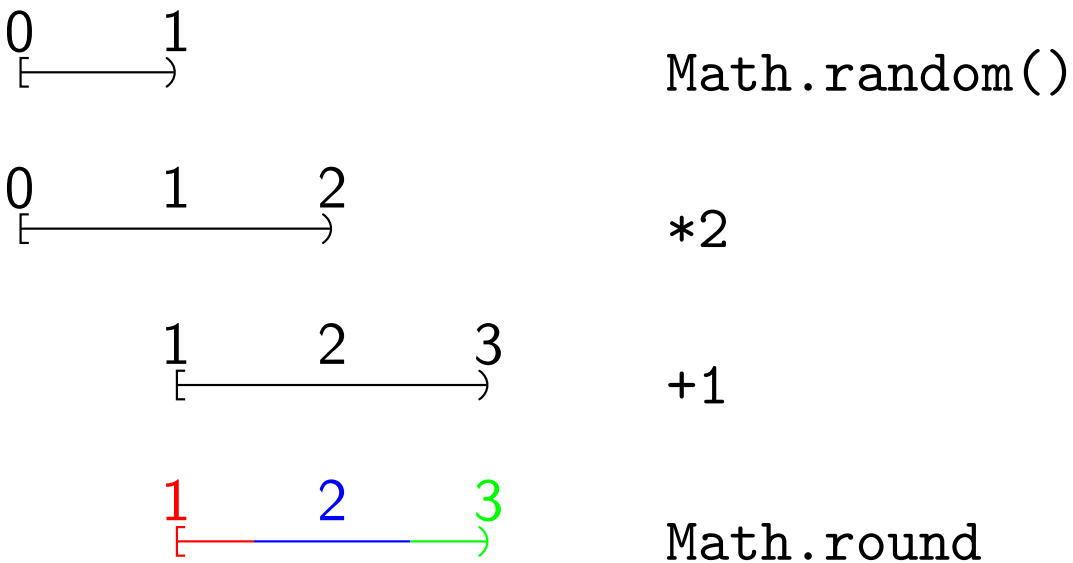
Soweit so gut.

Der naive Ansatz (2)

Folgender Code soll zufällig 1,2 oder 3 zurückgeben:

```
(int)(Math.round(Math.random()*2 + 1));
```

Veranschaulichung:



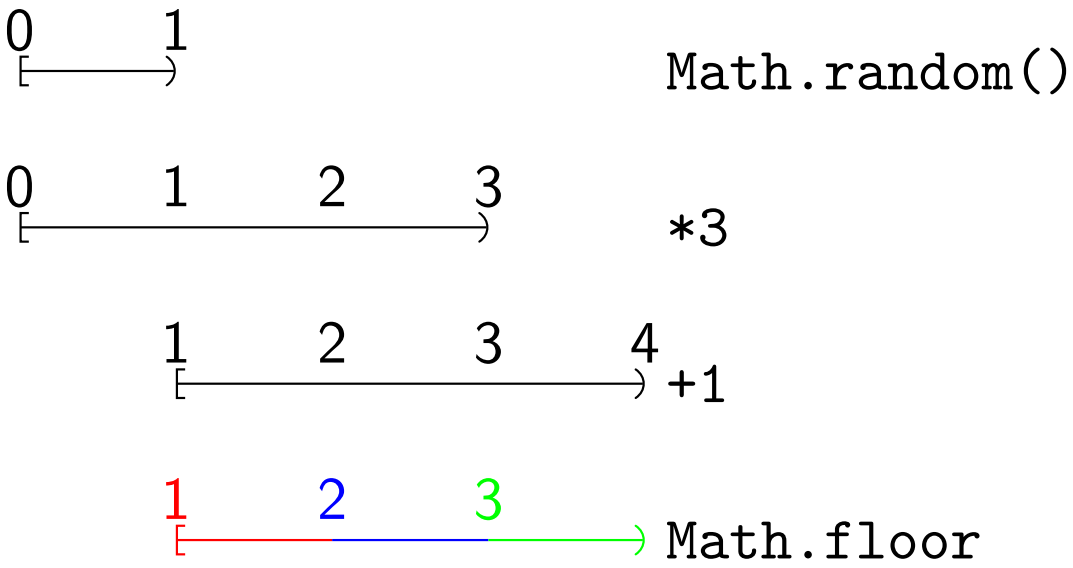
Die 2 ist dopppelt so wahrscheinlich wie die 1 oder 3!

Der bessere Ansatz

Folgender Code soll zufällig 1,2 oder 3 zurückgeben:

```
(int)(Math.floor(Math.random()*3 + 1));
```

Veranschaulichung:



Schon besser.

sowas wie Feng-Shui

Semi-Thue-Systeme

sind Textersetzungssysteme, d.h., Regeln geben vor, welchen Textteil man durch welchen anderen Text ersetzen darf.
Die Reihenfolge der Regeln sind beliebig, es gibt also nicht immer ein eindeutiges Ergebnis, und es muss nicht immer eines geben.

Wer war Semi, und was thut der?

Semi-Thue-Systeme sind Verallgemeinerungen der Thue-Systeme, benannt nach Axel Thue. Bei diesen gilt jede Ersetzungsregel stets in beide Richtungen. Bei uns gilt jedoch nur eine, also „halb so viele“ Regeln, demnach **Semi**-Thue-System.
Motivation für Thue war eine logische Untersuchung der Mathematik (vergleiche die Umformungsregeln einer Algebra).

sowas wie Feng-Shui

Semi-Thue-Systeme

sind Textersetzungssysteme, d.h., Regeln geben vor, welchen Textteil man durch welchen anderen Text ersetzen darf.
Die Reihenfolge der Regeln sind beliebig, es gibt also nicht immer ein eindeutiges Ergebnis, und es muss nicht immer eines geben.

Wer war Semi, und was thut der?

Semi-Thue-Systeme sind Verallgemeinerungen der Thue-Systeme, benannt nach Axel Thue. Bei diesen gilt jede Ersetzungsregel stets in beide Richtungen. Bei uns gilt jedoch nur eine, also „halb so viele“ Regeln, demnach **Semi**-Thue-System.
Motivation für Thue war eine logische Untersuchung der Mathematik (vergleiche die Umformungsregeln einer Algebra).

Beispielsystem

Aufgabe:

Schreiben sie ein Semi-Thue-System, das in einem Wort (über $\Sigma = \{a, b, c, \alpha, \beta, \gamma\}$) die griechischen Buchstaben durch ihre lateinischen Äquivalente ersetzt.

Beispielsystem

Aufgabe:

Schreiben sie ein Semi-Thue-System, das in einem Wort (über $\Sigma = \{a, b, c, \alpha, \beta, \gamma\}$) die griechischen Buchstaben durch ihre lateinischen Äquivalente ersetzt.

Lösung

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

Beispielsystem

Aufgabe:

Schreiben sie ein Semi-Thue-System, das in einem Wort (über $\Sigma = \{a, b, c, \alpha, \beta, \gamma\}$) die griechischen Buchstaben durch ihre lateinischen Äquivalente ersetzt und andersherum.

Beispielsystem

Aufgabe:

Schreiben sie ein Semi-Thue-System, das in einem Wort (über $\Sigma = \{a, b, c, \alpha, \beta, \gamma\}$) die griechischen Buchstaben durch ihre lateinischen Äquivalente ersetzt und andersherum.

Lösung

Diese Aufgabe ist mit Semi-Thue-Systemen nicht lösbar, da auch im gewünschten Endzustand Regeln anwendbar sein müssten.

Beispielsystem

Aufgabe:

Schreiben sie ein Semi-Thue-System, das in einem Wort (über $\Sigma = \{a, b, c, \alpha, \beta, \gamma, \aleph\}$) die griechischen Buchstaben durch ihre lateinischen Äquivalente ersetzt und andersherum. Am Anfang des Wortes steht ein $\aleph$, das im Ergebnis irgendwo stehen darf.

Beispielsystem

Aufgabe:

Schreiben sie ein Semi-Thue-System, das in einem Wort (über $\Sigma = \{a, b, c, \alpha, \beta, \gamma, \aleph\}$) die griechischen Buchstaben durch ihre lateinischen Äquivalente ersetzt und andersherum. Am Anfang des Wortes steht ein $\aleph$, das im Ergebnis irgendwo stehen darf.

Lösung

$$\aleph a \to \alpha\aleph \qquad \aleph\alpha \to a\aleph$$
$$\aleph b \to \beta\aleph \qquad \aleph\beta \to b\aleph$$
$$\aleph c \to \gamma\aleph \qquad \aleph\gamma \to c\aleph$$

Auf hoher See

Wir haben gesehen, dass Semi-Thue-Syteme schlecht sind, wenn wir die Regeln „kontrollierter“ anwenden wollen. Daher einigen wir uns bei den Markov-Algorithmen auf folgende Meta-Regeln:

1. Wende stets die erste mögliche Regel an.
2. Wende sie so weit links wie möglich an.
3. Nach einer Halteregel ($\rightarrow\bullet$) beende den Algorithmus
4. Wenn keine Regel anwendbar ist, beende den Algorithmus

Schiffchen

Oft muss man bei Markov-Algorithmen wissen, wo man gerade ist. Dazu führt man Zeichen ein, die am Anfang noch nicht in der Eingabe waren und hinterher wieder gelöscht wurden.
Diese heißen Schiffchen, und sind üblicherweise griechische Buchstaben (α, β, γ,...).

Auf hoher See

Wir haben gesehen, dass Semi-Thue-Syteme schlecht sind, wenn wir die Regeln „kontrollierter“ anwenden wollen. Daher einigen wir uns bei den Markov-Algorithmen auf folgende Meta-Regeln:

1. Wende stets die erste mögliche Regel an.
2. Wende sie so weit links wie möglich an.
3. Nach einer Halteregel ($\rightarrow\bullet$) beende den Algorithmus
4. Wenn keine Regel anwendbar ist, beende den Algorithmus

Schiffchen

Oft muss man bei Markov-Algorithmen wissen, wo man gerade ist. Dazu führt man Zeichen ein, die am Anfang noch nicht in der Eingabe waren und hinterher wieder gelöscht wurden.
Diese heißen Schiffchen, und sind üblicherweise griechische Buchstaben (α, β, γ,...).

Beispiel

Aufgabe

Schreiben sie einen Markov-Algorithmus, das in einem Wort (über $\Sigma = \{a, b, c, \alpha, \beta, \gamma\}$) die griechischen Buchstaben durch ihre lateinischen Äquivalente ersetzt und andersherum.

Beispiel

Aufgabe

Schreiben sie einen Markov-Algorithmus, das in einem Wort (über $\Sigma = \{a, b, c, \alpha, \beta, \gamma\}$) die griechischen Buchstaben durch ihre lateinischen Äquivalente ersetzt und andersherum.

Lösung

1. $\aleph a \to \alpha\aleph$
2. $\aleph\alpha \to a\aleph$
3. $\aleph b \to \beta\aleph$
4. $\aleph\beta \to b\aleph$
5. $\aleph c \to \gamma\aleph$
6. $\aleph\gamma \to c\aleph$
7. $\aleph \to_\bullet \varepsilon$
8. $\varepsilon \to \aleph$

Beispiel: Sortieren

Aufgabe

Sortieren sie mit einem Markov-Algorithmus die Zeichen in einem Wort über $\Sigma = \{a, b, c, d\}$.
Ist dies auch mit einem Semi-Thue-System möglich?

Beispiel: Sortieren

Aufgabe

Sortieren sie mit einem Markov-Algorithmus die Zeichen in einem Wort über $\Sigma = \{a, b, c, d\}$.
Ist dies auch mit einem Semi-Thue-System möglich?

Lösung

1. $ba \longrightarrow ab$
2. $ca \longrightarrow ac$
3. $da \longrightarrow ad$
4. $db \longrightarrow bd$
5. $dc \longrightarrow cd$

Ja, das ist ohne weiteres mit diesen Regeln auch als Semi-Thue-System zu lösen.

Potente Algorithmen

Was macht dieser Markov-Algorithmus

Eingabe: $|^n \cdot ||\hat{}|^m$

1. $|\alpha \rightarrow \alpha||$
2. $\alpha \rightarrow \varepsilon$
3. $\cdot||\hat{}| \rightarrow \alpha \cdot ||\hat{}$
4. $\cdot||\hat{} \rightarrow_\bullet \varepsilon$

Potente Algorithmen

Was macht dieser Markov-Algorithmus

Eingabe: $|^n \cdot ||\hat{}|^m$

1. $|\alpha \rightarrow \alpha||$
2. $\alpha \rightarrow \varepsilon$
3. $\cdot||\hat{}| \rightarrow \alpha \cdot ||\hat{}$
4. $\cdot||\hat{} \rightarrow_\bullet \varepsilon$

Antwort

Er berechnet $n \cdot 2^m$ in Stichles-Schreibweise.

Was sind Grammatiken?

Markov-Algorithmen waren eine Variante der Semi-Thue-Systeme mit sinnvollen Einschränkungen. Für Grammatiken kann man das ähnlich machen:

- Trennung des Alphabets in Terminalsymbole Σ und Nicht-Terminale N.
- Die Regeln heißen Produktionen und stecken in der Menge P
- Man hat keine Eingebe, sondern beginnt immer mit dem gleichen Symbol A, genannt Axiom (Sinnvollerweise ein Nicht-Terminal)
- Das ganze packt man in ein Tupel $G = (\Sigma, N, P, A)$ und nennt man Grammatik.
- Statt für eindeutige Ableitungsergebnisse interessieren wir uns für alle möglichen Ableitungen des Axioms A aus Σ^*, genannt die Sprache der Grammatik und geschrieben $L(G)$

Was sind Grammatiken?

Markov-Algorithmen waren eine Variante der Semi-Thue-Systeme mit sinnvollen Einschränkungen. Für Grammatiken kann man das ähnlich machen:

- Trennung des Alphabets in Terminalsymbole Σ und Nicht-Terminale N.
- Die Regeln heißen Produktionen und stecken in der Menge P
- Man hat keine Eingebe, sondern beginnt immer mit dem gleichen Symbol A, genannt Axiom (Sinnvollerweise ein Nicht-Terminal)
- Das ganze packt man in ein Tupel $G = (\Sigma, N, P, A)$ und nennt man Grammatik.
- Statt für eindeutige Ableitungsergebnisse interessieren wir uns für alle möglichen Ableitungen des Axioms A aus Σ^*, genannt die Sprache der Grammatik und geschrieben $L(G)$

Was sind Grammatiken?

Markov-Algorithmen waren eine Variante der Semi-Thue-Systeme mit sinnvollen Einschränkungen. Für Grammatiken kann man das ähnlich machen:

- Trennung des Alphabets in Terminalsymbole Σ und Nicht-Terminale N.
- Die Regeln heißen Produktionen und stecken in der Menge P
- Man hat keine Eingebe, sondern beginnt immer mit dem gleichen Symbol A, genannt Axiom (Sinnvollerweise ein Nicht-Terminal)
- Das ganze packt man in ein Tupel $G = (\Sigma, N, P, A)$ und nennt man Grammatik.
- Statt für eindeutige Ableitungsergebnisse interessieren wir uns für alle möglichen Ableitungen des Axioms A aus Σ^*, genannt die Sprache der Grammatik und geschrieben $L(G)$

Was sind Grammatiken?

Markov-Algorithmen waren eine Variante der Semi-Thue-Systeme mit sinnvollen Einschränkungen. Für Grammatiken kann man das ähnlich machen:

- Trennung des Alphabets in Terminalsymbole Σ und Nicht-Terminale N.
- Die Regeln heißen Produktionen und stecken in der Menge P
- Man hat keine Eingebe, sondern beginnt immer mit dem gleichen Symbol A, genannt Axiom (Sinnvollerweise ein Nicht-Terminal)
- Das ganze packt man in ein Tupel $G = (\Sigma, N, P, A)$ und nennt man Grammatik.
- Statt für eindeutige Ableitungsergebnisse interessieren wir uns für alle möglichen Ableitungen des Axioms A aus Σ^*, genannt die Sprache der Grammatik und geschrieben $L(G)$

Was sind Grammatiken?

Markov-Algorithmen waren eine Variante der Semi-Thue-Systeme mit sinnvollen Einschränkungen. Für Grammatiken kann man das ähnlich machen:

- Trennung des Alphabets in Terminalsymbole Σ und Nicht-Terminale N.
- Die Regeln heißen Produktionen und stecken in der Menge P
- Man hat keine Eingebe, sondern beginnt immer mit dem gleichen Symbol A, genannt Axiom (Sinnvollerweise ein Nicht-Terminal)
- Das ganze packt man in ein Tupel $G = (\Sigma, N, P, A)$ und nennt man Grammatik.
- Statt für eindeutige Ableitungsergebnisse interessieren wir uns für alle möglichen Ableitungen des Axioms A aus Σ^*, genannt die Sprache der Grammatik und geschrieben $L(G)$

Was sind Grammatiken?

Markov-Algorithmen waren eine Variante der Semi-Thue-Systeme mit sinnvollen Einschränkungen. Für Grammatiken kann man das ähnlich machen:

- Trennung des Alphabets in Terminalsymbole Σ und Nicht-Terminale N.
- Die Regeln heißen Produktionen und stecken in der Menge P
- Man hat keine Eingebe, sondern beginnt immer mit dem gleichen Symbol A, genannt Axiom (Sinnvollerweise ein Nicht-Terminal)
- Das ganze packt man in ein Tupel $G = (\Sigma, N, P, A)$ und nennt man Grammatik.
- Statt für eindeutige Ableitungsergebnisse interessieren wir uns für alle möglichen Ableitungen des Axioms A aus Σ^*, genannt die Sprache der Grammatik und geschrieben $L(G)$

Chomsky-Hierarchie

Wir können die Grammatiken in vier Klassen einteilen, Chomsky-0 bis Chomsky3. Dabei gilt:

- Kleinere Chomsky-Nummern enthalten die größeren
- Größere Chomsky-Nummern enthalten die kleineren **nicht**
- Je kleiner die Nummer, desto mächtiger die Grammatik
- Chomsky-0 ist so gut wie Semi-Thue und Markov
- Chomsky-2-Grammatiken sind äquivalent zur (Erweiterten-)Backus-Naur-Form
- Auch die Sprachen werden in diese Klassen gepackt: Jede Sprache hat die Chomsky-Klasse der einfachsten Grammatik, die sie erzeugt. (Einfach $\hat{=}$ große Chomsky-Nummer)

Chomsky-Hierarchie

Wir können die Grammatiken in vier Klassen einteilen, Chomsky-0 bis Chomsky3. Dabei gilt:

- Kleinere Chomsky-Nummern enthalten die größeren
- Größere Chomsky-Nummern enthalten die kleineren **nicht**
- Je kleiner die Nummer, desto mächtiger die Grammatik
- Chomsky-0 ist so gut wie Semi-Thue und Markov
- Chomsky-2-Grammatiken sind äquivalent zur (Erweiterten-)Backus-Naur-Form
- Auch die Sprachen werden in diese Klassen gepackt: Jede Sprache hat die Chomsky-Klasse der einfachsten Grammatik, die sie erzeugt. (Einfach $\hat{=}$ große Chomsky-Nummer)

Chomsky-Hierarchie

Wir können die Grammatiken in vier Klassen einteilen, Chomsky-0 bis Chomsky3. Dabei gilt:

- Kleinere Chomsky-Nummern enthalten die größeren
- Größere Chomsky-Nummern enthalten die kleineren **nicht**
- Je kleiner die Nummer, desto mächtiger die Grammatik
- Chomsky-0 ist so gut wie Semi-Thue und Markov
- Chomsky-2-Grammatiken sind äquivalent zur (Erweiterten-)Backus-Naur-Form
- Auch die Sprachen werden in diese Klassen gepackt: Jede Sprache hat die Chomsky-Klasse der einfachsten Grammatik, die sie erzeugt. (Einfach $\hat{=}$ große Chomsky-Nummer)

Chomsky-Hierarchie

Wir können die Grammatiken in vier Klassen einteilen, Chomsky-0 bis Chomsky3. Dabei gilt:

- Kleinere Chomsky-Nummern enthalten die größeren
- Größere Chomsky-Nummern enthalten die kleineren **nicht**
- Je kleiner die Nummer, desto mächtiger die Grammatik
- Chomsky-0 ist so gut wie Semi-Thue und Markov
- Chomsky-2-Grammatiken sind äquivalent zur (Erweiterten-)Backus-Naur-Form
- Auch die Sprachen werden in diese Klassen gepackt: Jede Sprache hat die Chomsky-Klasse der einfachsten Grammatik, die sie erzeugt. (Einfach $\hat{=}$ große Chomsky-Nummer)

Chomsky-Hierarchie

Wir können die Grammatiken in vier Klassen einteilen, Chomsky-0 bis Chomsky3. Dabei gilt:

- Kleinere Chomsky-Nummern enthalten die größeren
- Größere Chomsky-Nummern enthalten die kleineren **nicht**
- Je kleiner die Nummer, desto mächtiger kann die Grammatik sein
- Chomsky-0 ist so gut wie Semi-Thue und Markov
- Chomsky-2-Grammatiken sind äquivalent zur (Erweiterten-)Backus-Naur-Form
- Auch die Sprachen werden in diese Klassen gepackt: Jede Sprache hat die Chomsky-Klasse der einfachsten Grammatik, die sie erzeugt. (Einfach $\hat{=}$ große Chomsky-Nummer)

Chomsky-Hierarchie

Wir können die Grammatiken in vier Klassen einteilen, Chomsky-0 bis Chomsky3. Dabei gilt:

- Kleinere Chomsky-Nummern enthalten die größeren
- Größere Chomsky-Nummern enthalten die kleineren **nicht**
- Je kleiner die Nummer, desto mächtiger kann die Grammatik sein
- Chomsky-0 ist so gut wie Semi-Thue und Markov
- Chomsky-2-Grammatiken sind äquivalent zur (Erweiterten-)Backus-Naur-Form
- Auch die Sprachen werden in diese Klassen gepackt: Jede Sprache hat die Chomsky-Klasse der einfachsten Grammatik, die sie erzeugt. (Einfach $\hat{=}$ große Chomsky-Nummer)

Chomsky-Hierarchie

Wir können die Grammatiken in vier Klassen einteilen, Chomsky-0 bis Chomsky3. Dabei gilt:

- Kleinere Chomsky-Nummern enthalten die größeren
- Größere Chomsky-Nummern enthalten die kleineren **nicht**
- Je kleiner die Nummer, desto mächtiger kann die Grammatik sein
- Chomsky-0 ist so gut wie Semi-Thue und Markov
- Chomsky-2-Grammatiken sind äquivalent zur (Erweiterten-)Backus-Naur-Form
- Auch die Sprachen werden in diese Klassen gepackt: Jede Sprache hat die Chomsky-Klasse der einfachsten Grammatik, die sie erzeugt. (Einfach $\hat{=}$ große Chomsky-Nummer)

Chomsky-Hierarchie

Wir können die Grammatiken in vier Klassen einteilen, Chomsky-0 bis Chomsky3. Dabei gilt:

- Kleinere Chomsky-Nummern enthalten die größeren
- Größere Chomsky-Nummern enthalten die kleineren **nicht**
- Je kleiner die Nummer, desto mächtiger kann die Grammatik sein
- Chomsky-0 ist so gut wie Semi-Thue und Markov
- Chomsky-2-Grammatiken sind äquivalent zur (Erweiterten-)Backus-Naur-Form
- Auch die Sprachen werden in diese Klassen gepackt: Jede Sprache hat die Chomsky-Klasse der einfachsten Grammatik, die sie erzeugt. (Einfach $\hat{=}$ große Chomsky-Nummer)

Checkliste für die Erkennung des Typs

Folge einfach den Fallunterscheidungen

- auf der linken Seite befinden sich nur einzelne Nicht-Terminale
  - auf der rechten Seite sind nur einzelne Terminal oder ein Nichtterminal gefolgt von einem Terminal (CH-3)
  - auf der rechten Seite sind nur einzelne Terminal oder ein Terminal gefolgt von einem Nichtterminal (CH-3)
  - sonst: (CH-2)
- falls Längenbeschränktheit zutrifft (CH-1) (Kontextsensitivität ist nur zur Veranschaulichung)
- sonst: (CH-0)

Chomskygrammatiken und -Sprachen

Aufgabe

Gegeben ist die Grammatik
$G = (\{a, b, c\}, \{S, A, B, C\}, P, S)$
mit den folgenden Produktionen P:

1. $S \to AB$
2. $A \to a$
3. $Ab \to aab$
4. $B \to b$
5. $B \to C$
6. $C \to cCc$
7. $C \to c$

Gebt die Chomsky-Klasse von G, $L(G)$ sowie die Klasse von $L(G)$ an.

Chomskygrammatiken und -Sprachen

Aufgabe

Gegeben ist die Grammatik $G = (\{a, b, c\}, \{S, A, B, C\}, P, S)$ mit den folgenden Produktionen P:

1. $S \to AB$
2. $A \to a$
3. $Ab \to aab$
4. $B \to b$
5. $B \to C$
6. $C \to cCc$
7. $C \to c$

Gebt die Chomsky-Klasse von G, $L(G)$ sowie die Klasse von $L(G)$ an.

Lösung

- Die Grammatik ist in CH-1: Alle Regeln sind längenbeschränkt.
- $L(G) = \{ab, aab\} \cup \{ac^{2n+1} | n \in \mathbb{N}_0\}$
- $L(G)$ ist CH-3 („regulär“)

Surfen mal anders

HTML

1. Erstelle eine Grammatik von möglichst einfachen Typ der einzelne HTML Tags als Sprache produziert. Ein solche Tag hat die Form:
   `<tag> text </tag>`
   Sowohl `tag` als auch `text` sollen sich aus normalen kleinen Buchstaben zusammensetzen.
2. Gib dann auch noch den Chomskey-Typ der Grammatik an!

HTML-Grammatik

$$G = (\{<, >, /, a, \ldots, z\}, \{S, A, B, C, D, E, F, T\}, P, S)$$

1. $S \rightarrow < A$
2. $A \rightarrow aA| \cdots |zA|aB| \cdots |zB$
3. $B \rightarrow > T$
4. $T \rightarrow aT| \cdots |zT|aC| \cdots |zC$
5. $C \rightarrow < D$
6. $D \rightarrow /E$
7. $E \rightarrow aE| \cdots |zE|aF| \cdots |zF$
8. $F \rightarrow >$

HTML-Grammatik

$G = (\{<, >, /, a, \ldots, z\}, \{S, A, B, C, D, E, F, T\}, P, S)$

1. $S \rightarrow < A$
2. $A \rightarrow aA| \cdots |zA|aB| \cdots |zB$
3. $B \rightarrow > T$
4. $T \rightarrow aT| \cdots |zT|aC| \cdots |zC$
5. $C \rightarrow < D$
6. $D \rightarrow /E$
7. $E \rightarrow aE| \cdots |zE|aF| \cdots |zF$
8. $F \rightarrow >$

Der Typ der Grammatik ist CH-3.

Geht das auch einfacher?

Frage

Es scheint so als wenn die beschriebene Grammtik sehr komplizert aussieht.
Geht das auch einfacher?

Geht das auch einfacher?

Frage

Es scheint so als wenn die beschriebene Grammtik sehr komplizert aussieht.
Geht das auch einfacher?

$G = (\{<, >, /, a, \ldots, z\}, \{S, T\}, P, S)$

1. $S \rightarrow < T > T < /T >$
2. $T \rightarrow aT| \cdots |zT|a| \cdots |z$

Geht das auch einfacher?

Frage

Es scheint so als wenn die beschriebene Grammtik sehr komplizert aussieht.
Geht das auch einfacher?

$G = (\{<, >, /, a, \ldots, z\}, \{S, T\}, P, S)$

1. $S \rightarrow < T > T < /T >$
2. $T \rightarrow aT| \cdots |zT|a| \cdots |z$

Wichtig ist, dass man hier unterscheidet, zwischen dem Typ der Grammatik und der Sprache! Denn...

Geht das auch einfacher?

Frage

Es scheint so als wenn die beschriebene Grammtik sehr komplizert aussieht.
Geht das auch einfacher?

$G = (\{<, >, /, a, \ldots, z\}, \{S, T\}, P, S)$

1. $S \rightarrow < T > T < /T >$
2. $T \rightarrow aT| \cdots |zT|a| \cdots |z$

Wichtig ist, dass man hier unterscheidet, zwischen dem Typ der Grammatik und der Sprache! Denn...
Typ der Grammatik hier ist: CH-2. Aber die Sprache bleibt die gleiche!

Akzeptor

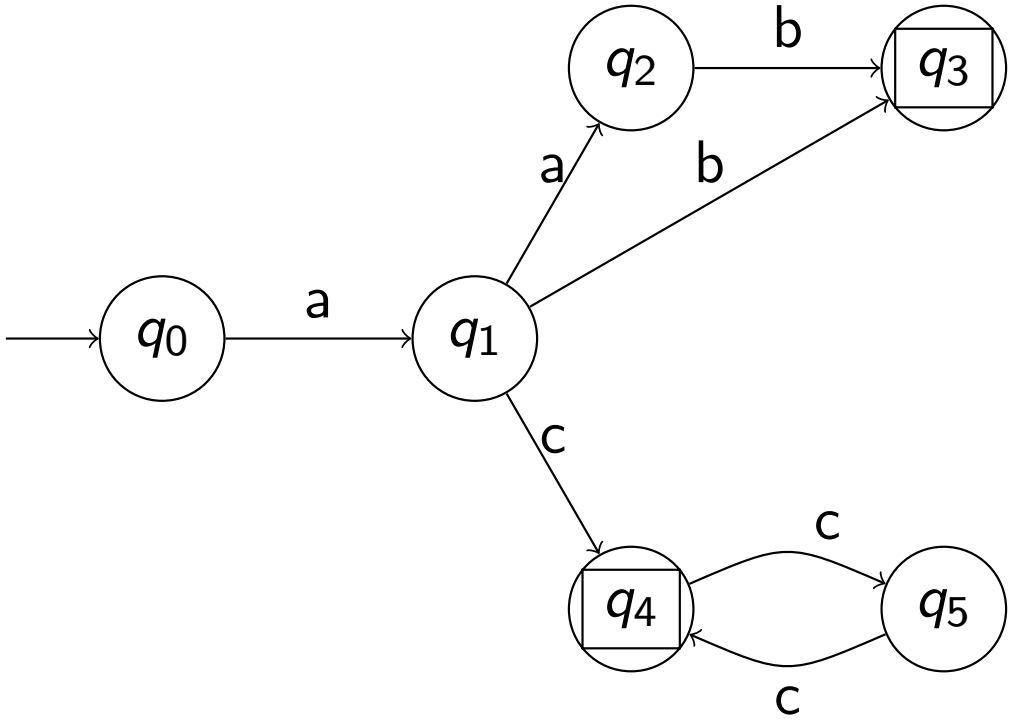
Aufgabe

Konstruiere einen vollständigen Automaten, der die Sprache $\{ab, aab\} \cup \{ac^{2n+1} | n \in \mathbb{N}_0\}$ akzeptiert.

Akzeptor

Aufgabe

Konstruiere einen vollständigen Automaten, der die Sprache $\{ab, aab\} \cup \{ac^{2n+1} | n \in \mathbb{N}_0\}$ akzeptiert.

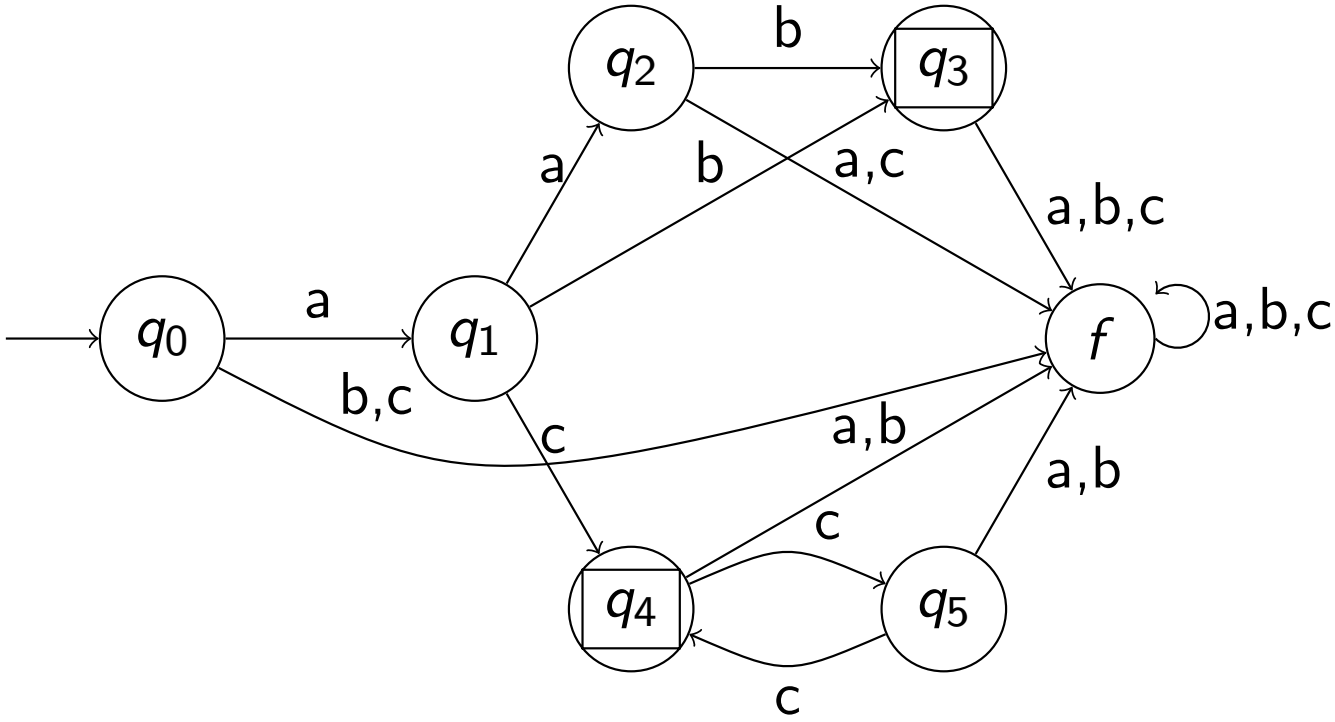


Achtung: der Graph ist noch nicht vollständig!

Akzeptor

Aufgabe

Konstruiere einen vollständigen Automaten, der die Sprache $\{ab, aab\} \cup \{ac^{2n+1} | n \in \mathbb{N}_0\}$ akzeptiert.



Vervollständigung durch die Einführung des Fehlerzustandes f

Was ist wahr und was ist falsch? (1)

- Grammatiken in CH-1 sind entweder längenbeschränkt oder kontextsensitiv
- Sind $P = \{A \to B, B \to Bx|x\}$ die Produktion einer Grammatik, so ist die erzeugte Sprache vom Typ CH-3.
- CH-2 Grammatiken können CH-3 Sprachen beschreiben.
- CH-3 Grammatiken können CH-2 Sprachen beschreiben.

Was ist wahr und was ist falsch? (2)

- Eine Grammatik hat die Form: $G = (\Sigma, P, N, A)$, wobei Σ das Alphabet, P die Menge der Produktionen, N die Menge der Nichtterminale, A das Axiom ist.
- Kontextfreiheit bedeutet mehr Mächtigkeit.
- Chomsky lebt.
- BNF wird benutzt, um Sprachen des Typs CH-3 zu beschreiben.