

## 2 Reguläre Ausdrücke

Wir wollen (i.a. unendliche) Sprachen mit endlichen Mitteln darstellen, z.B. durch Grammatiken, nach denen die Sätze der Sprache gebildet werden dürfen. Es ist sinnvoll, von der Wahl des jeweiligen Alphabets  $\Sigma$  oder konkreten Sprachen  $L_0, L_1, L_2$  zu abstrahieren, wenn es um die Gültigkeit gewisser Gesetze wie

$$L_0 \cdot (L_1 \cup L_2) = L_0 \cdot L_1 \cup L_0 \cdot L_2$$

geht. Wir beschäftigen uns jetzt mit Möglichkeiten, Sprachen zu konstruieren. Dabei beschränken wir uns auf einige der schon betrachteten Operationen:

**Definition 2.1** REGULÄRE AUSDRÜCKE ÜBER  $\Sigma$  sind die folgenden und nur diese:

- (i) 0 und 1 sind reguläre Ausdrücke über  $\Sigma$ .
- (ii) Für jedes  $a \in \Sigma$  ist  $a$  ein regulärer Ausdruck über  $\Sigma$ .
- (iii) Sind  $r$  und  $s$  reguläre Ausdrücke über  $\Sigma$ , so auch  $(r \cdot s)$ ,  $(r + s)$ , und  $r^*$ .

Statt  $(r \cdot s)$  schreiben wir in Zukunft meistens  $rs$ , gelegentlich auch  $(r; s)$ .

In den folgenden Abschnitten behandeln wir drei verschiedene natürliche Interpretationen für reguläre Ausdrücke: Sprachen, zweistellige Relationen und iterative sequentielle Programme.

Für das Komplement und die Inverse  $R^{-1}$  einer Relation  $R$  bzw. die Spiegelung  $L^{-1}$  einer Sprache  $L$  wurde kein regulärer Ausdruck eingeführt. Der Grund liegt darin, daß diese Operationen bei der Interpretation als Programme keinen Sinn machen. Dasselbe gilt für die Divisionen  $\backslash$  und  $/$ , auf die wir aber in Abschnitt 3.4 zurückkommen.

### 2.1 Reguläre Ausdrücke als formale Sprachen

Sei  $(\Sigma^*, \cdot, \varepsilon)$  das Monoid der Wörter über  $\Sigma$ . Dann können wir jeden regulären Ausdruck  $r$  über  $\Sigma$  durch eine Sprache  $L(r)$  interpretieren:

$$\begin{array}{ll} L(0) & := \emptyset \\ L(1) & := \{\varepsilon\} \\ L(a) & := \{a\}, \quad \text{für } a \in \Sigma \end{array} \qquad \begin{array}{ll} L(r + s) & := L(r) \cup L(s) \\ L(r \cdot s) & := L(r) \cdot L(s) \\ L(r^*) & := L(r)^*. \end{array}$$

Mit anderen Worten, wir interpretieren reguläre Ausdrücke als formale Sprachen in der ALGEBRA ALLER SPRACHEN ÜBER  $\Sigma$ ,

$$\mathcal{L}_\Sigma := (2^{\Sigma^*}, \cup, \emptyset, \cdot, \{\varepsilon\}, *, \{a\})_{a \in \Sigma}.$$

Verschiedene Ausdrücke können dieselbe Sprache bezeichnen, z.B.

$$L(r) = L(r + 0), \quad L(0^*) = L(1), \quad L(r + s) = L(s + r), \quad L(r \cdot 0) = L(0).$$

Sprachen lassen sich natürlich auch anders als durch reguläre Ausdrücke beschreiben, etwa durch eine charakteristische Eigenschaft:

$$\begin{aligned} L((a + b)^* a) &= L((a + b)^*) \cdot L(a) \\ &= (L(a) \cup L(b))^* \cdot L(a) \\ &= (\{a\} \cup \{b\})^* \cdot L(a) = \{a, b\}^* \{a\} \\ &= \{w \in \{a, b\}^* \mid w \neq \varepsilon \text{ und der letzte Buchstabe von } w \text{ ist } a\} \end{aligned}$$

**Definition 2.2** Die Klasse  $Reg_\Sigma$  der REGULÄREN SPRACHEN ÜBER  $\Sigma$  ist die kleinste Klasse  $\mathcal{L}$  mit

- (i)  $\emptyset \in \mathcal{L}$  und  $\{\varepsilon\} \in \mathcal{L}$ .
- (ii)  $\{a\} \in \mathcal{L}$  für jedes  $a \in \Sigma$ .
- (iii)  $L_1 \in \mathcal{L}, L_2 \in \mathcal{L} \implies L_1^* \in \mathcal{L}, L_1 \cdot L_2 \in \mathcal{L}, L_1 \cup L_2 \in \mathcal{L}$ .

**Satz 2.3** Eine Sprache  $R$  über  $\Sigma$  ist regulär genau dann, wenn es einen regulären Ausdruck  $r$  mit  $R = L(r)$  gibt.

**Beweis:** Sei  $\mathcal{L}(\Sigma) := \{L(r) \mid r \text{ ist ein regulärer Ausdruck über } \Sigma\}$ .

$Reg_\Sigma \subseteq \mathcal{L}(\Sigma)$ : Beachte, daß  $\mathcal{L}(\Sigma)$  abgeschlossen ist unter den Operationen  $\emptyset, 1, \{a\}$  für  $a \in \Sigma, +, \cdot$  und  $*$ . Da  $Reg_\Sigma$  die kleinste Klasse  $\mathcal{L} \subseteq 2^{\Sigma^*}$  ist, die unter diesen Operationen abgeschlossen ist, gilt  $Reg_\Sigma \subseteq \mathcal{L}(\Sigma)$ .

$\mathcal{L}(\Sigma) \subseteq Reg_\Sigma$ : Für jeden regulären Ausdruck  $r$  über  $\Sigma$  zeigen wir durch Induktion, daß  $L(r) \in Reg_\Sigma$ .

Induktionsanfang:  $L(0) = \emptyset \in Reg_\Sigma$ ,  $L(1) = \{\varepsilon\} \in Reg_\Sigma$  und  $L(a) = \{a\} \in Reg_\Sigma$  auf Grund der Definitionen.

Induktionsschritt: Seien schon  $L(r) \in Reg_\Sigma$  und  $L(s) \in Reg_\Sigma$ . Dann gelten:

$$\begin{aligned} L(r \cdot s) &= L(r) \cdot L(s) \in Reg_\Sigma, \\ L(r + s) &= L(r) \cup L(s) \in Reg_\Sigma, \\ L(r^*) &= L(r)^* \in Reg_\Sigma, \end{aligned}$$

da  $Reg_\Sigma$  abgeschlossen ist unter  $\cdot$ ,  $+$  und  $*$  auf Grund der Definition.  $\square$

Die ALGEBRA DER REGULÄREN SPRACHEN ÜBER  $\Sigma$ ,

$$Reg_\Sigma^{\mathcal{L}} := (Reg_\Sigma, \cup, \emptyset, \cdot, \{\varepsilon\}, *, \{a\}_{a \in \Sigma}),$$

besteht aus der Menge  $Reg_\Sigma$  und der Einschränkung der angegebenen Operationen von  $\mathcal{L}_\Sigma$  auf diese Teilmenge von  $2^{\Sigma^*}$ . Man nennt  $Reg_\Sigma^{\mathcal{L}}$  auch die Algebra der regulären EREIGNISSE, da sie von S.C.Kleene ursprünglich zur Darstellung des Verhaltens von Nervenzellen verwendet wurde.  $Reg_\Sigma^{\mathcal{L}}$  ist eine echte Unter algebra von  $\mathcal{L}_\Sigma$ :

**Proposition 2.4** Nicht jede Sprache über  $\Sigma \neq \emptyset$  ist regulär.

**Beweis:** Sei  $\{w_n \mid n \in \mathbb{N}\}$  eine Aufzählung aller Wörter über  $\Sigma$ , wobei etwa zuerst die der Länge 0, dann die der Länge 1, usw. genannt werden. Sei analog  $\{r_n \mid n \in \mathbb{N}\}$  eine Aufzählung aller regulären Ausdrücke über  $\Sigma$ . Betrachte  $r_n$  als charakteristische Funktion  $r_n : \Sigma^* \rightarrow \{0, 1\}$  der Sprache  $L(r_n)$ , also  $r_n(w_m) = 1 \iff w_m \in L(r_n)$ . Nach Satz 1.3 ist die durch “Diagonalisierung” über die unendliche Tabelle

	$w_1$	$w_2$	$w_3$	$w_4$	$\dots$
$r_1$	0	1	1	0	$\dots$
$r_2$	1	1	0	0	$\dots$
$\vdots$			$\ddots$		

aller  $r_n$  definierte Funktion  $r : \Sigma^* \rightarrow \{0, 1\}$ , mit  $r(w_n) := 1 - r_n(w_n)$ , von allen  $r_n$  verschieden. Daher ist  $L_r := \{w \in \Sigma^* \mid r(w) = 1\}$  von allen Sprachen  $L(r_n)$  verschieden, also keine reguläre Sprache.  $\square$

Genauer gesagt, gibt es nur abzählbar viele reguläre Ausdrücke, aber überabzählbar viele Sprachen über  $\Sigma \neq \emptyset$ . Mit regulären Ausdrücken kann man also bei weitem nicht alle Sprachen bezeichnen. (Beispiele nicht-regulärer Sprachen werden später angegeben.) Dasselbe gilt für die später betrachteten kontextfreien Grammatiken und überhaupt alle Sprachdefinitionen mit endlichen Mitteln.

**Beispiel 2.5** (i)  $w \in \Sigma^* \implies \{w\}$  ist regulär:  $L(w) = \{w\}$ .

(ii)  $L \subseteq \Sigma^*$  endlich  $\implies L$  ist regulär:  $L = \{w_1, \dots, w_k\} = L(w_1 + \dots + w_k)$ .

(iii)  $\Sigma^*$  ist regulär:  $\Sigma^* = L((a_1 + \dots + a_n)^*)$ , für  $\Sigma = \{a_1, \dots, a_n\}$ .

- (iv) In Beschreibungen von Programmiersprachen werden oft Teilsprachen durch reguläre Ausdrücke angegeben. Dabei benutzt man  $r \mid s$  statt  $r + s$  und definiert mehrere Sprachen simultan (was noch genauer betrachtet werden muß). Zahlausdrücke und Bezeichner definiert man etwa so:

$$\begin{aligned} \text{digit} &:= 0 \mid 1 \mid \dots \mid 9 \\ \text{letter} &:= a \mid \dots \mid z \mid A \mid \dots \mid Z \\ \text{number} &:= ((\varepsilon \mid -)(1 \mid \dots \mid 9) \text{ digit}^* \mid 0)(\varepsilon \mid \times(1 \mid \dots \mid 9) \text{ digit}^* \mid \times 0) \\ \text{identifizier} &:= \text{letter} (\text{letter} \mid \text{digit})^* \end{aligned}$$

- (v) In der Unix-Shell werden Dateinamen mit regulären Ausdrücken beschrieben, sodaß Befehle sehr kurz formuliert werden können. Beispielsweise bedeutet  $ls a * .asm$  dann: "Liste alle Assembler-Dateien, deren Namen mit  $a$  beginnen, auf". Assembler-Dateien haben in der Regel das Suffix ".asm".

Die regulären Sprachen erhält man natürlich auch als den Abschluß *aller endlichen* Sprachen unter Vereinigung, Sprachprodukt und Sprachiteration.

Sei allgemeiner  $\mathcal{K} = (K, +, 0, \cdot, 1, *)$  eine Menge  $K$  mit ausgezeichneten Elementen  $0, 1 \in K$ , zweistelligen Operationen  $+$  und  $\cdot$  und einer einstelligen Operation  $*$  (mit den in Abschnitt 3.2 angegebenen Eigenschaften). Die VON  $A \subseteq K$  ERZEUGTE REGULÄRE UNTERALGEBRA  $Reg_A^{\mathcal{K}}$  von  $\mathcal{K}$  sei die Menge

$$Reg_A := \bigcap \{ M \mid A \subseteq M \subseteq K, 0, 1 \in M, M + M \subseteq M, M \cdot M \subseteq M, M^* \subseteq M \}$$

zusammen mit den Einschränkungen der Operationen aus  $\mathcal{K}$  auf  $Reg_A$ .

Erlaubt man beim Aufbau der regulären Ausdrücke statt der Konstanten  $a \in \Sigma$ , die fest interpretiert sind, Variable  $x_1, x_2, \dots$ , so gilt analog zu Satz 2.3:

$$Reg_A^{\mathcal{K}} = \{ r^{\mathcal{K}}(a_1, \dots, a_m) \mid r(x_1, \dots, x_m) \text{ ein regulärer Ausdruck, } a_1, \dots, a_m \in A \}.$$

Dabei ist  $r^{\mathcal{K}}(a_1, \dots, a_m)$  der WERT des Ausdrucks  $r$  in  $\mathcal{K}$ , definiert durch

$$\begin{aligned} 0^{\mathcal{K}}(a_1, \dots, a_m) &:= 0, & (r + s)^{\mathcal{K}}(a_1, \dots, a_m) &:= r^{\mathcal{K}}(a_1, \dots, a_m) + s^{\mathcal{K}}(a_1, \dots, a_m), \\ 1^{\mathcal{K}}(a_1, \dots, a_m) &:= 1, & (r \cdot s)^{\mathcal{K}}(a_1, \dots, a_m) &:= r^{\mathcal{K}}(a_1, \dots, a_m) \cdot s^{\mathcal{K}}(a_1, \dots, a_m), \\ x_i^{\mathcal{K}}(a_1, \dots, a_m) &:= a_i, & (r^*)^{\mathcal{K}}(a_1, \dots, a_m) &:= (r^{\mathcal{K}}(a_1, \dots, a_m))^*. \end{aligned}$$

Die Algebra  $Reg_{\Sigma}^{\mathcal{L}}$  der regulären Sprachen über  $\Sigma$  ist gerade die von  $\Sigma$  erzeugte reguläre Unter algebra von  $\mathcal{L} = \mathcal{L}_{\Sigma}$ .

## 2.2 Reguläre Ausdrücke als zweistellige Relationen

Sei  $M$  eine Menge  $\neq \emptyset$  und  $2^{M \times M}$  die Menge aller zweistelligen Relationen auf  $M$ . Darauf hatten wir die speziellen Relationen  $\emptyset$  und  $1_M = Id_M$  definiert, sowie die Operationen

$$\begin{aligned} R_1 \cup R_2 &:= \{(a, b) \in M \times M \mid (a, b) \in R_1 \text{ oder } (a, b) \in R_2\}, \\ R_1 \circ R_2 &:= \{(a, c) \in M \times M \mid \text{Es gibt } b \in M \text{ mit } (a, b) \in R_1 \text{ und } (b, c) \in R_2\}, \\ R^* &:= \bigcap \{S \subseteq M \times M \mid R \subseteq S, S \text{ ist reflexiv und transitiv}\}. \end{aligned}$$

Wie für  $L^*$  bei Sprachen sieht man ähnlich wie im Beweis von Lemma 1.2:

$$R^* = \bigcup_{n=0}^{\infty} R^n \quad \text{mit } R^0 := 1_M, R^{n+1} := R^n \circ R.$$

Sei für jedes  $a \in \Sigma$  eine Relation  $R_a \subseteq M \times M$  gegeben, z.B.  $\{(a, a)\}$  im Falle  $\Sigma \subseteq M$ . Dann können wir jeden regulären Ausdruck  $r$  über  $\Sigma$  in der Algebra

$$\mathcal{R}_M := (2^{M \times M}, \emptyset, \cup, 1_M, \circ, *, R_{a_1}, \dots, R_{a_n})$$

interpretieren durch

$$\begin{aligned} R(0) &:= \emptyset & R(r \cdot s) &:= R(r) \circ R(s) \\ R(1) &:= 1_M & R(r + s) &:= R(r) \cup R(s) \\ R(a) &:= R_a \quad \text{für } a \in \Sigma, & R(r^*) &:= R(r)^*. \end{aligned}$$

Man beachte, daß für diese Interpretation Gleichungen gelten können, die bei der Interpretation durch Sprachen nicht gelten: Für verschiedene  $a, b, c \in \Sigma$  kann etwa  $R(a \cdot b) = R(c)$  sein, während stets  $L(a \cdot b) = \{ab\} \neq \{c\} = L(c)$  ist.

Die von  $A := \{R_a \mid a \in \Sigma\}$  erzeugte reguläre Unteralgebra

$$\text{Reg}_A^{\mathcal{R}} = \{r^{\mathcal{R}}(R_{a_1}, \dots, R_{a_m}) \mid r(x_1, \dots, x_m) \text{ ein regulärer Ausdruck, } R_{a_1}, \dots, R_{a_m} \in A\}$$

von  $\mathcal{R}_M$  ist die Algebra der von  $A$  erzeugten REGULÄREN RELATIONEN AUF  $M$ . Oft schreiben wir auch  $\mathcal{R}_M$  für  $(2^{M \times M}, \cup, \emptyset, \circ, 1_M, *)$  ohne ausgezeichnete Relationen.

## 2.3 Reguläre Ausdrücke als Programme

Wir können reguläre Ausdrücke auch als eine abstrakte Form von (sequentiellen, nicht deterministischen) Programmen ansehen, oder als Aktionen, die man ausführen soll.

Wir geben Programme als Flußdiagramme an. Dies sind hierarchisch aufgebaute Graphen mit einem ausgezeichneten Anfangs- und Endpunkt:

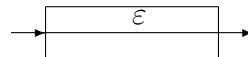


Die Grundprogramme oder BASISAKTIONEN  $P(a)$ , für  $a \in \Sigma$ , werden nicht weiter analysiert; ihre Namen  $a$  werden zur Markierung der Kanten des Graphen verwendet.

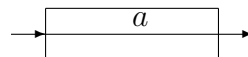
$P(0) = \text{loop forever} =$



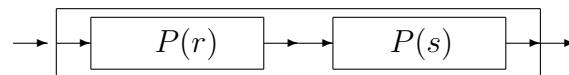
$P(1) = \text{skip} =$



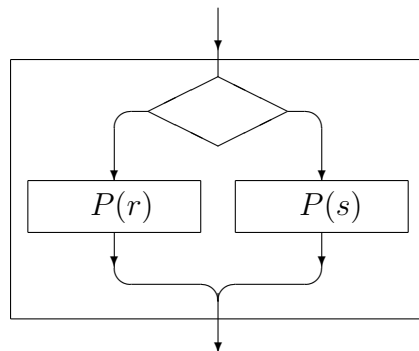
$P(a) = \text{do } a =$



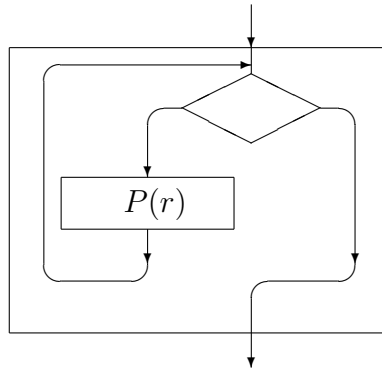
$P(r \cdot s) = (\text{first do } r \text{ and then do } s) =$



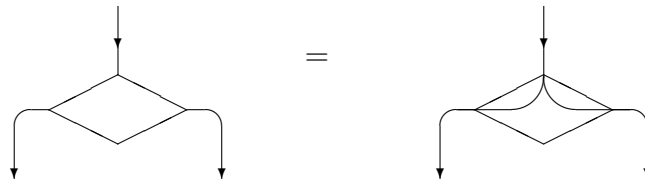
$P(r + s) = (\text{do } r \text{ or do } s) =$



$P(r^*) = (\text{do } r \text{ repeatedly}) = (\text{choose } n \in \mathbb{N} \text{ and then do } r \text{ } n \text{ times}) =$



Da wir bei der Parallelschaltung  $P(r + s)$  und bei der Iteration  $P(r^*)$  die Wahl der Ausführung nicht von einer Fallunterscheidung abhängig machen, sind die Verzweigungsrauten wie in



zu verstehen. Wie man hier Tests einfügt, um von nicht-deterministischen zu deterministischen Programmen zu kommen, tragen wir im folgenden Abschnitt nach.

## 2.4 Der Zusammenhang zwischen den Interpretationen

Zwischen den drei bisher betrachteten Interpretationen regulärer Ausdrücke besteht ein wichtiger Zusammenhang.

Wir können erstens die regulären Ausdrücke  $r$  über  $\Sigma$  mit den Flußdiagrammen  $P(r)$  oder den entsprechenden Programmen identifizieren. Dabei betrachten wir alle Flußdiagramme als gleich, die sich nur durch Vergrößerungen und Streckungen voneinander unterscheiden. Davon abgesehen ist der Aufbau der Flußdiagramme der gleiche wie der der regulären Ausdrücke. Wir sprechen daher statt von regulären Ausdrücken von den REGULÄREN PROGRAMMEN über den Grundbefehlen  $\Sigma$ .

Betrachtet man die regulären Ausdrücke als Programme, so entsprechen die Sprach- und die Relationeninterpretation zwei verschiedenen Arten, die Bedeutungen von Programmen zu erklären:

Die OPERATIONALE BEDEUTUNG oder das ENDLICHE VERHALTEN (englisch: TRACE) eines Programms ist die Gesamtheit aller Folgen atomarer Befehle, die in terminierenden Ausführungen des Programms durchlaufen werden.<sup>1</sup> Diese operationale Bedeutung entspricht sehr direkt dem Verhalten einer Maschine, die das Programm ausführt. Wir setzen also:

$$\begin{aligned} & \textit{Operationale Bedeutung von } r \\ &= \textit{Menge aller Folgen von Grundbefehlen, die einen Weg durch das} \\ & \quad \textit{Flußdiagramm } P(r) \textit{ zu } r \textit{ vom Start- zum Endpunkt markieren,} \\ &= \mathcal{L}(r). \end{aligned}$$

Die DENOTATIONALE BEDEUTUNG eines Programms ist die Beziehung zwischen Ein- und Ausgaben des Programms. Von Ein- und Ausgaberroutinen abgesehen, ist eine EINGABE einfach eine Belegung des Speichers *vor* und eine AUSGABE eine Belegung des Speichers *nach* der Programmausführung.

Eine Belegung des Speichers, oder ein SPEICHERZUSTAND, besteht im konkreten Fall in der Angabe, welche Werte in welchen Speicherzellen stehen; für den Programmierer genügt es zu wissen, welche Werte die im Programm vorkommenden Variablen haben. Da wir hier die Grundbefehle (z.B. Zuweisungen  $x := y + 3$ ) der Programme nicht weiter analysieren, betrachten wir eine beliebige Menge  $M$  von Speicherzuständen.

$$\begin{aligned} & \textit{Denotationale Bedeutung von } r \\ &= \{ (m, n) \in M \times M \mid \textit{der Speicherzustand } m \textit{ wird durch eine} \\ & \quad \textit{Ausführung von } r \textit{ in den Speicherzustand } n \textit{ überführt} \} \\ &= \mathcal{R}(r). \end{aligned}$$

Der folgende einfache Satz beschreibt einen Zusammenhang zwischen der operationalen und der denotationalen Semantik regulärer Programme: die denotationale Bedeutung eines Programms setzt sich aus den Speicheränderungen der Grundoperationen entsprechend den möglichen Programmabläufen zusammen.

**Satz 2.6** Zu jedem  $a \in \Sigma$  sei eine Relation  $\vdash_a \subseteq M \times M$  gegeben. Definiere auf der Algebra  $\mathcal{R} = (2^{M \times M}, \cup, 0_M, \circ, 1_M, *, \vdash_a)_{a \in \Sigma}$  zu jedem  $L \subseteq \Sigma^*$  eine Relation  $\vdash_L \subseteq M \times M$  durch

$$\vdash_L := \bigcup \{ \vdash_w \mid w \in L \}, \quad \text{wobei } \vdash_\varepsilon := 1_M, \quad \vdash_{va} := \vdash_v \circ \vdash_a.$$

Dann gilt  $\mathcal{R}(r) = \vdash_{L(r)}$  für jeden regulären Ausdruck  $r$  über  $\Sigma$ .

**Beweis:** Man sieht direkt, daß  $\mathcal{R}(0) = \vdash_{L(0)}$ ,  $\mathcal{R}(1) = \vdash_{L(1)}$  und  $\mathcal{R}(a) = \vdash_{L(a)}$ . Für zusammengesetzte Ausdrücke folgt die Behauptung aus folgenden leicht nachzurechnenden

<sup>1</sup>Beachte, daß wegen der nichtdeterministischen Fallunterscheidung in  $(r+s)$  und der nicht festgelegten Anzahl von Iterationen bei  $r^*$  verschiedene Abläufe desselben Programms möglich sind.

Eigenschaften:

$$\begin{aligned} \vdash_{(L_1 \cup L_2)} &= \vdash_{L_1} \cup \vdash_{L_2} \\ \vdash_{(L_1 \cdot L_2)} &= \vdash_{L_1} \circ \vdash_{L_2} \\ \vdash_{L^*} &= (\vdash_L)^* \end{aligned}$$

□

Da die Wahl von  $M$  und den Ausgangsrelationen  $\vdash_a$  beliebig war, zeigt der Satz, daß die Sprachinterpretation feiner ist als die Relationeninterpretation: *jede* Relationeninterpretation  $\mathcal{R}$  läßt sich aus der Sprachinterpretation  $\mathcal{L}$  und den Grundrelationen  $\mathcal{R}(a)$  durch  $\mathcal{R}(r) = \vdash_{L(r)}$  gewinnen.

**Folgerung 2.7** Aus  $\mathcal{L}(r) = \mathcal{L}(s)$  folgt  $\mathcal{R}(r) = \mathcal{R}(s)$ .

Die Umkehrung gilt natürlich nicht: zwei Programme können dieselbe Relation oder Funktion berechnen, ohne daß ihre möglichen Programmabläufe übereinstimmen.

Auch einzelne Programmabläufe  $v, w \in \Sigma^*$  können dieselbe Ein/-Ausgabebeziehung bestimmen, ohne daß die Abläufe gleich sind:  $\vdash_v = \vdash_w \not\Rightarrow v = w$ .

Interpretiert man reguläre Ausdrücke als Programme, so schreibt man oft  $(r; s)$  statt  $(r \cdot s)$  für die sequentielle Verkettung von Programmen.

**Bemerkung 2.8** Bei den Programmen wurde bisher vom Inhalt der Tests abstrahiert. Um Programme mit Tests und deterministische Programme zu erfassen, muß man die regulären Ausdrücke erweitern. Seien dazu neben den Basisaktionen  $\text{AKTION} = \Sigma$  eine endliche Menge  $\text{TEST} = \{b_1, \bar{b}_1, \dots, b_k, \bar{b}_k\}$  von Basistests gegeben. Verallgemeinerte reguläre Programme kann man dann durch folgende Definitionen einführen:<sup>2</sup>

$$\begin{aligned} \text{test} &:= && b_1 \mid \dots \mid b_k \mid \bar{b}_1 \mid \dots \mid \bar{b}_k \\ &&& \mid (\text{test} \vee \text{test}) \mid (\text{test} \wedge \text{test}) \\ \text{program} &:= && a_1 \mid \dots \mid a_n \mid \text{loop} \mid \text{skip} \\ &&& \mid (\text{test} \cdot \text{program}) \\ &&& \mid (\text{program} + \text{program}) \\ &&& \mid (\text{program} \cdot \text{program}) \\ &&& \mid \text{program}^* \end{aligned}$$

Dabei ist die neue Konstruktionsmöglichkeit als

$$(\text{test} \cdot \text{program}) := (\text{if } \text{test} \text{ then do } \text{program})$$

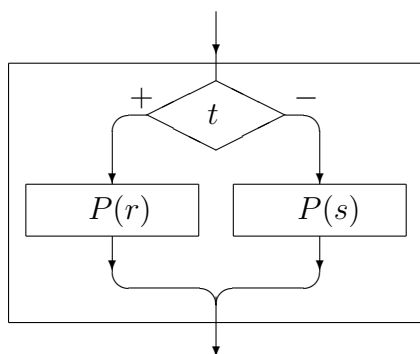
<sup>2</sup>Diese Kurzschreibweise soll heißen:  $\text{test}$  ist die kleinste Menge  $T$  mit (i)  $\{b_1, \dots, b_k, \bar{b}_1, \dots, \bar{b}_k\} \subseteq T$ , (ii) Für  $t \in T$  und  $s \in T$  ist auch  $(t \vee s) \in T$  und  $(t \wedge s) \in T$ . Entsprechend für die Programme.

zu verstehen. Für so zusammengesetzte  $t$  definiert man den komplementären Test  $\bar{t}$  durch

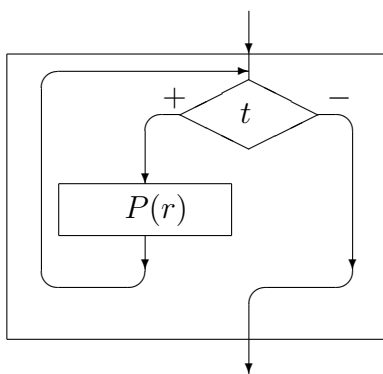
$$\overline{(t_1 \vee t_2)} := (\bar{t}_1 \wedge \bar{t}_2) \quad \text{und} \quad \overline{(t_1 \wedge t_2)} := (\bar{t}_1 \vee \bar{t}_2).$$

DETERMINISTISCHE (REGULÄRE) PROGRAMME sind dann diejenigen, bei denen die regulären Operationen  $+$  und  $*$  nur auf folgende Weisen verwendet werden:

$$P(t \cdot r + \bar{t} \cdot s) = (\text{if } t \text{ then do } r \text{ else do } s) =$$



$$P((t \cdot r)^* \cdot \bar{t}) = (\text{while } t \text{ do } r) =$$



Hierbei sind die Tests so zu verstehen:



Um die angegebene denotationelle Semantik auf Programme mit Tests auszudehnen, können wir wie folgt vorgehen. Sei  $\Delta = \{b_1, \dots, b_k, \bar{b}_1, \dots, \bar{b}_k\}$  eine zu  $\Sigma$  disjunkte Menge von atomaren Tests.

In der Relationeninterpretation ordnen wir jedem Test  $t$  die Menge  $Test(t) \subseteq M$  derjenigen Speicherzustände zu, die den Test  $t$  erfüllen. Das ist gleichwertig damit, daß den Tests die folgende Relation zugeordnet wird:

$$\vdash_t := 1_{Test(t)} = \{(m, m) \mid m \in Test(t)\} \subseteq M \times M.$$

Basistests  $b$  und ihre Komplementärtests  $\bar{b}$  sind dual zueinander zu interpretieren:

$$\vdash_{\bar{b}} := 1_M - \vdash_b = \{(m, m) \mid m \notin Test(b)\}.$$

Für die Sprachinterpretation ist zu beachten, daß es – im Unterschied zur Relationeninterpretation – keine Sprachen  $L$  mit  $\emptyset = L(0) \subset L \subset L(1) = \{\varepsilon\}$  gibt, die als  $L(t)$  in Frage kämen. Man kann die Programme aber als formale Sprachen  $L \subseteq (\Sigma \cup \Delta)^*$  interpretieren; dabei merkt man sich im Programmablauf  $w \in L(r) \subseteq (\Sigma \cup \Delta)^*$  auch, welche Basistests durchlaufen wurden.

## Aufgaben

**Aufgabe 2.1** Zeige für beliebige Sprachen  $L, L_1, L_2 \subseteq \Sigma^*$ :

- (a) Wenn  $L_1 \subseteq L_2$  ist, so ist auch  $L_1^* \subseteq L_2^*$ .
- (b)  $L^* = \bigcup_{n=0}^{\infty} L^n$ , wobei  $L^n$  durch  $L^0 := \{\varepsilon\}$  und  $L^{n+1} := L^n \cdot L$  definiert ist.
- (c)  $L \cdot L^* = L^* \cdot L$ .
- (d)  $(L^*)^* = L^*$ .

Zeige, daß diese Gleichungen auch für Relationen  $L \subseteq M \times M$  gelten.

**Aufgabe 2.2** Zeige durch Nachrechnen, daß folgende Gleichungen zwischen Sprachen wahr sind, für beliebige Sprachen  $L_1, L_2, L_3 \subseteq \Sigma^*$ .

- (a)  $(L_1 \cup L_2)^* = (L_1^* \cdot L_2^*)^*$ .
- (b)  $L_1(L_2 \cup L_3) = (L_1 \cdot L_2) \cup (L_1 \cdot L_3)$ .
- (c)  $\{\varepsilon\} \cup (L_1^* \cdot L_1^*) \cup L_1 = L_1^*$ .

Zeige, daß folgende Gleichungen bei der Sprachinterpretation nicht für alle regulären Ausdrücke  $r, s$  über  $\Sigma = \{a, b\}$  gelten:

- (a)  $(r + s)^* = r^* + s^*$ .  
 (b)  $((r \cdot s) + r)^* \cdot r = r \cdot ((r \cdot s) + r)^*$ .

Für *spezielle* Ausdrücke  $r$  und  $s$  können diese Gleichungen aber doch gelten. Gib auch dafür Beispiele an.

**Aufgabe 2.3** Zeige, daß die Klasse der regulären Sprachen unter Substitution abgeschlossen ist. Das heißt:

Ist  $R$  und für jedes  $a \in \Sigma$  auch  $L_a$  eine reguläre Sprache über  $\Sigma$ , so ist auch

$$L_R := \bigcup \{L_w \mid w \in R\} \quad \text{mit} \quad L_{va} := L_v \cdot L_a \text{ und } L_\varepsilon := \{\varepsilon\}$$

eine reguläre Sprache über  $\Sigma$ .

Hinweis: Benutze reguläre Ausdrücke  $r$  für  $R$  und  $r_a$  für  $L_a$ .

**Aufgabe 2.4** Gib einen regulären Ausdruck an, der die Sprache

$$\{w \in \{a, b, c\}^* \mid ac \text{ ist kein Teilwort von } w\}$$

beschreibt.

**Aufgabe 2.5** Sei  $M = \mathbb{N}^3$  die Menge aller Tripel  $(k, m, n)$  von natürlichen Zahlen. Man stelle sich vor, daß diese Zahlen im Datenspeicher einer Maschine unter den Adressen  $x$ ,  $y$ , und  $z$  gespeichert sind. Ist  $f : \mathbb{N}^3 \rightarrow \mathbb{N}$ , so hat die Zuweisung

$$\cdot \rightarrow \boxed{x \leftarrow f(x, y, z)} \rightarrow \cdot$$

als relationale Bedeutung die folgende Relation zwischen dem Speicherzustand vor und nach Ausführung der Zuweisung:

$$\boxed{x \leftarrow f(x, y, z)}^{R(M)} := \{((l, m, n), (f(l, m, n), m, n)) \mid l, m, n \in \mathbb{N}\}.$$

Betrachte solche Anweisungen  $(x \leftarrow f(x, y, z))$  als Elemente eines Alphabets  $\Sigma$  von atomaren Programmen.

- (a) Seien  $a := (y \leftarrow x + z)$  und  $b := (x \leftarrow x - 1)$  atomare Programme. Die Bedeutung der Differenz sei für  $n > k$  durch  $k \dot{-} n := 0$  festgelegt. Berechne die relationale Bedeutung  $(a; b)^{R(M)}$  des zusammengesetzten Programms

$$\cdot \rightarrow \boxed{y \leftarrow x + z} \rightarrow \cdot \rightarrow \boxed{x \leftarrow x - 1} \rightarrow \cdot$$

- (b) Nun berechne die relationale Bedeutung des Programms

$$(\text{while } x > 0 \text{ do } ((y \leftarrow x + z); (x \leftarrow x - 1)))$$

in zwei Schritten.

Berechne zuerst die relationale Bedeutung  $(r^*)^{R(M)}$  des regulären Ausdrucks

$$r^* := ((y \leftarrow x + z); (x \leftarrow x - 1))^*.$$

Die Ausführung eines Tests wie  $(x > 0?)$  ändert den gegenwärtigen Zustand nicht, sondern verhindert weitere Zustandsübergänge, falls die Bedingung nicht erfüllt ist. Das wird durch Teilrelationen der Identität auf  $M$  wie folgt modelliert. Für den Test  $(x > 0?)$  auf die Eigenschaft

$$B := \{(k, m, n) \mid k, m, n \in \mathbb{N}, k > 0\}$$

(von Zuständen!) benutze die Relationen

$$\begin{aligned} 1_B &:= \{(k, m, n), (k, m, n) \mid k, m, n \in \mathbb{N}, k > 0\}, \\ 1_{\overline{B}} &:= \{(k, m, n), (k, m, n) \mid k, m, n \in \mathbb{N}, k = 0\} \end{aligned}$$

und interpretiere dann das angegebene *while*-Programm über

$$(\text{while } B \text{ do } r)^{R(M)} := ((1_B; r)^*; 1_{\overline{B}})^{R(M)}.$$

**Aufgabe 2.6** Definiere zu jedem regulären Ausdruck  $r$  über  $\Sigma$  und jedem Buchstaben  $a \in \Sigma$  rekursiv über den Aufbau von  $r$

- (a) einen regulären Ausdruck  $(r - 1)$  mit  $L((r - 1)) = L(r) - \{\varepsilon\}$ ,  
 (b) einen regulären Ausdruck  $(r - a)$  mit  $L((r - a)) = L(r) - \{a\}$ .