# Theoretische Grundlagen des Software Engineering

## 5: Reguläre Ausdrücke und Grammatiken

Stephan Schulz
schulz@eprover.org

Software
Systems
Engineering

Stiftung Universität Hildesheim
2003

# Reguläre Sprachen

Bisher: Charakterisierung von Sprachen über Automaten

- Gut für das schnelle Erkennen von Sprachen und Suche nach Worten

Aber:

- Sprache eines Automaten nicht offensichtlich
- Konstruktion der Automaten oft nicht-trivial

Geht das auch anders?

# Alternative Charakterisierungen

Reguläre Ausdrücke:

- Deklarative Beschreibung von regulären Sprachen
- Vielfach eingesetzt in textverarbeitenden Tools
  - grep, perl, awk, python, emacs...

Formale Grammatiken

- Generative Beschreibung von (regulären) Sprachen
- Standard-Beschreibung für Programmiersprachen
- Maschinen-lesbar für automatischen Compiler-Bau

# Reguläre Ausdrücke

Definition: Sei $\Sigma$ ein Alphabet. Die Menge der regulären Ausdrücke über $\Sigma$ ist rekursiv definiert wie folgt:

1. $a \in \Sigma$
2. $\epsilon$
3. $\emptyset$
4. $(R_1 \cup R_2)$ (wenn $R_1$ und $R_2$ reguläre Ausdrücke sind)
5. $(R_1 \circ R_2)$ (wenn $R_1$ und $R_2$ reguläre Ausdrücke sind)
6. $(R_1^*)$ (wenn $R_1$ ein regulärer Ausdruck ist)

# Sprache eines Regulären Ausdrucks

Definition: Sei $\Sigma$ ein Alphabet. Die Sprache eines regulären Ausdrucks über $\Sigma$ ist rekursiv definiert wie folgt:

1. $L(a) = \{a\}$ falls $a \in \Sigma$
2. $L(\epsilon) = \{\epsilon\}$
3. $L(\emptyset) = \{\}$
4. $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$
5. $L((R_1 \circ R_2)) = L(R_1) \circ L(R_2)$
6. $L((R_1^*)) = L(R_1)^*$

# Reguläre Ausdrücke und Reguläre Sprachen

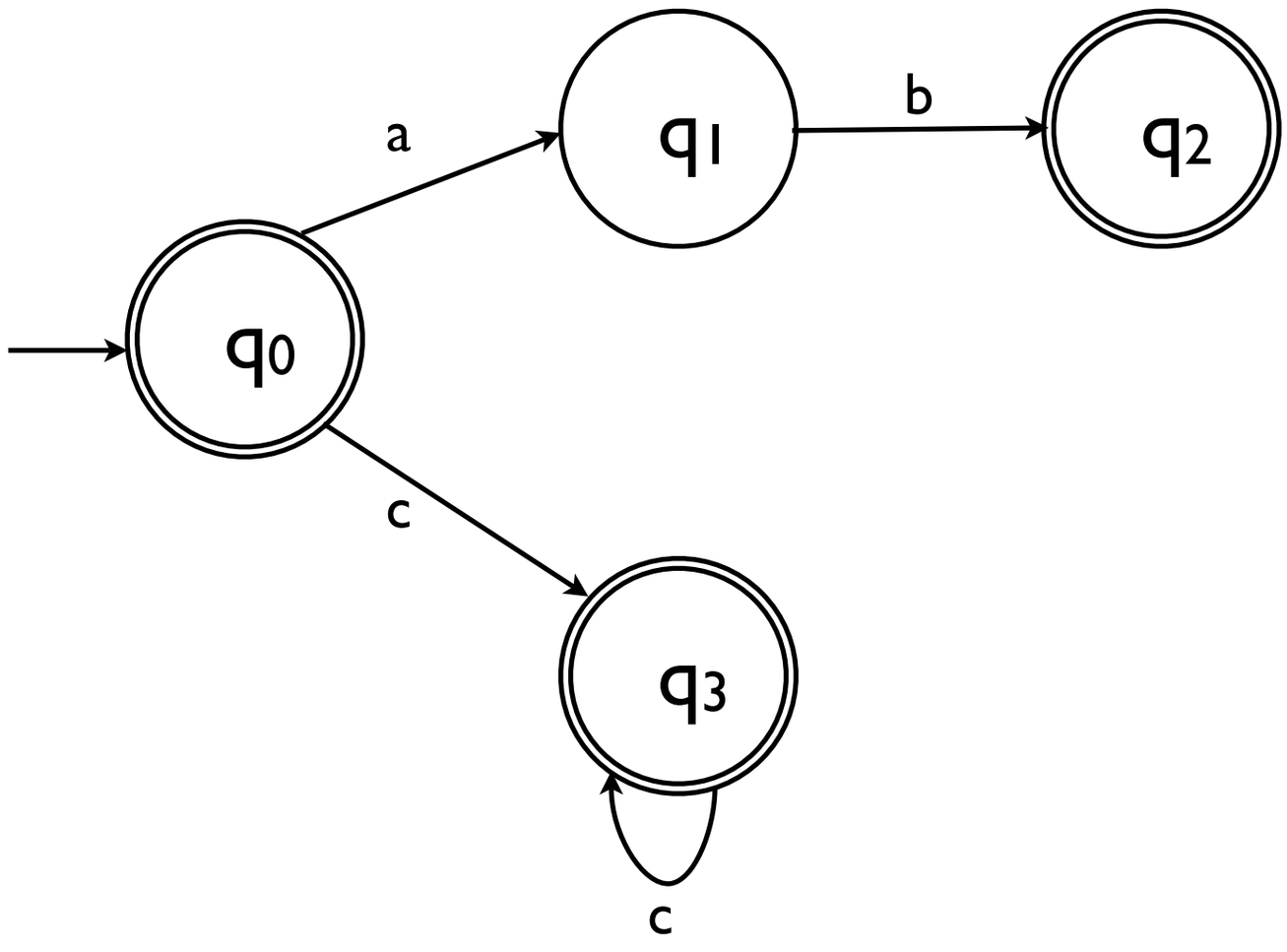
Theorem 6: Eine Sprache L über Σ ist genau dann regulär, wenn es einen regulären Ausdruck R mit L=L(R) gibt.

Also: Reguläre Ausdrücke beschreiben die gleichen Sprachen wie endliche Automaten!

- Fakt: Man kann reguläre Ausdrücke in Automaten übersetzen (relativ einfach)
- Fakt: Man kann endliche Automaten in reguläre Ausdrücke übersetzen (zu kompliziert, das formale sparen wir uns...)

# Beispiel

Sei $\Sigma=\{a,b,c\}$. Generiere Automaten für $L((a \circ b) \cup (c^*))$



# Beweis (Theorem 5 ⇒)

Behauptung: Sei R ein regulärer Ausdruck. Dann existiert ein ϵ-NEA E mit L(E)=L(R) (...und damit per Theorem 4 ein DEA A mit L(A)=L(R)

Beweis: Per struktureller Induktion (auch: "Induktion über den Aufbau")

- Basisfälle (Induktionsanfang): Definition RA 1-3
  - Für diese Sprachen gibt es Automaten (Tafel)

# Beweis (Theorem 5 ⇒) Induktionsschritt

Induktionsvorraussetzung: $R_1$ und $R_2$ seien reguläre Ausdrücke, $A_1$ und $A_2$ die ensprechenden Automaten mit $L(A_1)=L(R_1)$ und $L(A_2)=L(R_2)$

Induktionsschritt: Definition RA 4-6

- Faule Variante: Laut Induktionsvorraussetzung sind $L(R_1)$ und $L(R_2)$ regulär mit Automaten $A_1$ und $A_2$.
  - Laut Theorem 2 (b) ist $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$ regulär,

    laut Definition der regulären Sprachen gibt es also einen Automaten A mit $L(A)= L((R_1 \cup R_2))$

# Beweis (Theorem 5 ⇒) Induktionsschritt (2)

- Faule Variante (Teil 2): Laut Induktionsvorraussetzung sind $L(R_1)$ und $L(R_2)$ regulär mit Automaten $A_1$ und $A_2$.
  - Laut Theorem 2 (d) ist $L((R_1 \circ R_2)) = L(R_1) \circ L(R_2)$ regulär, laut Definition der regulären Sprachen gibt es also einen Automaten A mit $L(A) = L((R_1 \circ R_2))$
  - Laut Theorem 2 (e) ist $L((R_1^*)) = L(R_1)^*$ regulär, laut Definition der regulären Sprachen gibt es also einen Automaten A mit $L(A) = L((R_1^*))$
- Konstruktive Variante: Tafel!

# Bequemlichkeiten...

Wir definieren folgende Konventionen:

- Es existiert eine Präzedenz der regulären Operatoren:
  - $^*$ bindet stärker als $\circ$, $\circ$ bindet stärker als $\cup$
  - Dadurch redundant gewordene Klammern können weggelassen werden
- Statt $a \cup b$ schreiben wir auch $\{a,b\}$

# Beispiele

Sei $\Sigma=\{a,b,c\}$. Reguläre Ausdrücke für:

| | |
|---|---|
| $\{w \in \Sigma^* \mid w \text{ enthält a und b}\}$ | $\{abc\}^*a\{abc\}^*\ b\{abc\}^* \cup \{abc\}^*b\{abc\}^*a\{abc\}$ |
| $\{w \in \Sigma^* \mid \lvert w\rvert=3\}$ | $\{abc\}\{abc\}\{abc\}$ |
| $\{w \in \Sigma^* \mid \lvert w\rvert_a$ gerade, $\lvert w\rvert_b$ ungerade$\}$ | ? |

# Theorie und Praxis

## Reguläre Ausdrücke in UNIX: $\Sigma=\Sigma_{ascii}$

| Formal | Unix |
|---|---|
| a | a oder \a |
| {a,b,c} | [abc] |
| $R^*$ | R* |
| $R_1 \cup R_2$ | $R_1 \mid R_2$ |
| Kein Äquivalent | [^abc] |
| $R_1 \circ R_2$ | $R_1 R_2$ |

# Formale Grammatiken

Definition: Eine formale Grammatik ist ein 4-Tupel $G=(N, \Sigma, P, S)$, wobei:

- N ist ein endliches Alphabet (die nichtterminalen Symbole oder Variablen)
- $\Sigma$ ist ein endliches Alphabet (die terminalen Symbole)
- P ist eine Menge von Regeln oder Produktionen, wobei jede Regel die Form $(N \cup \Sigma)^* \rightarrow (N \cup \Sigma)^*$ hat.
- $S \in N$ ist das Startsymbol von G

# Ableitungen, L(G)

Definition: Sei G=(N, Σ, P, S) eine Grammatik. G definiert eine Ableitungsrelation $\Rightarrow_G$ auf $(N \cup \Sigma)^*$ wie folgt:

- $a \Rightarrow_G b$ falls $\exists u,v,p,q \in (N \cup \Sigma)^*$ und a=upv, b=uqv und $p \rightarrow q \in P$
- $a \overset{n}{\Longrightarrow}_G b$, falls $\exists a_0, ..., a_n \in (N \cup \Sigma)^*$, $a=a_0$, $b=a_n$ und $a_i \Rightarrow_G a_{i+1}$ für alle $i \in \{0,...n\}$
- $a \overset{*}{\Longrightarrow}_G b$, falls $\exists n \in \mathbf{N}$: $a \overset{n}{\Longrightarrow}_G b$

Die Sprache von G ist $L(G)=\{b \in \Sigma^* | S \overset{*}{\Longrightarrow}_G b\}$

# Beispiel

Betrachte $G=(\{S\}, \{abc\}, P, S)$ mit $P=\{S \rightarrow aSb, S \rightarrow \epsilon\}$

- $S \Rightarrow_G aSb \Rightarrow_G aaSbb \Rightarrow_G aaaSbbb \Rightarrow_G aaabbb$
- $L(G) = \{a^n b^n | n \in \mathbf{N_0}\}$
- Betrachte $H=(\{S\}, \{abc\}, P, S)$ mit $P=\{S \rightarrow aSa, S \rightarrow \epsilon\}$
- $S \Rightarrow_H aSa \Rightarrow_H aaSaa \Rightarrow_H aaaSaaa \Rightarrow_H aaaaaa$
- $L(G) = \{a^{2n} | n \in \mathbf{N_0}\}$ (Beweis per Induktion nach Ableitungslänge - Tafel)

# Beispiel (2)

Betrachte G=({S,A,B}, {abc}, P, S).

P={(1) S→A, (2) S→B, (3) A→aA, (4) B→bB, (5) A→ϵ,
(6) B→ϵ}

L(G) = ?

L(G) = $\{a^n | n \in \mathbf{N_0}\} \cup \{b^n | n \in \mathbf{N_0}\}$

# Rechtslineare Grammatiken

Definition: Sei G=(N, Σ, P, S). Eine Grammatik. G heißt rechtslinear, falls alle Produktionen in P von einer der Formen

- $A \rightarrow a$ ($A \in N, a \in \Sigma$)
- $A \rightarrow aB$ ($A,B \in N, a \in \Sigma$)
- $A \rightarrow \epsilon$ ($A \in N$)

Theorem 6: Eine Sprache L hat eine rechtslineare Grammatik G gdw. L(G) eine reguläre Sprache ist.

# Kontext-Freie Grammatiken

Definition: Sei G=(N, Σ, P, S) eine Grammatik. G heißt kontext-frei falls alle Produktionen in P von der Form

$A \rightarrow \gamma$ ($\gamma \in (\Sigma \cup N)^*$)

sind.

Eine Sprache L heißt kontext-frei, falls es eine kontextfreie Grammatik G mit L=L(G) gibt.

Kontextfreie Grammatiken werden für die Definition der meisten Programmiersprachen verwendet (als BNF/EBNF).

# Beispiele für kontext-freie Sprache

$L=\{a^n b^n | n \in \mathbf{N_0}\}$ ist kontext-frei.

- $G=(\{S\}, \{abc\}, P, S)$ mit $P=\{S \rightarrow aSb, S \rightarrow \epsilon\}$

Klassische kontext-freie Sprachen sind

- Arithmetische Audrücke “(a+b*(c-d))”
- Logische Formeln “((a∧¬b)→c)”

# Kontext-Sensitive Grammatiken

Definition: Sei G=(N, Σ, P, S). Eine Grammatik. G heißt kontext-sensitiv, falls alle Produktionen in P von einer der Formen

- $\alpha A \beta \rightarrow \alpha \gamma \beta$ ($\alpha, \beta, \gamma \in (\Sigma \cup N)^*, A \in N$)

- $S \rightarrow \epsilon$ (falls S nicht auf der rechten Seite eine Produktion vorkommt)

Kontext-sensitive Sprachen sind entscheidbar. Algol-68 war mit kontext-sensitiver Grammatik spezifiziert.

# Allgemeine Grammatiken

Definition: Sei $G=(N, \Sigma, P, S)$. G ist eine allgemeine Grammatik.

- Produktionen haben die Form $\alpha \rightarrow \beta$, $\alpha, \beta \in (\Sigma \cup N)^*$

Nicht alle von allgemeinen Grammatiken erzeugten Sprachen sind entscheidbar.

# Chomsky-Hierarchie

Die verschiedenen Klassen von Sprachen bilden eine Hierarchie:

- Alle regulären Sprachen sind kontext-frei
- Alle kontext-freien Sprachen sind kontext sensitiv
- Alle kontext-sensitiven Sprachen sind allgemein

Dies gilt leider aus historischen Gründen nicht für die Grammatiken!

# Ende

# Aufgaben

Sei $\Sigma$={0...9, a...z}. Charakterisieren Sie die folgenden Sprachen:

- $L(\{a..z\} \circ \{0...9,a...z\}^*)$
- $L((\{0...9\} \circ \{0...9\})^* \cup \{a\} \circ \{b\} \circ \{c\})$

Geben Sie reguläre Ausdrücke für folgende Sprache an:

- {x | x ist Dezimaldarstellung einer geraden Zahl}
- {x | x ist Hexadezimaldarstellung einer durch 4 teilbaren Zahl}

# Aufgaben

Links-lineare Grammatiken erlauben Produktionen der Form

- $A \rightarrow \epsilon, A \rightarrow a, A \rightarrow Ba$ ($A, B \in N, a \in \Sigma$)

- Sind die von links-linearen Grammatiken erzeugten Sprachen regulär?

Rechts-links-lineare Grammatiken erlauben Produktionen der Form

- $A \rightarrow \epsilon, A \rightarrow a, A \rightarrow Ba, A \rightarrow aB$ ($A, B \in N, a \in \Sigma$)

- Sind die von rechts-links-linearen Grammatiken erzeugten Sprachen regulär?