Technical Note

A Note on Homomorphic Representation of Recursively Enumerable Languages with Insertion Grammars

Kaoru Onodera†

In this note we obtain some representation theorems for recursively enumerable languages by using insertion systems and morphisms. Specifically, we show that for any recursively enumerable language L, there effectively exist an insertion system γ and homomorphisms h_1, h_2 such that $L = h_1 h_2^{-1}(L(\gamma))$, where γ is of weight $(3,3)$. This result remarkably improves an existing one where the corresponding insertion system is of weight $(4,7)$.

1. Introduction

DNA computing theory involves the use of *insertion* and *deletion* operations. Intuitively, for a given string $xuvy$, an insertion operation with context (u,w,v) produces a new string $xuwvy$. Conversely, for a string $xuwvy$, a deletion operation with context (u,w,v) produces a new string $xuvy$.

In this paper, using only insertion operations, we consider the generative power of insertion systems. We focus on characterising recursively enumerable languages by using insertion systems and some morphisms.

There have been a number of homomorphic characterizations for classes of languages. As a negative result, there exists no regular language L_0 such that for each regular language L, $L = h_1(h_2^{-1}(L_0))$, where h_1, h_2 are morphisms [4]. For the class of context-free languages, there exist fixed context-free language L_0 and morphisms h_1, h_2 such that $L = h_1(h_2^{-1}(L_0))$ [5].

A traditional topic in insertion systems has been the representation of the class of recursively enumerable languages by simple operations on a small subclass of it. Geffert [1] has shown that each recursively enumerable language $L \subset \Sigma^*$ can be expressed as $L = \{h_1(h_2^{-1}(x)) |\ x \in \Gamma^+\} \cap \Sigma^*$, where Γ is an alphabet and h_1, h_2 is a pair of morphisms. By using insertion systems, the following representation is known [2]: for any recursively enumerable language L, there exist an insertion system γ and morphisms h_1, h_2 such that $L = h_1 h_2^{-1}(L(\gamma))$, where γ is of weight $(4,7)$. In this paper, we improve the result in a remarkable way, and show that in the representation $L = h_1 h_2^{-1}(L(\gamma))$, γ can be simplified to be of weight $(3,3)$.

† Graduate Course of Mathematics Education, School of Education, Waseda University

2. Preliminaries

An *insertion system* (*ins system*, for short) is a triple $\gamma = (V, P, S)$, where V is an alphabet, S is a finite set of strings over V called axioms, and P is a finite set of *insertion rules* of the form (u,x,v), where $u,x,v \in V^*$. We define a relation $\Rightarrow$ on V^* by $w \Rightarrow z$ iff $w = w_1 uvw_2$ and $z = w_1 uxvw_2$ for some insertion rule $(u,x,v) \in P$, $w_1, w_2 \in V^*$. If there is a danger of confusion, we write $\Rightarrow_\gamma$. As usual, $\Rightarrow^*$ denotes the reflexive and transitive closure of $\Rightarrow$. A language generated by γ is denoted by $L(\gamma) = \{w \in V^* |\ s \Rightarrow^* w,\ s \in S\}$.

An insertion system $\gamma = (V, P, S)$ is said to be of *weight*(m,n) iff

$$m = \max\{lg(x) |\ (u,x,v) \in P\},$$
$$n = \max\{lg(u) |\ (u,x,v) \in P \text{ or } (v,x,u) \in P\},$$

where $lg(x)$ denotes the length of x. By INS_m^n, we denote the family of all languages generated by ins systems of weight (m', n'), where $m' \leq m$ and $n' \leq n$. When the parameter is not bounded, we replace m or n with $*$.

The classes of finite, context-free, context-sensitive, and recursively enumerable languages are denoted by FIN, CF, CS, and RE, respectively.

3. Main Result

The characterization of recursively enumerable languages is now sharpened into the following form originally established in Ref. 2).

Theorem 1 *For each recursively enumerable language L, there exist a morphism h, a weak coding g, and a language $L' \in INS_3^3$ such that $L = g(h^{-1}(L'))$.*

Proof. Consider a recursively enumerable lan-

guage L over T^*, generated by a type-0 Chomsky grammar $G = (N, T, P, S)$ in the Penttonen normal form. In this normal form, the rules in P are of the following types:

- type 1
  $X \to x$, where $X \in N, x \in (N \cup T)^*, lg(x) \leq 2$,
- type 2
  $XY \to XZ$, where $X, Y, Z \in N$.

Consider new symbols $\#, c$ and construct an insertion system $\gamma = (N \cup T \cup \{\#, c\}, P', \{Scc\})$, where P' contains the following insertion rules:

- **Group 1**
  For each rule $r : X \to YZ \in P$ of type 1, with $X \in N$, $Y, Z \in N \cup T \cup \{\lambda\}$, we construct the following insertion rule:
  (r_1) $(X, \#YZ, \alpha_1\alpha_2)$, where $\alpha_1\alpha_2 \in (N \cup T \cup \{c\})^2$.
- **Group 2**
  For each rule $r : XY \to XZ \in P$ of type 2, with $X, Y, Z \in N$, we construct the following insertion rule:
  (r_2) $(XY, \#Z, \alpha_1\alpha_2)$, where $\alpha_1\alpha_2 \in (N \cup T \cup \{c\})^2$.
- **Group 3** (Relocation task for X)
  For each $X, Y \in N \cup T$, we construct the following insertion rules:
  (r_3) $(XY\#, \#X, \alpha)$, where $\alpha \in (N \cup T \cup \{c\})$,
  (r_4) $(X, \#, Y\#\#)$,
  (r_5) $(\#Y\#, Y, \#X)$.

Consider the symbol a_x for each string $x \in V\{\#\}$, where $V = N \cup T$. Let A be the set of a_x symbols.

We define a morphism $h : (A \cup T \cup \{c\})^* \to (N \cup T \cup \{\#, c\})^*$ by

$$h(a_x) = x,\ x \in V\{\#\},$$
$$h(b) = b,\ b \in T,$$
$$h(c) = c.$$

We also define a weak coding $g : (A \cup T \cup \{c\})^* \to T^*$ by

$$g(a_x) = \lambda,\ x \in V\{\#\},$$
$$g(b) = b,\ b \in T,$$
$$g(c) = \lambda.$$

We then obtain $L(G) = g(h^{-1}(L(\gamma)))$, which will be proved in the sequel.

We call the symbol # a *marker*. A symbol in $V(= N \cup T)$ followed by # is said to be #-*marked* (*marked*, for short). A symbol in V which is not marked is said to be *unmarked*. Since the symbols c and # are special symbols, they are neither marked nor unmarked. A string xcc, where x is in $(V\{\#\} \cup N \cup T)^*$, is a *legal* string.

Intuitively, a marked symbol means that the symbol has been used for some derivation. At each step, an unmarked symbol of a legal string corresponds to a sentential form of G. Marked symbols and the markers are considered wrecks for some derivations. From the definitions of h and g, we get legal strings; if we erase the wrecks and the symbol c, then we get unmarked symbols of the legal strings.

By using the rules of groups 1 and 2, we can simulate rules of types 1 and 2 respectively. By using the rules of group 3, we move an unmarked symbol to the right across a block $M\#$, where $M \in N \cup T$. Thus we can create nonterminal pairs, which are needed for simulating rules of type 2. When we use the rules of group 3 repeatedly, we can relocate all the blocks $M\#$ in the left of the string.

In order to prove the equality $L(G) = g(h^{-1}(L(\gamma)))$, we first prove the inclusion $L(G) \subseteq g(h^{-1}(L(\gamma)))$.

Fact 1 *For the rule $r_1 : (X, \#YZ, \alpha_1\alpha_2)$, the symbol X is unmarked before the derivation but marked after the derivation. The symbols Y, Z are unmarked after the derivation.*

Fact 2 *For the rule $r_2 : (XY, \#Z, \alpha_1\alpha_2)$, the symbol X is unmarked before and after the derivation. The symbol Y is unmarked before the derivation, but marked after the derivation. The symbol Z is unmarked after the derivation.*

Lemma 1 *The rules in group 3 can replace a substring $XY\#\alpha$ ($\alpha \in N \cup T \cup \{c\}$) by a substring consisting of the strings in $V\{\#\}$ and ending with $X\alpha$, where $V = N \cup T$. The symbol X is unmarked before and after the derivations.*

Proof. Rule r_3 can be applied to a string $XY\#\alpha$, where $X, Y \in N \cup T$, $\alpha \in N \cup T \cup \{c\}$. After applying rule r_3, we have $XY\#\#X\alpha$. Then rule r_4 can be applied for the substring $XY\#\#$, and we have $X\#Y\#\#X\alpha$. Now we apply rule r_5 for the substring $\#Y\#\#X$, and the substring is replaced by $\#Y\#Y\#X$.
Therefore, the substring $XY\#$ is replaced by $X\#Y\#Y\#X$, which has the unmarked symbol X in the rightmost position. (q.e.d.)

Thus the insertion rules in γ simulate the rules in G, and generate legal strings from the legal string Scc.

Denote by $umk(x)$ a string consisting of unmarked symbols in a legal string x generated by γ. Note that since c is a special symbol, neither marked nor unmarked, $umk(x)$ does not

contain a suffix cc. We thus we have the next lemma:

Lemma 2 *If S derives x in G, then there exists a derivation $Scc \Rightarrow_\gamma^* x'$ in γ such that $umk(x') = x$.*

Proof. We prove the lemma by induction on the derivation steps in G. Consider a derivation $S \Rightarrow_G^n x$ in G.

If $n = 0$, then for the axiom Scc in γ, $umk(Scc) = S$, and thus the claim holds.

We suppose that the claim holds for any $n \leq k$. Now consider a derivation $S \Rightarrow_G^k x \Rightarrow_G y$. From the induction hypothesis, there exists a derivation in γ such that $Scc \Rightarrow_\gamma^* x'$, where $umk(x') = x$. If the rule applied for x is of type 1 or 2, then we use the corresponding insertion rule in group 1 or 2 respectively for the string x'. However, in the latter case, if the insertion rule in group 2 cannot be applied for x', we need to apply some rules in group 3. From Lemma 1, after application of the rules in group 3, no unmarked string of a legal string changes. We denote this derivation by $x' \Rightarrow^* x'' \Rightarrow y'$, where x'' is derived by using rules r_3, r_4, r_5 in group 3 and y' is derived by using a rule in group 1 or 2. Note that $umk(x') = umk(x'')$. Then, in either case, from Facts 1 and 2 we eventually have $umk(y') = y$.

Therefore the claim holds for $k+1$. (q.e.d)

In view of the manner of constructing the morphisms g, h, we have the following fact:

Fact 3 *For some $y \in L(\gamma)$, if an equation $x = g(h^{-1}(y))$ holds, then y is legal and $x = umk(y)$, where $x \in T^*$, $y \in (\{N\#\} \cup T)^*\{cc\}$. And if $y \in L(\gamma)$ and $umk(y) \in T^*$, then $umk(y) = g(h^{-1}(y))$.*

From Lemmas 1 and 2 and Fact 3, we obtain the inclusion $L(G) \subseteq g(h^{-1}(L(\gamma)))$.

Next we prove the inverse inclusion.

Fact 4 *From the constructions, the rules in groups 1 and 2 can only simulate rules of types 1 and 2 respectively in G on unmarked symbols.*

We will give separate consideration to the case of using the rules in group 3.

Lemma 3 *Once we apply the rule r_3 to obtain a substring of a legal string, we have to use the rules r_4 and r_5 in this order.*

Proof. Let us consider a substring $XY\#\alpha$, where $X, Y \in N \cup T$ and $\alpha \in N \cup T \cup \{c\}$. After using rule r_3, we obtain $XY\#\#X\alpha$. Because of the symbols $\#\#$, rules r_1, r_2, r_3 cannot be applied for the symbols X or Y that are followed by $\#\#$. In view of the form of rule r_5, we cannot apply r_5 for $XY\#\#$. Hence, the only applicable rule for $XY\#\#$ is r_4.

For the symbol X following $\#\#$, we have a chance to apply one of the rules r_1, r_2, r_3, and r_4. If we apply r_1 or r_2, we may take it as the first step of simulation for type 1 or 2 respectively. Note that, during these simulations, X remains immediately to the right of $\#\#$. If we apply r_3 or r_4, we may take it independently as a new relocation task. Note that, after application of r_3 or r_4, X remains immediately to the right of $\#\#$. Therefore, in all cases the symbol $\#\#$ is followed by X. Further, since the symbol X was originally unmarked in $XY\#\alpha$, X provides the possibility of applying one of the rules r_1, r_2, r_3, r_4. Hence this application causes no trouble with the current relocation task.

After using rule r_4 for $XY\#\#$, we obtain $X\#Y\#\#$. From the above notation, since X always follows the symbols $\#\#$, after applying r_4, we obtain $X\#Y\#\#X$. In the substring $X\#Y\#\#$, both of the symbols X and Y are already marked and, in view of the form of the rules, none of r_1, r_2, r_3, r_4 can be used for this substring. Hence, the only applicable rule for $X\#Y\#\#X$ is r_5. After applying this rule, we have $X\#Y\#X\#X$, which is the substring of a legal string.

Hence, to obtain a substring of a legal string, whenever we use rule r_3, we have to use r_4 and r_5 in this order. (q.e.d)

From Fact 4, Lemma 3, and a similar argument in the proof of Lemma 2, every string of a form $umk(xcc)$ is generated by the grammar G, where xcc is a legal string generated by γ.

We now return to the proof of Theorem 1. If h^{-1} is defined for some $y' \in L(\gamma)$ and $y' \in (\{V\#\} \cup T)^*\{cc\}$, where $V = N \cup T$, then there exists a string $y = h^{-1}(y')$ such that $S \Rightarrow_G^* g(y)$. This means that the inclusion $g(h^{-1}(L(\gamma))) \subseteq L(G)$ holds, which completes the proof of Theorem 1. (q.e.d)

Consider the following regular language $R = \{(\alpha\#)^*\beta |\ \alpha \in (N \cup T),\ \beta \in T^*\}\{cc\}$. On the basis of the above theorem, we obtain

Corollary 1 *Each recursively enumerable language L can be written as follows: $L = g(R \cap L')$, where g is a weak coding, R is a regular language, and $L' \in INS_m^n$, for $m \geq 3, n \geq 3$.*

4. Conclusions

This paper is motivated by Ref. 2), in which it is proved that for each language $L \in RE$, there are a morphism h, a weak coding g, and a language $L' \in INS_4^7$ such that $L = g(h^{-1}(L'))$.

In Ref. 2), an insertion system simulates the Kuroda normal form, and introduces two different marker symbols # and \$, while in our paper we started with the Penttonen normal form, which enables us to simulate rules of type 2 (i.e, context-sensitive rules), and reduce the number of insertion rules, as well as the length of context checking. By introducing a single new marker symbol # with suffix *cc*, we were able to remarkably improve the parameters of the weight from $(4, 7)$ to $(3, 3)$.

The following results exist for families of insertion languages [3]:

- $FIN \subset INS_*^0 \subset INS_*^1 \cdots \subset INS_*^* \subset CS$.
- $INS_*^1 \subset CF$.
  CF is incomparable with all INS_*^n $(n \geq 2)$, and INS_*^*.
  INS_2^2 contains non-semilinear languages.

Further, there exists a fact that CF is closed under inverse morphisms and any morphisms. On the basis of these facts, for the insertion language L' in the representation of RE, the weight of an insertion system γ cannot be $(n, 0)$ or $(n, 1)$ for any $n \geq 1$. It is an open problem whether the insertion system of weight $(n, 2)$, for some $n \geq 1$, can express RE in such a form.

Acknowledgments I would like to express my thanks to T. Yokomori for discussions and suggestions.

References

1) Geffert, V.: A representation of recursively enumerable languages by two homomorphisms and a quotient, *Theoretical Computer Science*, Vol.62, pp.235–249 (1988).
2) Martin-Vide, C., Păun, G. and Salomaa, A.: Characterizations of recursively enumerable languages by means of insertion grammars, *Theoretical Computer Science*, Vol.205, pp.195–205 (1998).
3) Păun, Gh., Rozenberg, G. and Salomaa, A.: *DNA Computing*, Springer-Verlag (1998).
4) Salomaa, A.: *Jewels of Formal Language Theory*, Computer Science Press, Inc. (1981).
5) Yokomori, T.: On purely morphic characterizations of context-free languages, *Theoretical Computer Science*, Vol.51, pp.301–308 (1987).

(Received November 5, 2002)
(Accepted February 4, 2003)

Kaoru Onodera was born in Tokyo. She graduated from Department of Mathematics, School of Education, Waseda University in 2000 and received her M.S. degree in 2002. Since then, she has been in the doctoral program at Graduate School of Education, Waseda University. Her current research interests include the theory of DNA computing and formal language theory. She is a member of IPSJ.