# Markov-Algorithmen

Indem man die Reihenfolge und gewisse syntaktische Eigenschaften der Substitutionen eines Wortersetzungssystems festlegt, kann man Algorithmen erhalten.

---

Ein Markov-Algorithmus ist ein Wortersetzungssystem über einem gewissen Alphabet $A$, wobei die Subsitutionen in einer Reihenfolge $R_1, \ldots, R_n$ gegeben sind, und von der Form $l \to r$ oder auch der Form $l \to \cdot r$ sein dürfen. Im zweiten Fall sagen wir die Substitution ist terminierend. Wir schreiben $R_i = l_i \to r_i$ wo $r_i$ möglicherweise mit "$\cdot$" beginnt.

Sei ein Eingabewort gegeben. Einen Schritt des Algorithmus durchzuführen bedeutet die erste anwendbare Regel an der (von links nach rechts) ersten möglichen Stelle anzuwenden. Wenn keine Regel anwendbar ist, sagt man dass der Algorithmus mit dem Ausgangswort $w$ terminiert. Die Regel $R_i$ ist auf $w$ anwendbar, bedeutet dass $w = xl_iy$ für gewisse Worte $x, y$. Wenn überhaupt eine Regel auf $w$ anwendbar ist, sei $k$ die kleinste Zahl, sodass $R_k$ anwendbar ist. Weiters sei $x$ das kürzeste Wort, sodass $w = xl_ky$ mit einem gewissem Wort $y$. Dann ist ein Algorithmusschritt das Wort $w$ in das Wort $xr_ky$ zu transformieren, und wir schreiben $w \overset{k}{\to} u$. Der Index $k$ am Pfeil dient dabei nur der Lesbarkeit, um im Auge zu behalten welche Regel angewendet wurde. Ist $R_k$ terminierend, dann terminiert der Algorithmus mit Ausgangswort $u$. Wenn $R_k$ nicht terminierend ist, fährt der Algorithmus mit dem Wort $u$ fort.

---

Hier ist ein Beispiel eines Markov-Algorithmus ($\varepsilon$ bezeichnet dabei das leere Wort):

$$
\begin{aligned}
1.\ 00 &\to \varepsilon \\
2.\ 01 &\to \varepsilon \\
3.\ 10 &\to \varepsilon \\
4.\ 11 &\to \varepsilon \\
5.\ 0 &\to \cdot 1 \\
6.\ 1 &\to \cdot 1 \\
7.\ \varepsilon &\to \cdot 0
\end{aligned}
$$

Beispiel einer Anwendung des Algorithmus ist: $11001 \overset{1}{\to} 111 \overset{4}{\to} 1 \overset{6}{\to} \cdot 1$
Was gibt dieser Algorithmus aus?

---

Markov-Algorithmen sind Turing-vollständig. Intuitiv bedeutet dies, dass alle Probleme die mit einem Algorithmus lösbar sind, schon mit einem Markov-Algorithmus lösbar sind.

**Anmerkung:** Es kann dabei sein dass man das Alphabet geeignet erweitern muss (auch in den folgenden Aufgaben).

**Aufgabe 1** Finde einen Markov-Algorithmus, der:

- für eine gegebene natürliche Zahl, also ein Wort im Alphabet $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, die kleinste natürliche Zahl ausgibt, die sich mit den gleichen Ziffern formen läßt (wobei Nullen am Beginn des Wortes weggelassen werden). Hat man also zum Beispiel 109283 als Eingabe, so soll der Algorithmus 12389 ausgeben.
- für eine gegebene natürliche Zahl, also ein Wort im Alphabet $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, die größte natürliche Zahl ausgibt, die sich mit den gleichen Ziffern formen läßt. Hat man also zum Beispiel 109283 als Eingabe, so soll der Algorithmus 983210 ausgeben.

$\triangleleft$

**Aufgabe 2**

Finde einen Markov-Algorithmus der zu zwei natürlichen Zahlen $n$ und $m$ die in unärer Notation gegeben sind (also repräsentiert durch die entsprechende Anzahl von |-en), und durch einen "$*$" getrennt sind, die Zahl $n + m$, ebenso in unärer Notation, ausgibt. Zum Beispiel soll also bei der Eingabe $||| * |||||$ das Wort $||||||||$ ausgegeben werden. $\triangleleft$

**Aufgabe 3**

Finde einen Markov-Algorithmus der zu zwei natürlichen Zahlen $n$ und $m$ die in unärer Notation gegeben sind (also repräsentiert durch die entsprechende Anzahl von |-en), und durch einen "$*$" getrennt sind, die Zahl $2n + m$, ebenso in unärer Notation, ausgibt. Zum Beispiel soll also bei der Eingabe $||| * |||||$ das Wort $||||||||||$ ausgegeben werden. $\triangleleft$

**Aufgabe 4*** Finde einen Markov-Algorithmus der zu einer natürlichen Zahl die in Binärdarstellung gegeben ist die unäre Darstellung der Zahl ausgibt. Zum Beispiel soll also auf die Eingabe 1011 die Ausgabe $||||||||||$ erfolgen. $\triangleleft$