1 Introduction

Sticker system is a formal model based on sticker operations, which is an abstraction of the Watson-Crick complementarity. The basic operation of a sticker system is the Sticking operation, which uses the DNA strands called axioms and dominoes that are DNA strands with sticky ends [1]. By using the sticking operations, complete double stranded sequences can be obtained. Rozenberg [3] in his paper explained one concrete method to transform an automaton to a sticker system. In this paper we introduce more simple efficient transformation from an automaton to a sticker system and implement it using Haskell module functions. We improve the insufficient results in [3]. We modify the expression of domino and sticker operations for realizing them by Haskell functions. We denoted the set of all dominoes over \(\Sigma^*\times \Sigma^*\times Z\) as a domino over \((\Sigma,\rho)\), if following conditions hold:

- If \(x \geq 0\) then \((\lfloor i + x \rfloor, r[i]) \in \rho\),
- If \(x < 0\) then \((\lfloor i \rfloor, r[i - 1]) \in \rho\).

We denote the set of all dominoes over \((\Sigma,\rho)\) by \(D\).

3.1 Sticker Operation

Definition 6 Sticker operation \(\mu : D \times D \rightarrow D \cup \{\bot\}\) is defined as follows:

\[
\mu((l_1, r_1, x_1), (l_2, r_2, x_2)) = \begin{cases} 
(l_1 l_2, r_1 r_2, x_1) & \text{(if \(^*)\)} \\
\bot & \text{(otherwise)}
\end{cases}
\]

(*) \((l_1 l_2, r_1 r_2, x_1) \in D\) and \(x_1 + \text{length}(r_1) - \text{length}(l_1) = x_2\).

3.2 Sticker System

Definition 7 Sticker system \(\gamma = (\Sigma, \rho, A, R)\) of the alphabet set \(\Sigma\), \(\rho \subseteq \Sigma \times \Sigma\), the finite set of axioms \(A \subseteq D\) and the finite set of dominoes \(R \subseteq D \times D\).

3.3 Sticker System vs Automaton

For a finite automaton \(M = (Q, \Sigma, \delta, q_0, F_M)\), the sticker system \(\gamma_M = (\Sigma, \rho, A, R)\) is defined similar to [3]. However we have modified the definition of domino \(F\) as

\[
F = \bigcup_{i=1}^{k+1} \left( \bigcup_{j=1}^{k+1} \{(\lambda, \lambda, 0), (x, vx, -|v|) | v \in \Sigma^j, x \in \Sigma^*, \delta(j-1, x) \in F_M\} \right)
\]

\[
k = |Q| - 1
\]

Theorem 1 ([3]) Let \(M\) be an automaton. Then \(L(\gamma_M) = L(M)\).
Example 2 Consider the following words generated by $L(M_1)$ in Example 1.

$\text{bbbbb}$ and $\text{abbab}$ are generated by $\gamma_M$ defined in our paper. But it is impossible to generate using the definition of $A$ and $F$ in [3]. $\text{bbbbb}$ is generated by $\text{bb}$ and $\text{abbab}$ is generated by $\text{ab}$ and $\text{bab}$.

Not only above words but also all the words with length 5 accepted by the automata defined in Example 1 can be generated by our system while it is impossible by the system defined in [3].

4 Grammar Module
Definition 8 (Grammar) A grammar is a four tuple $G = (T, N, R, S)$ of terminal symbols $T$, non-terminal symbols $N$, transformation rules $R$ and a start symbol $S$.

4.1 Sticker System vs Linear Grammar
Definition 9 For a linear grammar $G = (\Sigma, N, P, S)$, sticker system $\gamma_G = (\sigma, \rho, A, R)$ is defined similar to [3]. However we define dominoes $R$ as follows.

$$R = R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5 \cup R_6$$

$$R_1 = \bigcup_{i=1}^{k} \bigcup_{l=0}^{k+1} \{(ux, xv, |u|, (z, z, 0)) | (w, j, z) \in T(i, k + 1, l), u = Left(w, i), x = Right(w, i), v \in \Sigma^j \}$$

$$R_2 = \bigcup_{i=1}^{k} \bigcup_{l=0}^{k+1} \{(x, xv, 0, (zu, z, 0)) | (x, j, w) \in T(i, l, k + 1, z), z = Left(w, k + 1, i), u = Right(w, i), v \in \Sigma^j \}$$

$$R_3 = \bigcup_{l=1}^{k} \{(x, xv, 0, (zu, z, 0)) | (x, j, z) \in T(0, l, m), 0 \leq m \leq 2k + 1 - l, v \in \Sigma^j \}$$

$$R_4 = \bigcup_{i=1}^{k} \bigcup_{l=0}^{k+1} \{(z, z, 0, (zu, vx, -|v|)) | (z, j, w) \in T(i, l, k + 1, x = Left(w, k + 1, i), u = Right(w, i), v \in \Sigma^j \}$$

$$R_5 = \bigcup_{l=1}^{k} \bigcup_{l=0}^{k} \{(ux, z, |u|, (x, vx, -|v|)) | (x, j, z) \in T(i, k + 1, l), u = Left(w, i), x = Right(w, k + 1, i), v \in \Sigma^j \}$$

$$R_6 = \bigcup_{l=1}^{k} \bigcup_{l=0}^{k} \{(z, z, 0, (x, vx, -|v|)) | (z, j, x) \in T(0, m, l), 0 \leq m \leq 2k + 1 - l, v \in \Sigma^j \}$$

5 Compare our Haskell module with HaLex
HaLex library enables us to model, manipulate and animate regular languages [4]. It also introduces a number of Haskell data types and functions that manipulate them. Though Haskell has descriptions for infinite set HaLex doesn’t have such a property. In Haskell infinite set $\{a, b\}^*$ is denoted by finite length of expression $(\text{ststr}['a', 'b'])$. Function $(\text{ststr})$ computes the infinite set $\Sigma^*$ over $\Sigma$. Using $(\text{take})$ function to view contents of an infinite set.

References