Grundbegriffe der Theoretischen Informatik

Sommersemester 2018 - Thomas Schwentick

Teil A: Reguläre Sprachen

1: Reguläre Ausdrücke: Motivation, Definition, Beispiele

Version von: 12. April 2018 (14:12)

Textsuche, nicht ganz präzise

Beispiel

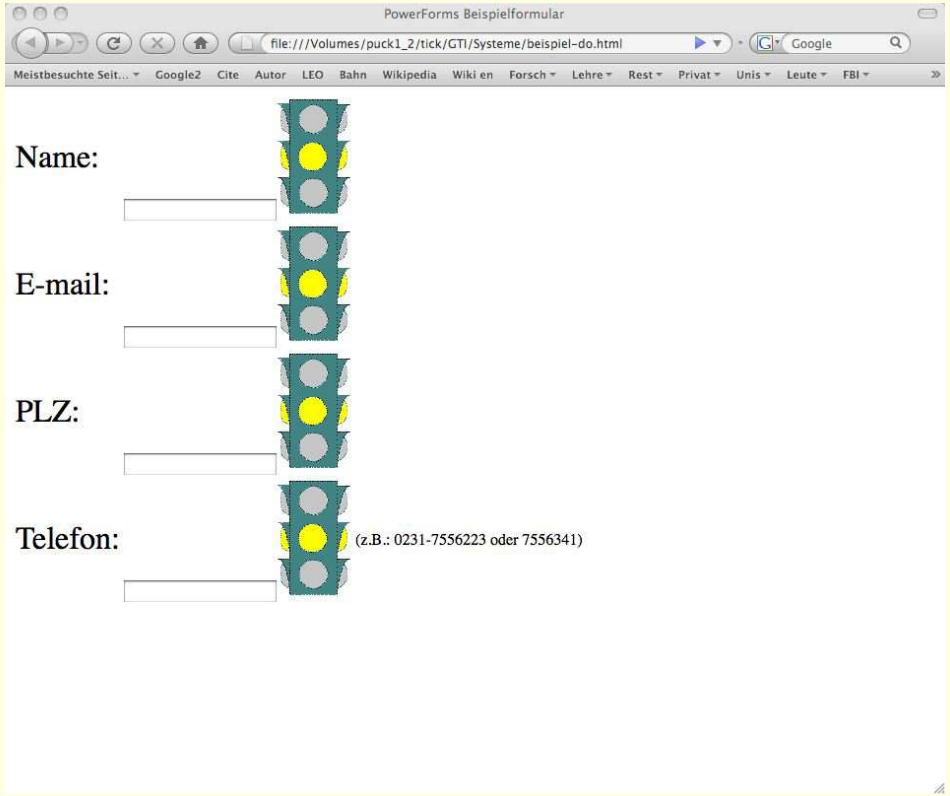
- Ich bin auf der Suche nach der Telefonnummer eines alten Schulfreundes
- Wie hieß er doch gleich?
  - Jansen?
  - Janssen?
  - Janssens?
  - Juan San?
- Am besten verwende ich ein Suchmuster:

  Jans[s]?en[s]?

Idee: Wim Martens

Inhalt

Beispiel: HTML-Formular mit e-Mail-Adresse



Hauptziele der nächsten vier Stunden

- Wir betrachten heute eine Methode, die es ermöglicht, **einfache** syntaktische Bedingungen für Zeichenketten **unzweideutig** zu beschreiben
  - Als Beispiel werden wir e-Mail-Adressen beschreiben

- Danach werden wir eine einfache Klasse von Programmen kennen lernen, die solche Bedingungen überprüfen können

  ☞ Dienstag

- Schließlich werden wir Methoden kennen lernen, mit denen wir Beschreibungen automatisch in Testprogramme übersetzen und diese minimieren können

e-Mail-Adressen: informelle Beschreibung

- Typische e-Mail-Adressen:
  - President@whitehouse.gov
  - thomas.schwentick@cs.tu-dortmund.de
  - zu_hause.hanna-mustermann82@irgendein.provider.de

- Die genauen **Regeln für e-Mail-Adressen** sind ziemlich **kompliziert**

➞ Wir betrachten deshalb nur eine **Annäherung** der korrekten e-Mail-Adressen

„Definition“: e-Mail-Adresse

- E-Mail-Adressen haben die Form „**Lokaler-Name@Domain-Name**“
- Ein „**Label**“ verwendet Zeichen aus: a-z, A-Z, 0-9, „-“, „_“
- Am Anfang eines Labels steht immer ein Buchstabe
- Der **lokale Name** besteht aus beliebig vielen (aber mindestens einem) Labels, die durch „.“ getrennt werden
- Der **Domain-Name** besteht aus beliebig vielen (aber mindestens zwei) Labels, die durch „.“ getrennt werden
- Das letzte Label im Domain-Namen besteht aus zwei bis vier Buchstaben

Zwischenziele für die heutige Vorlesung

- Formale Beschreibung aller Zeichenketten, die diesen Regeln entsprechen
- Da die Beschreibung (später) automatisch in ein Programm übersetzt werden soll, brauchen wir einen Beschreibungs-Formalismus mit einer klaren **Semantik**:

  reguläre Ausdrücke

✎ Vorher benötigen wir noch ein paar Grundbegriffe

Inhalt

Zeichenketten und Sprachen: informell

- Eine **Zeichenkette** (oder: ein **String**) ist eine (endliche) Folge von Zeichen
- Im Kontext von Computern spielen Zeichenketten und Mengen von Zeichenketten eine große Rolle

Beispiel

- Jedes Computer-Programm

```
public class Hallo
{
  public static void main(String[] args)
  {
    System.out.println("Hallo Welt!");
  }
}
```

ist eine Zeichenkette:

public class Hallo { public static void main(String[] args) { System.out.println("Hallo Welt!"); } }

- Die Menge der erlaubten Zeichen nennen wir **Alphabet**
- Eine Menge von Zeichenketten heißt **(formale) Sprache**

Beispiel

- Die Menge aller syntaktisch korrekten JAVA-Programme ist eine Sprache

Beispiel

- Die Menge aller syntaktisch korrekten JAVA-Programme, die „Hallo Welt!“ ausgeben, ist eine Sprache

Beispiel

- Die Menge der Befehlssequenzen einer Fernbedienung für einen Fernseher, nach deren Ausführung der Lautsprecher ausgeschaltet ist, ist eine Sprache über dem Alphabet der „Tasten“

✎ Jetzt: formale Definitionen

Alphabete, Wörter, Sprachen (1/3)

Definition (Alphabet)

- Ein **Alphabet** Σ ist eine endliche, nicht-leere Menge ☞ 1.1

- Die Elemente eines Alphabets heißen **Zeichen** oder **Symbole**

- Notation für Alphabete:
  - große, griechische Buchstaben, z.B. Σ („Sigma"), Δ („Delta"), Γ („Gamma")
- Notation für einzelne Zeichen:
  - kleine griechische Buchstaben, z.B. σ („sigma"), τ („tau"), $\ldots$

Beispiel

- $\{A, \ldots, Z\}$
- $\{0, 1\}$
- $\{\leftarrow, \uparrow, \downarrow, \rightarrow\}$

Definition (Wort)

- Ein **Wort** w über einem Alphabet Σ ist eine endliche Folge $\sigma_1 \cdots \sigma_n$ von Zeichen von Σ

- Wir bezeichnen Wörter auch als **Strings** oder **Zeichenketten**

- Notation für Wörter: $u, v, w, \ldots$
- Die **Länge** $|w|$ von $w = \sigma_1 \cdots \sigma_n$ ist die Anzahl n der Zeichen von w
- ϵ: Wort der Länge 0 (**leeres Wort**)

Beispiel

- $\Sigma = \{\text{a}, \ldots, \text{z}\}$
- Wörter über Σ:
  - informatik
  - jdgfsfsxnffishwrafdhug
  - abba
  - j
  - ϵ

Alphabete, Wörter, Sprachen (2/3)

Definition (Sprache)

- Eine **Sprache** über einem Alphabet Σ ist eine (endliche oder unendliche) Menge von Wörtern über Σ

Beispiel

- Menge aller syntaktisch korrekten e-Mail-Adressen
- Menge aller Strings über $\{0, 1\}$, die den Teilstring 010 enthalten
- Menge aller Strings über $\{0, 1\}$, die abwechselnd 0 und 1 enthalten

Alphabete, Wörter, Sprachen (3/3)

Beispiel

- Menge aller HTML-Tags:

$M_{\text{HTML}} \stackrel{\text{def}}{=} \{$⟨A⟩, ⟨ABBREV⟩, ⟨ACRONYM⟩, ⟨ADDRESS⟩, ⟨APPLET⟩, ⟨AREA⟩, ⟨AU⟩, ⟨AUTHOR⟩, ⟨B⟩, ⟨BANNER⟩, ⟨BASE⟩, ⟨BASEFONT⟩, ⟨BGSOUND⟩, ⟨BIG⟩, ⟨BLINK⟩, ⟨BLOCKQUOTE⟩, ⟨BQ⟩, ⟨BODY⟩, ⟨BR⟩, ⟨CAPTION⟩, ⟨CENTER⟩, ⟨CITE⟩, ⟨CODE⟩, ⟨COL⟩, ⟨COLGROUP⟩, ⟨CREDIT⟩, ⟨DEL⟩, ⟨DFN⟩, ⟨DIR⟩, ⟨DIV⟩, ⟨DL⟩, ⟨DT⟩, ⟨DD⟩, ⟨EM⟩, ⟨EMBED⟩, ⟨FIG⟩, ⟨FN⟩, ⟨FONT⟩, ⟨FORM⟩, ⟨FRAME⟩, ⟨FRAMESET⟩, ⟨H1⟩, ⟨H2⟩, ⟨H3⟩, ⟨H4⟩, ⟨H5⟩, ⟨H6⟩, ⟨HEAD⟩, ⟨HR⟩, ⟨HTML⟩, ⟨I⟩, ⟨IFRAME⟩, ⟨IMG⟩, ⟨INPUT⟩, ⟨INS⟩, ⟨ISINDEX⟩, ⟨KBD⟩, ⟨LANG⟩, ⟨LH⟩, ⟨LI⟩, ⟨LINK⟩, ⟨LISTING⟩, ⟨MAP⟩, ⟨MARQUEE⟩, ⟨MATH⟩, ⟨MENU⟩, ⟨META⟩, ⟨MULTICOL⟩, ⟨NOBR⟩, ⟨NOFRAMES⟩, ⟨NOTE⟩, ⟨OL⟩, ⟨OVERLAY⟩, ⟨P⟩, ⟨PARAM⟩, ⟨PERSON⟩, ⟨PLAINTEXT⟩, ⟨PRE⟩, ⟨Q⟩, ⟨RANGE⟩, ⟨SAMP⟩, ⟨SCRIPT⟩, ⟨SELECT⟩, ⟨SMALL⟩, ⟨SPACER⟩, ⟨SPOT⟩, ⟨STRIKE⟩, ⟨STRONG⟩, ⟨SUB⟩, ⟨SUP⟩, ⟨TAB⟩, ⟨TABLE⟩, ⟨TBODY⟩, ⟨TD⟩, ⟨TEXTAREA⟩, ⟨TEXTFLOW⟩, ⟨TFOOT⟩, ⟨TH⟩, ⟨THEAD⟩, ⟨TITLE⟩, ⟨TR⟩, ⟨TT⟩, ⟨U⟩, ⟨UL⟩, ⟨VAR⟩, ⟨WBR⟩, ⟨XMP⟩$\}$

- M_{HTML} ist eine Sprache über dem Alphabet

$$\Sigma \stackrel{\text{def}}{=} \{\langle, \rangle, /, A, \ldots, Z, 1, \ldots, 6\}$$

- Da M_{HTML} endlich ist, ist es selbst auch ein Alphabet

Beispiel

⟨HTML⟩
  ⟨HEAD⟩
    ⟨TITLE⟩GTI⟨/TITLE⟩
  ⟨/HEAD⟩
  ⟨BODY⟩
    ...
  ⟨/BODY⟩
⟨/HTML⟩

- ist ein Wort über dem Alphabet Σ,
- kann aber auch als Wort über dem Alphabet $M_{\text{HTML}} \cup M_{/\text{HTML}} \cup \{A, \ldots, Z\}$ aufgefasst werden
($M_{/\text{HTML}}$: Menge der schließenden Tags)

Konkatenation von Strings

Definition (Konkatenation)

- Das Zeichen „·“ bezeichnet die Operation der **Konkatenation** von Strings:
  - Sind u und v Wörter, so bezeichnet $u \cdot v$ das Wort, das entsteht, wenn v hinter u geschrieben wird

Beispiel

- $abb \cdot cda = abbcda$
- $a \cdot abc \cdot ba = aabcba$
- $abbc \cdot \epsilon = abbc$

Definition (Wiederholung)

- u^n bezeichnet die n-malige **Wiederholung** von u, also:
  - $u^0 = \epsilon$, $u^1 = u$, $u^2 = uu$, ...
- Induktiv: $u^0 = \epsilon$, $u^{n+1} = u^n \cdot u$

Beispiel

- $(ab)^2 = abab$
- $(aaa)^3 = aaaaaaaaa$
- $(abc)^0 = \epsilon$
- $\epsilon^3 = \epsilon$

- Wir werden das Operationssymbol · meistens weg lassen und einfach uv statt $u \cdot v$ schreiben
- Ist $u = abc$ und $v = bca$, so ist also $uv = abcbca$

Inhalt

Beschreibungsformalismus: Anforderungen

- Unser Beschreibungsformalismus soll
  - möglichst einfach sein, aber
  - genügend ausdrucksstark, um z.B. die Syntax von (unserer Definition von) Mailadressen beschreiben zu können
- Um Mailadressen adäquat beschreiben zu können, benötigen wir zumindest drei Konstruktionselemente:
  - Es muss möglich sein, Teilsprachen zu **konkatenieren**:
    - Lokaler-Name@Domain-Name
  - Es muss möglich sein, aus mehreren Alternativen **auszuwählen**:
    - „ „**Label**“ verwenden Zeichen aus: a-z, A-Z, 0-9, ...“
  - Es muss möglich sein, Elemente zu **wiederholen**:
    - „Der lokale Name besteht aus beliebig vielen (aber mindestens einem) Labels... “

Reguläre Ausdrücke: „Definition durch Beispiele“

- Die **Wiederholung** wird durch $*$ ausgedrückt:
  - b^* $\equiv$ „beliebig viele b“
  - Entspricht der Sprache
    $\{\epsilon, b, bb, bbb, bbbb, \ldots\}$

- Die **Konkatenation** wird wie ein Produkt geschrieben:
  - ab^* $\equiv$ ein a gefolgt von beliebig vielen b
  - Entspricht der Sprache
    $\{a, ab, abb, abbb, \ldots\}$

- Die **Auswahl** wird durch $+$ ausgedrückt:
  - $b + c^*$ $\equiv$ „ein b oder beliebig viele c“
  - Entspricht der Sprache $\{b, \epsilon, c, cc, ccc, \ldots\}$

✎ Die Beispiele entsprechen nicht genau der folgenden Definition der Syntax regulär Ausdrücke: es fehlen Klammern
  - Das Weglassen von Klammern werden wir aber später erlauben...

Reguläre Ausdrücke: Syntax

Definition (Regulärer Ausdruck: Syntax)

- Sei Σ ein Alphabet ☞ 1.2
- Die folgenden Regeln definieren die Menge der **regulären Ausdrücke über Σ**:

1) a) Das Zeichen $\varnothing$ ist ein regulärer Ausdruck
   b) Das Zeichen ϵ ist ein regulärer Ausdruck
   c) Für jedes $\sigma \in \Sigma$ ist σ ein regulärer Ausdruck

2) Sind α und β reguläre Ausdrücke, so auch
   a) $(\alpha\beta)$, und
   b) $(\alpha + \beta)$

3) Ist α ein regulärer Ausdruck, so auch (α^*)

✎ Wir verwenden die Abkürzung **RE** für „regulärer Ausdruck“, da der englische Begriff **regular expression** lautet
  – Entsprechend als Mehrzahl: **REs** für **regular expressions**

PINGO-Frage: pingo.upb.de

- Welche der folgenden Zeichenketten sind reguläre Ausdrücke?

(A) $(ab + c)^*$
(B) $(((ab)^*) + (b(a^*)))$
(C) $(a + +b)$
(D) $((b^*)^*)$

✎ Wir werden bald das Weglassen „überflüssiger“ Klammern erlauben

✎ Wir werden zur Bezeichnung regulärer Ausdrücke meist Buchstaben vom Anfang des griechischen Alphabets verwenden: $\alpha, \beta, \gamma, \dots$

Reguläre Ausdrücke: Semantik (1/3)

- Wir haben jetzt festgelegt, was reguläre Ausdrücke *sind*
- Wir haben also die **Syntax** regulärer Ausdrücke definiert
- Wir haben aber noch *nicht* definiert, was reguläre Ausdrücke *bedeuten*

- Im nächsten Schritt definieren wir deshalb die **Semantik** regulärer Ausdrücke
  - Dazu definieren wir zunächst Operatoren auf Sprachen, die den Konstruktionselementen regulärer Ausdrücke entsprechen
  - ✎ Zur Definition der Semantik der Auswahl benötigen wir keinen neuen Operator, da sie der Vereinigung entspricht

Definition ($\circ$, *)

- Sind L_1 und L_2 Sprachen, so sei:
  - $\underline{L_1 \circ L_2} \stackrel{\text{def}}{=} \{uv \mid u \in L_1, v \in L_2\}$
- Ist L eine Sprache, so sei
  - $\underline{L^*} \stackrel{\text{def}}{=} \{u_1 \cdots u_n \mid u_1, \ldots, u_n \in L, n \in \mathbb{N}_0\}$

1.3

Beispiel

- $\{a\}^* = \{\epsilon, a, aa, aaa, \ldots\}$
- Σ^* bezeichnet die Menge aller Strings über dem Alphabet Σ
- $\{a\}^* \circ \{b\}^* = \{\epsilon, a, b, aa, ab, bb, aaa, aab, abb, \ldots\}$
- $\{a\}^* \cup \{b\}^* = \{\epsilon, a, b, aa, bb, aaa, bbb, \ldots\}$

Reguläre Ausdrücke: Semantik (2/3)

- Die folgende Definition der Semantik regulärer Ausdrücke ordnet jedem regulären Ausdruck α eine Sprache $L(\alpha)$ zu
- Zum Beispiel soll gelten: $L((a^*)) = \{\epsilon, a, aa, aaa, \ldots\}$

Definition (Regulärer Ausdruck: Semantik)

- Für jeden regulären Ausdruck α über einem Alphabet Σ sei $L(\alpha)$ wie folgt definiert:
  - * $L(\varnothing) \stackrel{\text{def}}{=} \varnothing$
    * $L(\epsilon) \stackrel{\text{def}}{=} \{\epsilon\}$
    * $L(\sigma) \stackrel{\text{def}}{=} \{\sigma\}$, für jedes $\sigma \in \Sigma$
  - Sind α und β reguläre Ausdrücke so ist
    * $L((\alpha\beta)) \stackrel{\text{def}}{=} L(\alpha) \circ L(\beta)$,
    * $L((\alpha + \beta)) \stackrel{\text{def}}{=} L(\alpha) \cup L(\beta)$
  - Ist α ein regulärer Ausdruck, so ist $L((\alpha^*)) \stackrel{\text{def}}{=} L(\alpha)^*$

Definition (Reguläre Sprache)

- Eine Sprache L heißt **regulär**, falls es einen regulären Ausdruck α gibt mit $L = L(\alpha)$

Reguläre Ausdrücke: Semantik (3/3)

Regulärer Ausdruck: Semantik (Wdh.)

- Für jeden regulären Ausdruck α sei $L(\alpha)$ wie folgt definiert:
  - * $L(\varnothing) \stackrel{\text{def}}{=} \varnothing$
    * $L(\epsilon) \stackrel{\text{def}}{=} \{\epsilon\}$
    * $L(\sigma) \stackrel{\text{def}}{=} \{\sigma\}$, für jedes $\sigma \in \Sigma$
  - Sind α und β reguläre Ausdrücke so ist
    * $L((\alpha\beta)) \stackrel{\text{def}}{=} L(\alpha) \circ L(\beta)$,
    * $L((\alpha+\beta)) \stackrel{\text{def}}{=} L(\alpha) \cup L(\beta)$
  - Ist α ein regulärer Ausdruck, so ist $L((\alpha^*)) \stackrel{\text{def}}{=} L(\alpha)^*$

Beispiel

$$
\begin{aligned}
&L((((a^*)b) + (a(b^*))))\\
&= L(((a^*)b)) \cup L((a(b^*)))\\
&= (L((a^*)) \circ L(b)) \cup (L(a) \circ L((b^*)))\\
&= (\{\epsilon, a, aa, \ldots\} \circ \{b\}) \cup (\{a\} \circ \{\epsilon, b, bb, \ldots\})\\
&= \{b, a, ab, aab, abb, aaab, abbb, \ldots\}
\end{aligned}
$$

Zwischenbemerkung

Vorsicht

- Wir verwenden das Symbol $\varnothing$ mit zwei Bedeutungen:
  - $\varnothing$ bezeichnet einerseits die leere Menge
  - Andererseits kann $\varnothing$ auch als Zeichen in einem regulären Ausdruck vorkommen
- Auch das Symbol ϵ verwenden wir mit zwei Bedeutungen:
  - ϵ bezeichnet einerseits den Leerstring
  - Andererseits kann ϵ als Zeichen in einem regulären Ausdruck vorkommen
- Das ist so üblich
- Aus dem Kontext wird immer ersichtlich sein, was jeweils gemeint ist
- Desgleichen verwenden wir $*$ sowohl als syntaktisches Element von regulären Ausdrücken als auch als Operator

Inhalt

Reguläre Ausdrücke: Beispiele

- Die Beispiel-Ausdrücke sind schwer lesbar: zu viele Klammern

- Ohne Klammern ist die Semantik zunächst unklar: $ab+cd^*$ könnte bedeuten:
  - $((ab)+((cd)^*))$
  - $(((a(b+c))d)^*)$
  - $((ab)+(c(d^*)))$

- Deshalb verwenden wir Präzedenzregeln:
  - Klammern binden am stärksten
  - Dann *
  - Dann Konkatenation
  - Dann $+$

- Der Ausdruck $ab + cd^*$ steht also für $((ab) + (c(d^*)))$

Beispiel

- Die Menge aller Strings über $\{0, 1\}$, die 010 als Teilstring enthalten:

Beispiel

- Die Menge aller Strings über $\{0, 1\}$, die abwechselnd 0 und 1 enthalten:

  oder

Beispiel

- Die Menge aller Strings über $\{0, 1\}$, die gerade viele Einsen enthalten:

Reguläre Ausdrücke: Syntaktischer Zucker

- Um praktisch relevante Beispiele ausdrücken zu können, sind reguläre Ausdrücke manchmal etwas umständlich
- Deshalb werden in der Praxis meist **erweiterte reguläre Ausdrücke** verwendet, die abkürzende Schreibweisen wie die folgenden erlauben:

Definition (Erweiterter regulärer Ausdruck)

Abkürzung ...	... steht für ...	Erläuterung
$[a-z]$	$a+\cdots+z$	
$\alpha?$	$(\alpha+\epsilon)$	keinmal oder einmal α
α^+	$\alpha\alpha^*$	mindestens einmal α
α^n	$\alpha\cdots\alpha$	n mal α
$\alpha^{\{m,n\}}$	$\alpha^m(\alpha?)^{n-m}$	mindestens m-mal, höchstens n-mal α

- Damit können wir jetzt die Syntax von e-Mail-Adressen beschreiben:

$$([\text{a-zA-Z}][\text{a-zA-Z0-9}\backslash\text{-}\backslash\_]^*.)^*[\text{a-zA-Z}][\text{a-zA-Z0-9}\backslash\text{-}\backslash\_]^*@$$
$$([\text{a-zA-Z}][\text{a-zA-Z0-9}\backslash\text{-}\backslash\_]^*.)^+[\text{a-zA-Z}]^{\{2,4\}}$$

- Solange nicht ausdrücklich etwas anderes gesagt wird, werden wir uns aber im Folgenden auf „reine“ reguläre Ausdrücke beschränken!

Reguläre Ausdrücke: Äquivalenzen (1/4)

Satz 1.1

- Es gelten folgende Äquivalenzen für beliebige reguläre Ausdrücke α, β, γ:

Assoziativität bezüglich $+$ und $\circ$
  - $\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma$
  - $\alpha(\beta\gamma) \equiv (\alpha\beta)\gamma$

Kommutativität bezüglich $+$
  - $\alpha + \beta \equiv \beta + \alpha$

Neutrale Elemente bezüglich $+$ und $\circ$
  - $\varnothing + \alpha \equiv \alpha \equiv \alpha + \varnothing$
  - $\epsilon\alpha \equiv \alpha \equiv \alpha\epsilon$

Distributivität
  - $\alpha(\beta + \gamma) \equiv \alpha\beta + \alpha\gamma$
  - $(\alpha + \beta)\gamma \equiv \alpha\gamma + \beta\gamma$

Idempotenz des Stern-Operators
  - $(\alpha^*)^* \equiv \alpha^*$

Nullelement bezüglich $\circ$ und $*$
  - $\varnothing\alpha \equiv \varnothing \equiv \alpha\varnothing$
  - $\varnothing^* \equiv \epsilon$

Leerstring bzgl. $*$
  - $\epsilon^* \equiv \epsilon$

Definition (Äquivalenz von REs, $\equiv$)

- Für zwei reguläre Ausdrücke α, β schreiben wir $\alpha \equiv \beta$, falls $L(\alpha) = L(\beta)$

Beispiel

$$(1+\epsilon)(01)^*(0+\epsilon)$$
$$\equiv 1(01)^*(0+\epsilon) + (01)^*(0+\epsilon)$$
$$\equiv 1(01)^*0 + 1(01)^* + (01)^*0 + (01)^*$$

✎ Der letzte Ausdruck lässt sich vereinfachen zu $(10)^* + (10)^*1 + (01)^*0 + (01)^*$
  - Das lässt sich aber nicht mit den obigen Regeln herleiten

✎ Die Assoziativität bezüglich $\circ$ erlaubt uns, in einer Folge von Konkatenationen alle Klammern wegzulassen: $(((ab)c)d)e \equiv abcde \equiv a(b(c(de)))$

Reguläre Ausdrücke: Äquivalenzen (2/4)

- Wie lassen sich die Äquivalenzen aus Satz 1.1 **beweisen**?

- Wir betrachten einen Beispielbeweis für die Äquivalenz $\alpha(\beta+\gamma) \equiv \alpha\beta + \alpha\gamma$
- Wir müssen zeigen, dass
$$L(\alpha(\beta+\gamma)) = L(\alpha\beta+\alpha\gamma)$$

- Wir müssen also zeigen, dass zwei Sprachen gleich sind
- Sprachen sind Mengen von Strings...

- Gleichheit von zwei Mengen M_1, M_2 lässt sich meist am besten in zwei Schritten zeigen:
  - $M_1 \subseteq M_2$
  - $M_2 \subseteq M_1$
- Also:
  - für alle $w \in M_1$ gilt $w \in M_2$
  - für alle $w \in M_2$ gilt $w \in M_1$

Reguläre Ausdrücke: Äquivalenzen (3/4)

Beweis von $\alpha(\beta + \gamma) \equiv \alpha\beta + \alpha\gamma$

- Seien α, β, γ beliebig

- Wir wollen zuerst zeigen: $L(\alpha(\beta + \gamma)) \subseteq L(\alpha\beta + \alpha\gamma)$
- Sei also w ein beliebiger String aus $L(\alpha(\beta + \gamma))$

  ➨ es gibt $u \in L(\alpha)$ und $v \in L(\beta + \gamma)$ mit $w = uv$

  ☞ Semantik REs: Konkatenation

  ➨ $v \in L(\beta)$ oder $v \in L(\gamma)$ ☞ Semantik REs: Auswahl

- Wir unterscheiden zwei Fälle:

  – 1. Fall: $v \in L(\beta)$
    - ∗ Da $u \in L(\alpha)$ und $v \in L(\beta)$, ist $w = uv \in L(\alpha\beta)$

      ☞ Semantik REs: Konkatenation

    ➨ $w \in L(\alpha\beta + \alpha\gamma)$ ☞ Semantik REs: Auswahl

  – 2. Fall: $v \in L(\gamma)$

    ➨ $w = uv \in L(\alpha\gamma)$ ☞ Semantik REs: Konkatenation

    ➨ $w \in L(\alpha\beta + \alpha\gamma)$ ☞ Semantik REs: Auswahl

- Der Beweis von $L(\alpha\beta + \alpha\gamma) \subseteq L(\alpha(\beta + \gamma))$ ist ähnlich

Reguläre Ausdrücke: Äquivalenzen (4/4)

- Wie lässt sich beweisen, dass eine vermutete oder behauptete Äquivalenz **nicht** gilt?

- Wir betrachten als Beispiel die **nicht gültige** Äquivalenz
  „$\alpha + (\beta\gamma) \equiv (\alpha + \beta)(\alpha + \gamma)$“
- Wir formulieren zunächst eine mathematische Aussage, die die Ungültigkeit dieser Äquivalenz ausdrückt

Proposition 1.2

- Es gibt reguläre Ausdrücke α, β, γ mit $L(\alpha + (\beta\gamma)) \neq L((\alpha + \beta)(\alpha + \gamma))$

- Diese Proposition lässt sich sehr einfach durch Angabe eines **Beispiels** beweisen
  ✎ Das Beispiel belegt die Gültigkeit der Proposition, für die behauptete Äquivalenz ist es ein **Gegenbeispiel**

Beweis

- Wir wählen:
  – $\alpha = a, \beta = b, \gamma = c$

- Dann gilt:
  – $L(\alpha + (\beta\gamma)) = \{a, bc\}$
  – $L((\alpha + \beta)(\alpha + \gamma)) = \{aa, ba, ac, bc\}$

- Insbesondere:
  – $aa \notin L(\alpha + (\beta\gamma))$
  – $aa \in L((\alpha + \beta)(\alpha + \gamma))$

- Also gilt im allgemeinen **nicht** die Aussage $L((\alpha + \beta)(\alpha + \gamma)) \subseteq L(\alpha + (\beta\gamma))$ und deshalb **auch nicht** $L(\alpha + (\beta\gamma)) = L((\alpha + \beta)(\alpha + \gamma))$

✎ Bei Beweisen durch Gegenbeispiel sollte das Gegenbeispiel jeweils so konkret wie möglich gewählt werden

Reguläre Ausdrücke: Zusammenfassung

Themen dieses Kapitels

- Warum reguläre Ausdrücke?

- Grundbegriffe: Alphabete, Wörter, Sprachen

- Syntax regulärer Ausdrücke
- Semantik regulärer Ausdrücke
- Präzedenzregeln
- Erweiterte reguläre Ausdrücke

- Äquivalenzregeln für reguläre Ausdrücke

- Beweismethoden:
  - Mengengleichheit
  - Beweis durch Gegenbeispiel

Kapitelfazit

- Was haben wir bisher erreicht?
  - Wir können syntaktisch korrekte e-Mail-Adressen jetzt sauber spezifizieren
  - Wir können sie aber noch nicht automatisch in Algorithmen/Programme übersetzen

  ➞ Damit werden wir uns im nächsten Kapitel beschäftigen

Erläuterungen

Bemerkung 1.1

- Zu definierende **Begriffe** sind durch Unterstreichung, Fettdruck und dunkles Türkis zu erkennen
- Auch Schreibweisen wie u^n werden so gekennzeichnet
- Später werden diese Schreibweisen dann aber ohne Unterstreichung etc. verwendet

Bemerkung 1.3

✎ **Zu beachten:**

- $\mathbb{N}$ bezeichnet in dieser Vorlesung die Menge der natürlichen Zahlen **ohne 0**,
- $\mathbb{N}_0$ bezeichnet die natürlichen Zahlen mit 0
- L_1^* enthält also immer auch den Leerstring

Bemerkung 1.2

- Die Definition der Syntax regulärer Ausdrücke ist eine induktive Definition
  - ☞ Näheres dazu in der nächsten Vorlesung
- Die Menge aller regulären Ausdrücke über Σ ist selbst auch wieder eine Sprache
  - Alphabet: $\Sigma \cup \{\varnothing, \epsilon, +, ^*,), (\}$