Implementation of Haskell modules for automata and Sticker systems

K.K.K.R.Perera and Yoshihiro Mizoguchi

Received on March 11, 2009 / Revised on March 17, 2009

Abstract.
We realized operations appeared in the theory of automata using Haskell languages. Using the benefits of functions of lazy evaluations in Haskell, we can express a language set which contains infinite elements as concrete functional notations like mathematical notations. Our modules can be used not only for analyzing the properties about automata and their application systems but also for self study materials or a tutorial to learn automata, grammar and language theories. We also implemented the modules for sticker systems. Paun and Rozenberg explained a concrete method to transform an automaton to a sticker system in 1998. We modified their definitions and improved their insufficient results. Using our module functions, we can easily define finite automata and linear grammars and construct sticker systems which have the same power of finite automata and linear grammars.

Keywords. Automata, Language, Sticker System, DNA Computing, Haskell

1. Introduction

The sticker system is a formal model based on sticking operations, which is an abstraction of the Watson-Crick complementarity. We use the term domino to represent double stranded DNA sequences with sticky ends. By using the sticking operator, dominoes can be annealed and formed a complete double stranded sequence. Paun and Rozenberg [3] explained a concrete method to transform automata to sticker systems. In this paper we are trying to introduce simple efficient transformation and implement it using Haskell module functions. We also indicate and improve the insufficient results in [3]. We modify the expression of dominoes and the sticking operator for realizing Haskell functions. We change the definition of a domino \((D)\) from a string of pairs of alphabet to a triple \((l; r; x)\) of two string \(l\), \(r\) and an integer \(x\). For example \(\begin{array}{c}
\lambda \\
\lambda \\
AT \\
GC \\
\end{array}
\) in [3] is represented as \((ATGC; CTA; −1)\). According to this modification, the definition of sticking operator has been reformulated.

One of the benefits of using Haskell language is that it has descriptions for infinite set of strings using lazy evaluation schemes. For example, the infinite set \(\{a, b\}^*\) is denoted by finite length of expression \(\text{ssstar} [\{'a'\}, \{'b'\}]\). We use the \text{take} function to view contents of an infinite set (e.g. \(\text{take} 5 (\text{ssstar} [\{'a'\}, \{'b'\}]) = \{\text{aa}, \text{ab}, \text{ba}, \text{aa}, \text{bb}\}\)). Further using set theoretical notions in Haskell, we can easily realize the definitions of various kinds of set of dominoes. For example, to make a sticker system which generates the equivalent language of a finite automaton, we need an atom set

\[A_2 = \bigcup_{i=1}^{k+1} \{(xu, x, 0) | x \in \Sigma^*, u \in \Sigma^*, |xu| = k + 2, |u| = i, \delta^*(0, xu) = i - 1\}.
\]

In Haskell notations, we have following function definitions.

\[
aA2::\text{Automaton} \rightarrow [\text{Domino}]
\]
\[
aA2 m@(q, s, d, q0, f) = \text{concat} [(aA2' m i)| i<- [1..(k+1)]]
\]
\[
\text{where } k = (\text{length q})-1
\]
\[
aA2'::\text{Automaton} \rightarrow \text{Int} \rightarrow [\text{Domino}]
\]
\[
aA2' m@(q, s, d, q0, f) i = [(x++u, x, 0)| (x,u) <- \text{xpair}, (dstar d 0 (x++u)) ==(i-1)]
\]
\[
\text{where xpair} = [(x,u)|x<- (\text{sigman} s (k+2-i)), u<- (\text{sigman} s i)]
\]
\[
k = (\text{length q})-1
\]

The precise definition of the generated sticker system is described in Section 3. We prove that the generated languages are equal by using our formulations.

The Haskell module can be downloaded from our homepage\(^1\).

2. Automaton Module

Let \(\Sigma\) is an alphabet and \(\Sigma^*\) is the set of all strings over \(\Sigma\) including the empty string \(\lambda\). For a string \(w\), we denote the length of \(w\) by \(|w|\).

\(^1\)http://haskell.math.kyushu-u.ac.jp/
Definition 1. A finite automaton is a five-tuple of \( M = (Q, \Sigma, \delta, q_0, F) \), where \( Q \) is the finite set of states, \( \Sigma \) is the alphabet, \( q_0 \) is the initial state, \( F \) is the set of final states and \( \delta : Q \times \Sigma \rightarrow Q \) is the transition function.

A transition function \( \delta : Q \times \Sigma \rightarrow Q \) is generally extended to a function \( \delta^* : Q \times \Sigma^* \rightarrow Q \) by \( \delta^*(q, \lambda) = q \) and \( \delta^*(q, xw) = \delta^*(\delta(q, x), w) \) for \( q \in Q, x \in \Sigma \) and \( w \in \Sigma^* \).

Definition 2. For a finite automaton \( M = (Q, \Sigma, \delta, q_0, F) \), we define the language \( L(M) \) accepted by \( M \) by \( L(M) = \{ w \in \Sigma^* | \delta^*(q_0, w) \in F \} \).

Example 1. Automata \( M_1 \) and \( M_2 \) is defined as follows. 
\( M_1 = ((\{0,1\}, \{a,b\}, \delta_1, 0, \{1\}) \) and \( M_2 = ((\{0,1,2\}, \{a,b\}, \delta_2, 0, \{1\}) \), where \( \delta_1(0,a) = 0, \delta_1(0,b) = 1, \delta_1(1,a) = 1, \delta_1(1,b) = 0, \delta_2(0,a) = 1, \delta_2(0,b) = 2, \delta_2(1,a) = 2, \delta_2(1,b) = 0, \delta_2(2,a) = 2, \delta_2(2,b) = 2 \). Figure 1 is the transition diagram for \( M_1 \) and \( M_2 \). The examples are expressed as follows using our Haskell Modules.

\[
\begin{align*}
m1::\text{Automaton} &
\begin{cases}
(0,1), \{a',b'\}, d1, 0, \{1\} \\
(0,1), \{a',b'\}, d1, 0, \{1\}
\end{cases} \\
m2::\text{Automaton} &
\begin{cases}
(0,1,2), \{a',b'\}, d2, 0, \{1\} \\
(0,1,2), \{a',b'\}, d2, 0, \{1\}
\end{cases}
\end{align*}
\]

We note that \( L(M_1) = \{ w \in \Sigma^* | |w|_b = 1 \pmod 2 \} \) and \( L(M_2) = \{ a(ba)^n | n = 0,1,\ldots \} \). In our module the function \texttt{Automaton.language} returns the accepted language. To compute the accepted language generated by \( M_1 \), we use \texttt{Automaton.language m1}, where \( m1 \) is the automaton described in Haskell.

Following is a code for finding accepted language and their executions.

\[
\begin{align*}
\texttt{accepts: : Automaton} &\rightarrow [\text{String}] \\
\texttt{accepts m ss} &\rightarrow [\text{String}] \\
\texttt{accepts m ss} &\rightarrow [\text{String}]
\end{align*}
\]

\[
\begin{align*}
\texttt{language::Automaton} &\rightarrow \text{[String]} \\
\texttt{language m = accepts m (astar s)}
\end{align*}
\]

3. Sticker Module

Definition 3. Let \( \Sigma \) be a set of alphabet and \( Z \) a set of integers and \( \rho \subseteq \Sigma \times \Sigma \times Z \). An element \((i,r,n) \in \Sigma^* \times \Sigma^* \times Z \) is a domino over \((\Sigma,\rho)\), if the following conditions holds:

- If \( n \geq 0 \) then \((|i+n|,r[i]) \in \rho, \) for \( 1 \leq i \leq \min(|i|,n,r[i]) \)
- If \( n < 0 \) then \((|i|,r[r[i]-n]) \in \rho, \) for \( 1 \leq i \leq \min(n,|r|,|i|) \)

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

The possible shapes of the dominoes are illustrated as follows:

\[
\begin{align*}
&((x,u),(x',v)) \rightarrow ((\lambda,C), AT) \\
&((x,v),(x',u)) \rightarrow ((G,C), CG)
\end{align*}
\]

in our module. Similarly, \((G,\lambda)\) can be represented by \((GAT,TA,1)\).

Definition 4. The sticking operator \( \mu : D \times D \rightarrow D \cup \{\perp\} \) is defined as follows:

\[
\mu((l_1,r_1,n_1),(l_2,r_2,n_2)) \begin{cases} 
(l_1l_2,r_1r_2,n_1) & \text{(if \( ^* \))} \\
\perp & \text{(otherwise)}
\end{cases}
\]

\((^*) (l_1l_2,r_1r_2,n_1) \in D \) and \( n_1 + \left| r_1 \right| - \left| l_1 \right| = n_2 \)

Definition 5. Sticker System \( \gamma = (\Sigma, \rho, A, R) \) of an alphabet set \( \Sigma, \rho \subseteq \Sigma \times \Sigma \), a finite set of axioms \( A(\subseteq D) \) and a finite set of pairs of dominoes \( R \subseteq D \times D \).

Let \( Q = \{ q_0,q_1,\ldots,q_k \} \) be a finite set, which consists of \( k+1 \) elements. For a finite automaton \( M = (Q,\Sigma,\delta,q_0,F_M) \), the sticker system \( \gamma_M = (\Sigma,\rho,A,R) \) is defined as follows:

\[
\begin{align*}
\rho &\rightarrow \{(a,a) | a \in \Sigma\} \\
A &\rightarrow A_1 \cup A_2 \\
A_1 &\rightarrow \{(x,x,0) | x \in L(M), |x| \leq k + 2\}
\end{align*}
\]
For a sticker system $\gamma = (\Sigma, \rho, A, R)$, we define a relation $\Rightarrow$ on $D$ as follows.

$$x \Rightarrow y \overset{\text{def}}{\iff} y = \mu(a, \mu(x, \beta))$$

for some $(a, \beta) \in R$.

Let $\Rightarrow^*$ be the reflexive and transitive closure of $\Rightarrow$.

**Definition 6.** The set of dominos $LM(\gamma)$ generated by $\gamma$ is defined by $LM(\gamma) = \{(l, r, 0) | a \Rightarrow^* (l, r, 0), a \in A, |l| = |r|\}$. The language $L(\gamma)$ generated by $\gamma$ is defined by $L(\gamma) = \{l \in \Sigma^* | (l, r, 0) \in LM(\gamma)\}$.

**Example 3.** Consider the sticker system $\gamma_{M_1}$ generated by the automaton $M_1$ in Example 1. Since $\delta^*(0,0b) = 1$ then the domino $(0b, 0b) \in A$. Also we have $((\lambda, \lambda, 0), (bab, babbab, -2)) \in F$ by $\delta^*(1,bab) \in FM_1$. The domino $(bab, 0b)$ is figured as $b b b$ and $(bab, babbab, -2)$ is figured as $b b b$.

$$\mu((bab, 0b), (bab, babbab, -2)) = (babab, babbab, 0b)$$

Since $(bab, 0b) \in A$ and $(bab, babbab, -2) \Rightarrow^* (babab, babbab, 0b)$, we have $babbab \in L(\gamma_{M_1})$.

For $i = 1, \ldots, k + 1$, we define $X_i, Y_i$ and $F_i$ as follows:

$$X_i = \{(xu, x, 0) \in A | |xu| = k + 2, |u| = i, u, x \in \Sigma^*\}$$

$$Y_i = \{(xu, x, 0) | a \Rightarrow^* (xu, x, 0), a \in A, |u| = i, u, x \in \Sigma^*\}$$

$$Z_i = \{(\lambda, \lambda, 0), (xuz, -i) \in F | |v| = i, v, x \in \Sigma^*\}$$

**Lemma 1.** Define the sticker system $\gamma = \gamma_M$ for a finite automaton $M = (Q, \Sigma, \delta, q_0, F_M)$. For $i = 1, 2, \ldots, k + 1$ the followings hold.

(i) For $a \in A$. If $a \Rightarrow^* (l, r, n)$, then $n = 0$ and $|v| \leq |l| \leq |r| + k + 1$.

(ii) If $(l, r, 0) \in \gamma$, then $(l, r, 0) \in A$ and $(l, r, 0) \Rightarrow^* (l, r, 0)$, and there exist $x, u, x' \in \Sigma^*$ such that $|u| = i, 1 \leq |x|, 1 \leq |x'|, l = xuz$ and $(\lambda, \lambda, 0), (xuz, -i) \in F_i$. i.e. $\mu((xu, x, 0), (xu, xz, -i)) = (l, r, 0)$ and $(xu, x, 0) \in Y_i$.

(iii) $X_i = \{(xu, x, 0) | |u| = i, |xu| = k + 2, u, x \in \Sigma^*, \delta^*(q_0, xu) = q_{i-1}\}$.

(iv) If $(xu, x, 0) \in Y_i$ and $|xu| \leq k + 2$ then $(xu, x, 0) \in X_i$.

(v) If $(xu, x, 0) \in Y_i$ and $|xu| > k + 2$, then there exist $x''u', x' \in \Sigma^*$ such that $|x''u'| = k + 2$, $1 \leq |u'| \leq k + 1$, $x''u' \neq x$ and $(l, x''u', -u') \in F_i$. i.e. $\mu((x''u', x', 0), (x''u', x', -u')) = (xu, x, 0)$ and $(x''u', x', 0) \in Y_i$.

$$\begin{array}{c|c|c|c}
\delta(xu, xu) & u' & x''u' & u'' \\
\hline
xu & x & u'' & 0 \\
\end{array}$$

(vi) $Y_i = \{(xu, x, 0) | |u| = i, \delta^*(q_0, xu) = q_{i-1}, (k + 2) \leq |xu|, u, x \in \Sigma^*\}$.

(vii) $F = \bigcup_{i=1}^{k+1} Z_i$.

**(Proof)** (i),(iii),(iv) and (vii) are trivial.

(ii) Let $(l, r, 0)$ be a domino and $(\lambda, \lambda, 0), (xuz, -i) \in F_i$. If $\mu((l, r, 0), (xuz, -i)) \neq 1$ then $\mu((l, r, 0), (xuz, -i)) = (l, r, 0)$ and $(xuz, -i) \in \gamma$. Let $(l, r, 0) \in A$ and $(xuz, -i) \in F_i$ and $(xuz, -i) \Rightarrow (l, r, 0)$.

(v) Since $|xu| > k + 2$, there exists a domino $(x''u', x', 0)$ such that $a \Rightarrow^* (x''u', x', 0) \Rightarrow (xu, x, 0)$.

(vi) Since $|xu| > k + 2$, there exists a domino $(x''u', x', 0)$ such that $a \Rightarrow^* (x''u', x', 0) \Rightarrow (xu, x, 0)$.

(vi) Assume $(xu, x, 0) \in Y_i$ for any $xu \in \Sigma^*$ with $|xu| = n(k + 2)$. Let $(xu, x, 0)$ be a domino and $|xu| = (n + 1)(k + 2)$. We put $x = x''u'x''$ where $|x''u'| = k + 2, 1 \leq |u'| \leq k + 1$ and $\delta^*(q_0, x''u') = q_{n-1}$. Since $|x''u'| = |x''u'x''u'| = (n + 1)(k + 2)$, we have $(x''u', x', 0) \in Y_i$ by the hypothesis of the induction. Since $\delta^*(q_{n-1}, x''u') \neq q_{n-1}$, we have $(xu, x, 0) \in Y_i$.

The idea of the proof of the next theorem is originally introduced by Paun and Rozenberg in 1998. It lacks several conditions and formal proofs in their paper. We modified and improved them and proved it using our formulations.
Theorem 1. Define the sticker system $\gamma = \gamma_M$ for a finite automaton $M = (Q, \Sigma, \delta, q_0, F_M)$. Then $L(\gamma) = L(M)$.

(Proof) Let $w \in L(\gamma_M)$. Then we have $a \Rightarrow \gamma^* (w, w, 0)$ for some $a \in A$. If $(w, w, 0) \in A$ then $w \in L(M)$ by definition. If $(w, w, 0) \notin A$ then there exist $(xu, x, 0)$ and $((\lambda, \lambda, 0), (x', xu', -|u|) \in F$ such that $a \Rightarrow \gamma^* (xu, x, 0)$. Since $a \Rightarrow \gamma^* (xu, x, 0)$, we have $\delta^*(q_0, xu) = q_{|u|-1}$ from Lemma 1(vi). Since $((\lambda, \lambda, 0), (x', xu', -|u|) \in F$, we have $\delta^*(q_0, w) = \delta^*(q_0, xu') \Rightarrow \gamma^* (w, w, 0)$.

Example 4. Grammar Module

Deﬁnition 8. The language $L(G)$ generated by grammar $G = (\Sigma, N, R, S)$ is deﬁned as $L(G) = \{ w \in \Sigma^* | \gamma \Rightarrow \gamma^* w \}$. For a grammar $g = G$, (Grammar.language g) computes the language $L(G)$.

Example 4. The grammars $G_1 = (\{a, b\}, \{S\}, \{S \to aSb, S \to ab\}, \{S \to ab\}$. $G_2 = (\{a, b\}, \{S, A\}, \{S \to aSb, A \to aA, A \to a\}, \{\}$.}

\[ \begin{align*}
\text{Left} & = \{ (x, i, m) \in T T \mid m \leq 3k + 2 \} \\
\text{Right} & = \{ (w, m - i) \mid w \in T T, u = \text{Left}(w, i) \}
\end{align*} \]

Definition 9. Let $N = \{ X_1, X_2, \ldots, X_k \}$ be a ﬁnite set of $k$ non-terminal symbols and $S = X_1$. For a linear grammar $G = (\Sigma, N, P, S)$, the sticker system $\gamma_G = (\sigma, \rho, A, R)$ is deﬁned similar to [3] as follows.

\[ \begin{align*}
\rho & = \{ (a, a) \mid a \in \Sigma \} \\
X_1 & = S \{ i = 1 \text{ then } X_1 = S \} \\
T(i, k) & = \{ w \in \Sigma^* \mid |x_i \Rightarrow ^* w, |w| = k \} \\
T(i, l, r) & = \{ (w, j, r) \in (\Sigma^* \times N \times \Sigma^*) \mid \} \\
X_1 & = w X_1 w, |w| = l, |w| = r \} \\
A & = A_1 \cup A_2 \cup A_3 \\
A_1 & = \{ (x, 0, 0) \mid x \in T, m \leq 3k + 2 \} \\
A_2 & = \{ (w, x, 0) \mid w \in T T, i + 1 \leq m \leq 3k + 2, x = \text{Right}(w, m) \} \\
A_3 & = \{ (x, 0, 0) \mid w \in T T, i + 1 \leq m \leq 3k + 2, x = \text{Left}(w, m) \} \\
R & = \{ R_i \mid R_i \subseteq \text{Left}(w, i) \} \\
R_1 & = \{ (w, j, z) \in (T(i, k + 1), l) \mid u = \text{Left}(w, i), x = \text{Right}(w, i) \} \\
R_2 & = \{ (w, i) \mid w \in T T, i + 1 \leq m \leq 3k + 2 \} \\
\end{align*} \]
We modified the limitation of the production rules ([3]) in G to allow the form \( X \to xYy \) for \( |x| = |y| = 1 \). To prove the next generalized theorem, we change the limit length of w in A from 3k + 1 to 3k + 2, the length of z in \( R_1, R_2, R_3 \) and \( R_5 \) from k + 1 to k + 2, and the length of z in \( R_3 \) and \( R_6 \) from 2k + 2 to 2k + 2.

**Theorem 2** ([3]). Define the sticker system \( \gamma = \gamma_G \) for a linear grammar \( G = (\Sigma, N, P, S) \). If a linear grammar \( G \) has only production rules of the forms \( X \to xYy \) and \( X \to x \) for \( X, Y \in N, x, y \in T^* \), then \( |x|, |y| \leq 1 \), then \( L(\gamma_G) = L(G) \).

**(Proof)**

We define a set \( Y_i \) for \( i = 1, \ldots, k \) as follows.

\[
Y_i = \{ xu \in \Sigma^* | a \Rightarrow^*_x (xu, x, 0), a \in A, |u| = i \} \\
\cup \{ ux \in \Sigma^* | a \Rightarrow^*_x (ux, u, -|u|), a \in A, |u| = i \}
\]

It is easy to show that \( Y_i \subseteq \{ w | X_i \Rightarrow^*_G w, |w| \geq k + 1 \} \) and \( L(\gamma_G) \subseteq L(G) \). We prove \( Y_0 \supseteq \{ w | X_i \Rightarrow^*_G w, |w| \geq k + 1 \} \) using induction on the length of \( |w| \). Assume \( X_i \Rightarrow^*_G w \) and \( |w| \geq k + 1 \). If \( |w| \leq 3k + 2 \) then there exist x and u satisfying \( w = xu \) and \( |u| = i \) such that \( (x, u, x, 0) \in A_3 \). So we have \( w \in Y_i \). We assume \( X_i \Rightarrow^*_G w \) and \( w' \in Y_j \) for any \( w' \) and \( j \) satisfying \( X_j \Rightarrow^*_G w' \) and \( |w'| < |w| \). According to the limitation of production rules in G, we have \( X_i \Rightarrow^*_G x_1x_1y_1 \Rightarrow^*_G x_1x_2x_2y_2y_3y_1 = w \) for \( x_2, y_2, y_3 \in T^*, |x_2| \leq 1, |y_2| \leq 1 \) and \( 1 \leq |y_3|p \) \( (p = 1, \ldots, n) \). If \( |w| > 3k + 2 \) then there exist \( m \) and \( X_j \) such that \( (|x_1x_2 \cdots x_m| = k + 1 \) and \( |y_1y_2y_3| \leq k + 1 \) or \( (|x_1x_2 \cdots x_m| = k + 1 \) and \( |y_1y_2y_3| = k + 1 \) \) and \( X_j \Rightarrow^*_G x_1x_2 \cdots x_mx_jy_j \). We prove the case for \( x_1x_2 \cdots x_m \leq k + 1 \) and \( |y_1y_2y_3| = k + 1 \) in the following. The other case is similarly proved. Let \( w' \in x_{m+1} \cdots x_ny_\cdots y_{m+1} \). We note \( |w'| > 3k + 2 - (k + 1) = k \) by \( |w| > 3k + 2 \). Since \( X_j \Rightarrow^*_G w' \) and \( |w'| < |w| \), we have \( w' \in Y_j \) using the assumption of the induction. Since \( X_j \Rightarrow^*_G w' \) and \( |w'| \geq k + 1 \), there exist \( x' \) and \( u' \) satisfying \( w' = x'u' \) and \( |u'| = j \) such that \( a \Rightarrow^*_x (x'u', x', 0) \) for some \( a \in A \). Let \( x = y_n \cdots y_{n+i} \) and \( z = x_1x_2 \cdots x_n \). Since \( X_j \Rightarrow^*_G x_1x_2y_2y_1 \) and \( |u| = i \), we have \((x, z, 0), (x, u', -|u'|) \in R_1 \) and \((x'u', x', 0) \Rightarrow^*_x (z'u'xu, zu'u'x) \). Since \( z'u'xu = w \) and \(|w| = i \), we have \( w \in Y_i \).

Next we prove \( L(G) \subseteq L(\gamma_G) \). Let \( w \in L(G) \). If \(|w| > 3k + 2 \) then \(|w| \in A \) and \( w \in L(\gamma_G) \). Assume \(|w| > 3k + 2 \). According to the limitation of production rules in G, we have \( S \Rightarrow^*_G x_1x_1y_1 \Rightarrow^*_G x_1x_2X_2y_2y_1 \Rightarrow^*_G x_1x_2 \cdots x_ny_1 \cdots y_2y_1 \) for \( x_2, y_2, y_1 \in T^* \). If \( |y_1| \leq 1 \) and \(|y_2y_3| \leq k + 1 \) or \(|y_1y_2 | = k + 1 \) and \(|y_1y_2y_3| = k + 1 \) and \( S \Rightarrow^*_G x_1x_2 \cdots x_mX_1y_1y_2y_1 \). We prove the case for \( x_1x_2 \cdots x_m \leq k + 1 \) and \(|y_1y_2y_3| = k + 1 \) in the follows. Let \( w' = x_{m+1} \cdots x_ny_\cdots y_{m+1} \). All of elements in \( Y_i \) and \( y_1y_2y_3 \) exist \( x' \) and \( u' \) satisfying \( w' = x'u' \) and \( |u'| = i \) such that \( a \Rightarrow^*_x (x'u', x', 0) \) for some \( a \in A \). Since \( S = X_1 \Rightarrow^*_G x_1x_2 \cdots x_ny_1y_2y_3y_1 \) and \(|x_1x_2 \cdots x_m| + |y_1y_2y_3| \leq 2k + 2 \), we have \((x_1x_2 \cdots x_m, 0), (y_1y_2y_3, 0) \in R_6 \). Since \( \mu(x_1x_2 \cdots x_m, 0) \mu(x_1x_2 \cdots x_m, 0) \mu(x'u', x'), 0, (y_1y_2y_3, 0, 0) = (w, w, 0) \), we have \( w \in L(\gamma_G) \).

**Example 5.** Consider the Language generated by linear grammar \( G = (\{S\}, \{a, b\}, S \to ab, S \to abS) \).

The language generated by G is \( L(G) = \{a^nb^n | n \geq 1 \} \).

Now we can induce the domino \( \begin{array}{c} aaaaabbbbb \end{array} \) by using a pair of elements \( \begin{array}{c} a & b & bb \\ a & aa & bb \\ b & bb & ab \end{array} \) in \( R_4 \) and \( \begin{array}{c} aab \\ bba \end{array} \) in \( A_3 \). All of elements in A and R are listed in Appendix.

## 5. Conclusion

We can define the dominoes using set theoretical notations in Haskell and simulate sticker systems, finite automata and grammar systems. Using our system, we could find some insufficient conditions to construct the sticker systems written in [3]. One of related work is implementation of HaLex [5]. HaLex is a Haskell library enables us to model and manipulate a regular language. HaLex also provides the facilities for defining deterministic and non-deterministic finite automata, regular expressions etc. It does not represent an infinite set as a language. One of the merits of our modules is treating the generated languages as an infinite set using lazy evaluations.

**Acknowledgment**

We would like to thank AI Omokada for her contribution to write Haskell codes and the great support made during
the research.

REFERENCES


APPENDIX

In Appendix, we show examples of sticker systems generated from automata and grammar by using our Haskell module functions.

Example 6. For an automaton $M_1 = (\{0,1\}, \Sigma, \delta, 0, \{1\})$ in Example 1, we have the sticker system $\gamma_{M_1}$ as follows.

\[
\gamma_{M_1} = (\Sigma, \rho, A, R)
\]

\[
\rho = \{(a,a), (b,b)\}
\]

\[
A = A_1 \cup A_2
\]

\[
R = \{\bullet \rightarrow A, aSb \}
\]

Example 7. For a linear grammar $G_1 = \{S, \{a,b\}, S, \{S \rightarrow ab, S \rightarrow aSb\}\}$, we have the sticker system $\gamma_{G_1}$ as follows.

\[
\gamma_{G_1} = (\Sigma, \rho, A, R)
\]

\[
\rho = \{(a,a), (b,b)\}
\]

\[
A = A_1 \cup A_2 \cup A_3
\]

\[
R = R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5 \cup R_6
\]

K.K.K.R. Perera
Graduate School of Mathematics, Kyushu University, Japan.
E-mail: kisani(at)math.kyushu-u.ac.jp

Y. Mizoguchi
Faculty of Mathematics, Kyushu University, Japan.
E-mail: ym(at)math.kyushu-u.ac.jp