Gliederung des Kapitels 1.1

1.1 Algorithmen und Sprachen

~~1.1.1 Darstellung von Algorithmen~~
~~1.1.2 Grundlegende Datenbereiche~~
~~1.1.3 Realisierte Abbildung~~
1.1.4 (Künstliche) Sprachen
1.1.5 Grammatiken
1.1.6 BNF, Syntaxdiagramme
1.1.7 Sprachen zur Beschreibung von Sprachen
1.1.8 Übungsaufgaben

Abschnitt 1.1.4 soll Ihnen folgende Inhalte vermitteln:

Ein Programm ist eine Folge von Zeichen, die gewissen Regeln gehorchen müssen. Eine Menge von Zeichenfolgen (Wörtern), die nach bestimmten Regeln aufgebaut sind, nennt man (formale) Sprache. Programme bilden daher eine Sprache, nämlich die "Programmiersprache".

Sie lernen die für "Sprachen" und einige ihrer Operationen erforderlichen Definitionen und ein paar typische Beispiele.

1.1.4 (Künstliche) Sprachen

Wir haben Algorithmen mit Hilfe von Programmen beschrieben. Jetzt wollen wir den zulässigen Aufbau von Programmen mit Hilfe eines Regelsystems beschreiben.

Programme bestehen aus einer Folge von Zeichen. Diese Zeichen sind:

(1) Die Elemente von **Â**, die auf der Tastatur zu finden sind.

(2) Besondere Wörter ("Schlüsselwörter" der Programme) und zwar bisher die Elemente folgender Menge

SW = {program, is, begin, end, var, if, then, else, fi, while, do, od, for, to, downto, repeat, until, read, write, natural, Boolean, integer, real, character, array, of, skip, halt, true, false, not, and, or, div, mod}.

(Man erkennt sie u.a. daran, dass sie unterstrichen werden.)

Ein Programm ist also eine Zeichenfolge, wobei die Zeichen aus dem "Alphabet" $\hat{A} \cup SW$ stammen. Aber nicht jede Zeichenfolge ist erlaubt, sondern nur ganz bestimmte. Diese werden durch "Regeln" festgelegt, die wir in Abschnitt 1.1.5 betrachten werden.

Hier untersuchen wir zunächst einen sehr einfachen Fall, nämlich den Aufbau von "Bezeichnern". Bezeichner sind Zeichenketten über Buchstaben, Ziffern und dem Unterstrich "_". Seien also

$\Phi_B = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$

die Menge der Buchstaben und

$\Phi_Z = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ die Menge der Ziffern und

$\Phi = \Phi_B \cup \Phi_Z \cup \{ _ \}$ die Menge der 63 Zeichen, die für einen Bezeichner verwendet werden dürfen.

Definition 1.1.4.1 (umgangssprachlich): Ein Bezeichner ist eine Folge von Elementen aus Φ, die mit einem Buchstaben (d.h. mit einem Element aus Φ_B) beginnt und dann endlich viele Zeichen aus Φ besitzen darf.

Es folgt eine formalere Definition der Bezeichner.

1.1.4.2: *Induktive Definition:*

(1) Jedes Element aus Φ_B ist ein Bezeichner.

(2) Wenn w ein Bezeicher ist und $a \in \Phi$, dann ist auch w a ein Bezeichner.

(3) Nur Zeichenfolgen, die ausschließlich mit den Regeln (1) und (2) aufgebaut wurden, sind Bezeichner.

Zum Beispiel ist H z 4 ein Bezeichner.

1.1.4.3: In unserem einfachen Fall kann man die Menge der Bezeichner Bez auch direkt angeben:

$Bez = \{w \in \Phi^* \mid w = bv,\ b \in \Phi_B \text{ und } w \in \Phi^*\} = \Phi_B \Phi^*$.

Man kann die Menge der Bezeichner auch erzeugen lassen:

1.1.4.4: Es sei β ein neues Zeichen. Betrachte folgende Regeln:

$\beta \rightarrow \gamma$ für jedes $\gamma \in \Phi_B$,

$\beta \rightarrow \beta\alpha$ für jedes $\alpha \in \Phi$.

Eine Herleitung erfolgt, indem man mit dem Zeichen β beginnt und ein Zeichen, das links von "$\rightarrow$" in der Regel steht, durch die davon rechts stehende Zeichenfolge ersetzen darf. (In unserem Fall haben nur das eine Zeichen β, das ersetzt werden darf.) Man muss solange ableiten, bis das Zeichen β nicht mehr auftritt.

Bez ist dann die Menge der Zeichenfolgen, die man aus β in endlich vielen Schritten herleiten (oder ableiten) kann.

Zum Beispiel kann man H z 4 wie folgt herleiten:

Beginne mit β und wende die Regel $\beta \rightarrow \beta\,\alpha$ speziell für α=4 an:

$\beta \rightarrow \beta\,4$ Wende die Regel erneut für α=z an:

$\beta \rightarrow \beta\,4 \rightarrow \beta\,z\,4$ Wende nun die Regel $\beta \rightarrow \gamma$ mit γ=H an:

$\beta \rightarrow \beta\,4 \rightarrow \beta\,z\,4 \rightarrow H\,z\,4$

Da man H z 4 auf diese Weise herleiten konnte, ist diese Zeichenfolge also ein Bezeichner.

Allgemein ist
Bez = {w∈Φ* | Es gibt eine Herleitung von β nach w}

Auf die Regeln und die Herleitung von Zeichenfolgen aus speziellen Zeichen, die im Laufe einer Herleitung verschwinden müssen, gehen wir in Abschnitt 1.1.5 ein.

Wir halten fest: Die uns interessierenden Mengen ("Sprachen") bestehen aus Zeichenfolgen, die über einer festen Menge (z.B. Φ oder Â ∪ SW) gebildet werden. Wir definieren daher:

Definition 1.1.4.5: Eine endliche Menge $A = \{a_1, a_2, ..., a_n\}$, deren Elemente linear angeordnet sind ($a_1 < a_2 < ... < a_n$), heißt ein (endliches) Alphabet.
Jede Menge von Zeichenfolgen über A bezeichnen wir als Sprache über dem Alphabet A (engl.: "language").
D.h.: L ist eine Sprache über A genau dann, wenn $L \subseteq A^*$.

1.1.4.6 Operationen: Obige Definition hat den Vorteil, dass wir alle Operationen, die wir für Mengen und für Wortmengen kennen, auch für Sprachen nutzen können. Zum Beispiel:

Vereinigung von Sprachen: Wenn L_1 und L_2 Sprachen über dem gleichen Alphabet A sind, dann ist auch $L_1 \cup L_2$ Sprache über A.

Durchschnitt von Sprachen: Wenn L_1 und L_2 Sprachen über dem gleichen Alphabet A sind, dann ist auch $L_1 \cap L_2$ Sprache über A.

Komplement von Sprachen: Wenn L eine Sprache über A ist, dann ist auch das Komplement $A^*\backslash L = \overline{L}$ eine Sprache über A.

Konkatenation ("Verkettung") von Sprachen: Sind L_1 und L_2 Sprachen über A, dann ist auch ihre Konkatenation $L_1 \circ L_2 = \{u\,v \mid u \in L_1, v \in L_2\}$ eine Sprache über A.
Man schreibt wie bei Wörtern kurz L_1L_2 anstelle von $L_1 \circ L_2$.

1.1.4.6 Operationen (Fortsetzung)

Iteration von Sprachen: Wenn L eine Sprache über A ist, dann ist auch $L \circ L = LL$, also die Konkatenation von L mit sich, eine Sprache über A. Seien $L^0 = \{\varepsilon\}$ und L^i die i-fache Konkatenation von L mit sich, dann sei L^* die Vereinigung aller dieser Mengen L^i für $i \geq 0$ (bzw. L^+ für $i > 0$). *Formale Definition:*

$L^0 = \{\varepsilon\}$ *Beachte:* $L\{\varepsilon\} = L$, also $LL^0 = L^1 = L$.

$L^{i+1} = LL^i$ für alle $i \geq 0$

Die Iterierte von L
oder
*das von L erzeugte Untermonoid in A**

$$L^* = \bigcup_{i \geq 0} L^i = L^0 \cup L^1 \cup L^2 \cup L^3 \cup L^4 \ldots.$$

$$L^+ = \bigcup_{i \geq 1} L^i = L^1 \cup L^2 \cup L^3 \cup L^4 \ldots.$$

*Die von L erzeugte Unterhalbgruppe in A**

Es gilt: $L^* = L^+ \cup \{\varepsilon\}$.

Beispiele: Sei A = {0, 1} das zugrunde liegende Alphabet.

Sei K = {0}, dann ist $K^2 = KK = \{00\}$, $K^3 = \{000\}$ usw.,

$K^* = K^0 \cup K^1 \cup K^2 \cup K^3 \cup ... = \{\varepsilon, 0, 00, 000, ...\}$

$= \{0^i \mid i \geq 0\}$ und

$K^+ = \{0, 00, 000, ...\} = \{0^i \mid i > 0\}$.

Sei L = {ε, 0, 01}, dann ist $L^2 = \{\varepsilon, 0, 00, 01, 001, 010, 0101\}$,

$L^3 = \{\varepsilon, 0, 00, 01, 000, 001, 010, 0001, 0010, 0100, 0101,$
$00101, 01001, 01010, 010101\}$ usw.,

$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup ... =$?

Das lässt sich nicht mehr einfach hinschreiben! Jedes Wort aus L* muss man als Folge von Wörtern aus L darstellen können.

Gehört zum Beispiel 0100101000010 zu L*?

Ja, wegen der Zerlegung 01 0 01 01 0 0 0 01 0 ∈ L*

Beispiel 1.1.4.7: Alphabet A = {0, 1, 2}. Sei L die Menge aller Wörter über A, in denen kein von 0 verschiedenes Zeichen vor einer 0 und kein von 2 verschiedenes Zeichen nach einer 2 im Wort stehen darf.

Skizze, wie solche Wörter aussehen:

$a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8\ a_9\ a_{10}\ a_{11}\ a_{12}\ a_{13}\ a_{14}\ a_{15}\ a_{16}\ a_{17}$ ..... a_n =

.....0.....2.....

Hier dürfen nur Nullen stehen

Hier dürfen nur Zweien stehen

Beispiel 1.1.4.7: Alphabet A = {0, 1, 2}. Sei L die Menge aller Wörter über A, in denen kein von 0 verschiedenes Zeichen vor einer 0 und kein von 2 verschiedenes Zeichen nach einer 2 im Wort stehen darf.

Skizze, wie solche Wörter aussehen:

$a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8\ a_9\ a_{10}\ a_{11}\ a_{12}\ a_{13}\ a_{14}\ a_{15}\ a_{16}\ a_{17}$ ..... a_n =

.....0.....0.....2.....

Hier dürfen nur Nullen stehen

Hier dürfen nur Zweien stehen

Beispiel 1.1.4.7: Alphabet A = {0, 1, 2}. Sei L die Menge aller Wörter über A, in denen kein von 0 verschiedenes Zeichen vor einer 0 und kein von 2 verschiedenes Zeichen nach einer 2 im Wort stehen darf.

Skizze, wie solche Wörter aussehen:

$a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8\ a_9\ a_{10}\ a_{11}\ a_{12}\ a_{13}\ a_{14}\ a_{15}\ a_{16}\ a_{17}$ ..... a_n =

.....0.....0.....2.....2.....

Hier dürfen nur Nullen stehen

Hier dürfen nur Zweien stehen

Folglich können diese Wörter nur die Form
000...00111....11222.....2 = $0^i1^j2^k$ mit i, j, k ≥ 0 besitzen.

Beispiel 1.1.4.7: Alphabet A = {0, 1, 2}. Sei L die Menge aller Wörter über A, in denen kein von 0 verschiedenes Zeichen vor einer 0 und kein von 2 verschiedenes Zeichen nach einer 2 im Wort stehen darf.

Es gilt also $L = \{0^i 1^j 2^k \mid i, j, k \geq 0\} \subset A^*$.

Mit Hilfe unserer Operationen kann man L auch wie folgt beschreiben:

$L = \{1\}^* \{2\}^* \{3\}^*$.

Möchte man zusätzlich, dass die Zahl der Nullen und der Zweien gleich sein soll, so erhält man die Sprache
$L' = \{0^i 1^j 2^k \mid i, j, k \geq 0 \text{ mit } i = k\} \subset A^*$.

Weitere Beispiele in den Übungen.