Prof. Dr. Hans-Jörg Kreowski **Juni 2007**

Formale Sprachen: DNA Computing

Sticker-Systeme: Die Twin-Shuffle-Sprache

Das folgende Sticker-System erzeugt eine Variante der Twin-Shuffle-Sprache, die beim ersten Betrachten merkwürdig erscheinen mag. Mit dieser Sprache hat es aber eine besondere Bewandtnis, die sie für DNA Computing außerordentlich interessant macht.

$$\gamma_{TS} = (V, \Delta V, A_{TS}, D_{TS})$$

mit $V = U \cup \overline{U} \cup U'$ für ein beliebiges, aber nicht leeres Alphabet U, $A_{TS} = \{\begin{bmatrix} a'_0 \\ a'_0 \end{bmatrix}\}$ für ein beliebiges, aber fest gewähltes $a_0 \in U$ und $D_{TS} = \{(\binom{a'}{\lambda}, \begin{bmatrix} a \\ a \end{bmatrix}), (\binom{\lambda}{a'}, \begin{bmatrix} \overline{a} \\ \overline{a} \end{bmatrix}) \mid a \in U\}$. Dabei sind $\overline{U}$ und U' Kopien von U, derart dass zu jedem $a \in U$ genau eine Kopie $\overline{a} \in \overline{U}$ und eine Kopie $a' \in U'$ existieren.

Eine typische Berechnung in γ_{TS} sieht so aus:

$$\begin{bmatrix} a'_0 \\ a'_0 \end{bmatrix} \Longrightarrow {}^{a'_1}\begin{bmatrix} a'_0\ a_1 \\ a'_0\ a_1 \end{bmatrix} \Longrightarrow \begin{bmatrix} a'_1\ a'_0\ a_1\ \overline{a_1} \\ a'_1\ a'_0\ a_1\ \overline{a_1} \end{bmatrix} \Longrightarrow {}^{a'_2}\begin{bmatrix} a'_1\ a'_0\ a_1\ \overline{a_1}\ \overline{a_2} \\ a'_1\ a'_0\ a_1\ \overline{a_1}\ \overline{a_2} \end{bmatrix} \Longrightarrow {}^{a'_2a'_2}\begin{bmatrix} a'_1\ a'_0\ a_1\ \overline{a_1}\ \overline{a_2}\ \overline{a_0} \\ a'_1\ a'_0\ a_1\ \overline{a_1}\ \overline{a_2}\ \overline{a_0} \end{bmatrix} \Longrightarrow$$

$${}^{a'_0}\begin{bmatrix} a'_2\ a'_1\ a'_0\ a_1\ \overline{a_1}\ \overline{a_2}\ \overline{a_0}\ a_2 \\ a'_2\ a'_1\ a'_0\ a_1\ \overline{a_1}\ \overline{a_2}\ \overline{a_0}\ a_2 \end{bmatrix} \Longrightarrow \begin{bmatrix} a'_0\ a'_2\ a'_1\ a'_0\ a'_1\ \overline{a_1}\ \overline{a_2}\ a_2\ a_0 \\ a'_0\ a'_2\ a'_1\ a'_0\ a'_1\ \overline{a_1}\ \overline{a_2}\ a_2\ a_0 \end{bmatrix}$$

Allgemein werden die a' links oben immer gleichzeitig mit den a rechts angefügt, so dass sie spiegelbildlich sind, wenn man die Striche weglässt. Entsprechend werden die a' links unten spiegelbildlich zu den $\overline{a}$ rechts angefügt. Da die a' links beliebig oben und unten angefügt werden können, werden die a und $\overline{a}$ gemischt. Allerdings müssen die a' links oben und unten an derselben Stelle gleich sein, so dass die Reihenfolge der a einerseits und der $\overline{a}$ andererseits auch gleich sein muss. Zusammengefasst erzeugt γ_{TS} folgende Sprache:

$$L(\gamma_{TS}) = \{x'a'_0w \mid x \in U^*, w \in (U \cup \overline{U})^*, read_U(w) = trans(x) = read_{\overline{U}}(w)\}.$$

Dabei ist $trans$ die bekannte Transposition (Spiegelung) mit $trans(\lambda) = \lambda$ und $trans(au) = trans(u)a$. Die Operation $read_U$ löscht aus den Wörtern aus $U \cup \overline{U}$ alle Symbole aus $\overline{U}$, d.h. $read_U(\lambda) = \lambda, read_U(au) = a\ read_U(w)$ und $read_U(\overline{a}u) = read_U(w)$. Die Operation $read_{\overline{U}}$ macht dasselbe mit den Symbolen aus U, löscht zusätzlich aber noch den Querstrich, d.h. $read_{\overline{U}}(\lambda) = \lambda, read_{\overline{U}}(au) = read_{\overline{U}}(u)$ und $read_{\overline{U}}(\overline{a}u) = a\ read_{\overline{U}}(u)$. Schließlich ist x' für $x \in U^*$ das Wort, das aus x entsteht, wenn alle Symbole einen Strich erhalten, d.h. $\lambda' = \lambda$ und $(au)' = a'u'$.

Die eigentliche Twin-Shuffle-Sprache ist der Teil rechts von a'_0 :

$$TS_U = \{w \in (U \cup \overline{U})^* \mid read_U(w) = read_{\overline{U}}(w)\}.$$

Das gestrichene Spiegelbild links von a_0' und a_0' selbst dienen nur der Erzeugung.

Das Besondere an der Twin-Shuffle-Sprache ist ihre Verwandlungsfähigkeit. Sie lässt sich nämlich sogar schon für $U = \{0, 1\}$ mit Hilfe eines einfachen Konzepts in jede beliebige rekursiv aufzählbare Sprache transformieren. Das Konzept, das dafür verwendet werden kann, sind endliche Automaten mit Ausgabe, sogenannte verallgemeinerte endliche Automaten (generalized sequential machines, sequential transducers). Das sind normale endliche Automaten, die aber bei jedem Zustandsübergang noch ein Ausgabewort produzieren. Sammelt man die Ausgabewörter während der Erkennung eines Eingabewortes auf, werden Eingabewörter übersetzt. Da man das für beliebige Eingabewörter machen kann, wird insgesamt jede Sprache von Eingabewörtern übersetzt in eine Sprache von Wörtern über dem Ausgabealphabet. Eingabewörter, die nicht erkannt werden, liefern aber keinen Beitrag zur resultierenden Sprache. Sei gsm ein verallgemeinerter endlicher Automat, und L eine Sprache von Eingabewörtern, dann wird die transformierte Sprache mit $gsm(L)$ bezeichnet:

Dann gilt folgendes

Theorem
Sei L eine rekursiv aufzählbare Sprache. Dann existiert ein verallgemeinerter endlicher Automat gsm mit $gsm(TS_{\{0,1\}}) = L$.

Da sich bei Wörtern über $V = U \cup \overline{U} \cup U'$ die gestrichenen Symbole durch einen ganz einfachen verallgemeinerten endlichen Automaten löschen lassen, gilt auch:

Theorem
Sei n eine rekursiv aufzählbare Sprache. Dann existiert ein verallgemeinerter endlicher Automat gsm mit $gsm(\gamma_{TS}) = L$.

Bemerkenswert daran ist, das ein einziges Sticker-System, das ja eine spezielle Form des DNA Computing verkörpert, alles Berechenbare liefert. Die dazu notwendige Transformation lässt sich durch ein einfaches Konstrukt bewerkstelligen, nämlich durch einen endlichen Automaten mit Ausgabe. Programmieren wird also ganz leicht: ein bisschen visuell Modellieren und schon fertig. Oder wo ist der Haken?