On the Weight of Universal Insertion Grammars

Lila Kari[1] Petr Sosík[1,2]

[1]Department of Computer Science
University of Western Ontario, London, ON, Canada, N6A 5B7,
[lila,sosik]@csd.uwo.ca

[2]Institute of Computer Science
Silesian University, Opava, Czech Republic
petr.sosik@fpf.slu.cz

Abstract

We study the computational power of pure insertion grammars. We show that pure insertion grammars of weight 3 can characterize all recursively enumerable languages. This is achieved by either applying an inverse morphism and a weak coding, or a left (right) quotient with a regular language. A consequences for the closure properties of insertion grammars are shown. We also study an application in DNA computing and improve some known results concerning the power of insertion-deletion DNA systems.

1 Introduction

Insertion grammars have been introduced and studied in [2]. The motivation for their study originates in mathematical linguistics, as they are similar to contextual grammars [3, 5] and to pure grammars with a specific type of rules. Their computational power was further studied in [6] where the following result was proven: Insertion grammars of weight at least 7 can characterize all recursively enumerable languages via a weak coding and a morphism. An analogous result was shown using the left quotient with a regular language instead of a weak coding and a morphism, on the one hand. On the other hand, there exist linear languages which cannot be generated by any insertion grammar (without the aid of above mentioned operations). This fact suggests rather poor closure properties of the class of insertion languages. Another immediate consequence is the incomparability of this class with many other language classes closed either under weak codings and morphisms, or under left quotient with a regular language.

The upper bound for the weight necessary to characterize the family of recursively enumerable languages was improved to 5 in [4]. In this paper we show that weight 3 is enough to achieve universality in the above sense. Our construction is similar to that in [4] in that it uses a special symbol \$ to mark deleted symbols in a sentential form. In our proof we use two marking symbols \$, #, and we achieve an improvement in the length of context thanks to their mutual interplay.

Finally, we deal with insertion-deletion DNA systems. These are motivated by the fact that the contextual insertion and/or deletion in DNA frequently occurs in certain cell types. Formalization is based on operations very similar to those used in insertion grammars. We show that as a consequence of the above mentioned results, some theorems in [7] are improved.

Recall that a *pure grammar* is a grammar that has no non-terminals. Rewriting is defined in the standard way and a specific pure grammar is defined by axioms (a set of words over the terminal alphabet) and a finite set of rewriting rules. All words derivable from the axioms belong to the language of the pure grammar.

Definition 1 *A (pure)* insertion grammar of weight $n \geq 0$ *is a triple* $G = (V, A, P)$, *where*

- V *is a finite alphabet,*
- $A \subseteq V^*$ *is a finite set of* axioms,
- P *is a finite set of insertion rules of the form* (u, x, v), *for* $u, x, v \in V^*$,
- $n = \max\{|u| \mid (u, x, v) \in P$ *or* $(v, x, u) \in P\}$.

Definition 2 *A* derivation step *of a insertion grammar* $G = (V, A, P)$ *is defined by the relation* $\Rightarrow: V^* \longrightarrow V^*$ *such that*

$$y \Rightarrow z \text{ iff } y = w_1 u v w_2,\ z = w_1 u x v w_2,\ (u, x, v) \in P,\ w_1, w_2 \in V^*.$$

The language generated by an insertion grammar $G = (V, A, P)$ is defined in the usual manner as the set $L(G) = \{z \subseteq V^* \mid y \Rightarrow^* z,\ y \in A\}$, where $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$. We denote by S_n, $n \geq 0$ the families of languages generated by the insertion grammars of weight at most n. Clearly, $S_n \subseteq S_m$ for $n \leq m$.

We also recall the following definition from [8].

Definition 3 *A type 0 grammar $G = (N, T, P, S)$ is in* Penttonen normal form*, if each rule in P has one of the forms*

$$A \to a,\ A \to BC,\ AB \to AC,\ A \to \lambda,\ \textit{for } A, B, C \in N,\ a \in T.$$

A *weak coding* is a morphism that maps each letter onto a letter or onto the empty word.

2 Universality with inverse morphism and weak coding

Theorem 4 *For each recursively enumerable language L there exists a morphism h, a weak coding g and a language $L_1 \in S_3$ such that $L = g(h^{-1}(L_1))$.*

Proof. Let $G = (N, T, P, S)$ be a recursively enumerable grammar in Penttonen normal form, generating a language L. Consider the following insertion grammar:

$$G_1 = (V, cccS, P_1),$$

where $V = N \cup N' \cup T \cup \{\$, \#, c\}$, $N' = \{A' \mid A \in N\}$, and $\$, \#, c$ are symbols not in $N \cup N' \cup T$. We furthermore denote $N_c = N \cup \{c\}$. The set of rules P_1 is constructed as follows:

1. For each rule $A \to a$ in P there are rules $(x, a\$, Ay)$ in P_1, for all $x \in N_c^3$, $y \in (V \setminus \{\#\})$.
2. For each rule $A \to BC$ in P there are rules $(x, BC\$, Ay)$ in P_1, for all $x \in N_c^3$, $y \in (V \setminus \{\#\})$.
3. For each rule $AB \to AC$ in P there are rules $(xA, C\$, By)$ in P_1, for all $x \in (V \setminus \{\$\})$, $y \in (V \setminus \{\#\})$.
4. For each rule $A \to \lambda$ in P there are rules $(x, \$, Ay)$ in P_1, for all $x \in N_c^3$ and $y \in (V \setminus \{\#\})$.

In addition to the above, P_1 contains the following rules:

5. $(x, B', \$YB)$ for all $x \in (V^2 N_c \cup \$V\#)$, $Y \in (N \cup N')$, $B \in N$.
6. $(B'\$Y, \#\$, Bx)$ for all $x \in (V^2 \setminus \{\#B\})$, $Y \in (N \cup N')$, $B \in N$;
7. $(B', \#, \$Y\#)$ for all $Y \in (N \cup N')$, $B \in N$.
8. $(x, B\$, B'\#)$ for all $x \in (V \setminus \{\$\})$, $B \in N$.

9. $(x, B, \#B)$ for all $x \in (V(V \setminus \{\$\})(N \setminus \{B\}) \cup (N \cup \{\#\})\$(N \cup N'))$, $B \in N$.

10. $(xB\#, \$, B)$ for all $x \in (V \setminus \{\$\})$, $B \in N$.

The insertion grammar G_1 simulates step-by-step the derivation of G. However, G can rewrite or delete some symbols of the generated sentential form, which is impossible in insertion grammars. Hence a marking symbol \$ is introduced into G_1. A nonterminal from $N \cup N'$ which is preceded by \$ is marked as deleted. This is achieved by the rules of types 1–4 of G_1.

This marking system introduces another problem: pairs of unmarked nonterminals which should be subject to rules of the form $AB \to AC$ can be separated by one or more marked symbols. Such a string can adopt the form $A\$C_1 \ldots \$C_nB$, $n \geq 1$. Hence the rules 5–8 are added, allowing the symbol B to "migrate" to the left-hand side of a substring of this form. Having the substring $\$YB$ for an arbitrary $B \in N$, $Y \in (N \cup N')$, these rules allow for the derivation

$$\$YB \Rightarrow B'\$YB \Rightarrow B'\$Y\#\$B \Rightarrow B'\#\$Y\#\$B \Rightarrow B\$B'\#\$Y\#\$B \qquad (1)$$

At the end all the symbols from $N \cup N'$ are marked, except B. Iterating the above derivation, the string $A\$C_1 \ldots \$C_nB$ can be rewritten as ABx, where in x all the nonterminals are marked. Still, this scheme introduces another symbol $\#$ which can separate pairs of nonterminals. Therefore the rules 9 and 10 are introduced, allowing for the derivation

$$\#B \Rightarrow B\#B \Rightarrow B\#\$B \qquad (2)$$

for an arbitrary $B \in N$. This again allows the symbol B to migrate towards the left end of the string. Finally, let $h : (V \setminus \{\$\})^*) \longrightarrow V^*$ be the morphism defined by

$$h(x) = \$x, \ x \in (N \cup N'), \text{ and } h(y) = y, \ y \in (T \cup \{c, \#\}),$$

and $g : V^* \longrightarrow T^*$ be the weak coding defined by

$$g(x) = \lambda, \ x \in (V \setminus T), \text{ and } g(y) = y, \ y \in T.$$

We show that $L(G) = g(h^{-1}(L(G_1)))$.

(i) $L(G) \subseteq g(h^{-1}(L(G_1)))$.
One can easily verify that if the rules of G_1 are used in the above

described manner, then G_1 correctly simulates all the derivations of G. During this derivation, a certain amount of auxiliary substrings of the form $\$A$, $\$A'$ and $\#$ can be inserted into the derived string. The inverse morphism h^{-1} filters only the sentential forms where all the nonterminals from $N \cup N'$ are preceded by \$ signs. Hence h^{-1} selects from $L(G_1)$ those and only those sentential forms where all nonterminals have been "deleted". Finally, g removes all the nonterminals and auxiliary symbols. Hence, $L(G) \subseteq g(h^{-1}(L(G_1)))$.

(ii) $g(h^{-1}(L(G_1))) \subseteq L(G)$.
We must show that G_1 can produce no other sentential forms w with all the nonterminals marked than those that correspond to derivations in G. First, observe that the substrings $\$\$$, $\#\#$, $\$\#$ can never occur in w. Observe also that during an incomplete series of applications of rules 5–8 and 9–10 there exists always at least one unmarked nonterminal from $N \cup N'$. Consequently, only those sentential forms that are the result of a "complete" series of applications of rules 5-8 and 9-10 are selected when applying h^{-1} (hence when applying $g \cdot h^{-1}$).

Consider now the rules 5–8.

- Given a substring of the form $\$YB$ of w, rule 5 can be applied only once to it, producing $B'\$YB$. Observe also that $B'\$YB$ cannot be produced by an application of any other rule than 5.
- Following an application of rule 5, only rule 6 can follow, which inserts any symbols into $B'\$YB$ (not counting its prefixes or suffixes), resulting in the string $B'\$Y\#\$B$. Observe that 6 can be applied only to a string of the form $B'\$YB$ produced by rule 5.
- Similarly, the only rule that can insert anything into $B'\$Y\#\$B$ is rule 7. Again, the substring $B'\$Y\#\$$ to which rule 7 can be applied, can be produced only as a result of the two previous steps.
- Finally, the substring $xB'\#$, $x \neq \$$, allowing an application of the rule 8 (and of no other rule), can occur only as a result of the previously described steps.

Hence, after an application of rule 5, the whole derivation (1) must inevitably proceed.

Analogously, consider a substring of the form $xB\#B$, $x \in V$, produced by rule 9. The only rule which can insert anything into this string is rule 10, resulting in the derivation (2). Conversely, the application of 10 is allowed only by a previous application of 9.

Of course, the derivations (1) and (2) can be interlaced by an application of other rules in other parts of the sentential form w. However, these other rules cannot interfere with these derivations. Consider for instance the derivation

$$x\$Y\#B \underset{(9)}{\Rightarrow} x\$YB\#B \underset{(5)}{\Rightarrow} xB'\$YB\#B$$

for an $x \in V$, $Y \in (N \cup N')$, $B \in N$. In this situation rule 6 cannot be applied (because of its right context) until rule 10 was applied first. Hence the derivation (2) happening at the right-hand side of the string is completed, before the derivation (1) can continue.

We can conclude that each of the rules 6–8 and 10 requires a previous application of a rule with the index smaller by one, as we have shown above. Hence the rules 5–8 or 9–10 can be applied only as a part of derivations (1) or (2), respectively.

Consequently, unmarked nonterminals in a sentential form can only be changed by the rules 1–4, and their application can be simulated by the grammar G. Moreover, the inverse morphism h^{-1} filters only the sentential forms with all the nonterminals marked. Therefore, $g(h^{-1}(L(G_1))) \subseteq L(G)$.

□

Note: The weight 3 of the insertion grammar is needed in several of the above described rules. In particular, consider the rules 5–7 and 10 used for migrating through marked nonterminals. We need to regulate their application due to the symbol next to the marked (i.e. "deleted") one. As the substring consisting of the marked nonterminal together with its mark has length 2, this regulation needs a context of length 3. Therefore, it seems that the weight 3 cannot be further improved with the "mark and migrate" technique.

Note also that the above construction can be easily adapted to use a gsm instead of a morphism and a weak coding.

3 Universality with quotient

Theorem 5 *There exists a regular language R with the following property: for each recursively enumerable language L there is a language $L' \in S_3$ such* *that* $L = L' \longrightarrow_{\mathrm{rq}} R$.

Proof. We express the given recursively enumerable language L in the form [4, 6]

$$L = (L \cap \{\lambda\}) \cup \bigcup_{a \in T} \{a\}(L \longrightarrow_{\mathrm{lq}} \{a\}).$$

Denote by $G_0 = (N_0, T, P_0, S_0)$ the type-0 grammar generating L. Notice that grammars $G_a = (N_a, T, P_a, S_a)$ generating languages $L \longrightarrow_{\mathrm{lq}} \{a\}$, $a \in T$, can be effectively constructed starting from G_0. Let us assume that all the sets N_a, $a \in T$, and also N_0 are mutually disjoint. Denote $N = N_0 \cup \bigcup_{a \in T} N_a$ and $P = P_0 \cup \bigcup_{a \in T} P_a$.

To construct the required language $L' \in S_3$, we need to alter the construction of Theorem 4 as follows. Consider an insertion grammar

$$G_1 = (V, dcS \cup \bigcup_{a \in T} acS_a, P_1),$$

where $V = N \cup N' \cup \overline{N} \cup T \cup \{\$, \#, c, d\}$, $N' = \{A' \mid A \in N\}$, $\overline{N} = \{\overline{A} \mid A \in N\}$, and $\$, \#, c, d$ are symbols not in $N \cup N' \cup \overline{N} \cup T$. Again let $N_c = N \cup \{c\}$. The set of rules P_1 from the proof of Theorem 4 is changed as follows:

- the rules of type 1 are completely omitted and replaced by types 11–14 described bellow;
- in the rules of type 2 and 4, x becomes an element of $(N_c^3 \cup \{c\}N \cup \{c\})$;
- in the rules 5, 6 and 7, Y becomes an element of $(N_c \cup N' \cup \overline{N})$;
- in the rules of type 9, x becomes an element of $(V(V \setminus \{\$\})(N \setminus \{B\}) \cup (N \cup \{\#\})\$(N_c \cup N') \cup \overline{N})$,

The rest of the construction is unchanged except that the redefined set V is used in the rules. The following new rules are added:

11. $(bc, \overline{A}\$, A)$ for all $b \in T$ and $A \in N$;
12. $(b, ac\$, c\overline{A})$ for $a, b \in T$ and $A \in N$ such that $(A \to a) \in P$;
13. $(\$c, \$, \overline{A})$ for all $A \in N$;
14. (a, d, c) for all $a \in T$.

The rules 11–13 simulate a rule $A \to a$ by the derivation

$$bcA \Rightarrow bc\overline{A}\$A \Rightarrow bac\$c\overline{A}\$A \Rightarrow bac\$c\$\overline{A}\$A, \tag{3}$$

for $a, b \in T$ and $A \in N$. Similarly as in the previous proof, one can easily check that (i) this derivation cannot be altered by any of the previously defined rules, (ii) the rules 11–13 can be applied only in the described succession as a part of the above derivation. The purpose of the derivation (3) is to collect all the terminal symbols as a prefix of the derived string. Finally, consider the regular set

$$R = dc(\$N_c \cup \$N' \cup \$\overline{N} \cup \#)^*.$$

Rule 14 stops the production of terminal symbols and allows the right quotient with the set R to produce a nonempty result. One can notice that the application of the rule 14 can follow immediately after 11, but in this case the right quotient with R will be empty, as there remains a non-removable substring $dc\overline{A}$. If 14 is applied immediately after 12, then the rule 13 can still be applied later.

Now we are ready to show that $L(G_0) = L' \longrightarrow_{\mathrm{rq}} R$.

(i) $L(G_0) \subseteq L' \longrightarrow_{\mathrm{rq}} R$.
Any string in $L(G_0) \setminus \{\lambda\}$ is contained in $\{a\}L(G_a)$ for some $a \in T$. Moreover it can be obtained via a leftmost derivation of G_a. This derivation can be simulated by G_1 as follows. We start from the axiom acS_a and simulate the rules of the form $A \to BC$, $AB \to AC$, $A \to \lambda$ as in the previous proof.

A rule $A \to a$ can be simulated only by the rules 11–13. The simulation starts only if A is the right neighbor of c in the generated string. Therefore, A must be the leftmost unmarked nonterminal from N. If there are marked symbols to the left of A, then A must bubble to the left using rules 5–8 before the simulation of the rule $A \to a$.

The whole process is repeated until all the nonterminals in the string are marked by \$, and then the rule 14 is applied once. Then a generated string belongs to the set $xdc(\$N_c \cup \$N' \cup \$\overline{N} \cup \#)^*$, for an $x \in L(G_0)$, and therefore $x \in L' \longrightarrow_{\mathrm{rq}} R$.

If (and only if) $\lambda \in L(G_0)$, then starting from the axiom dcS we can produce a string in $dc(\$N_c \cup \$N' \cup \$\overline{N} \cup \#)^*$, and therefore $\lambda \in L' \longrightarrow_{\mathrm{rq}} R$.

(ii) $L' \longrightarrow_{\mathrm{rq}} R \subseteq L(G_0)$.

We have already shown in the proof of Theorem 4 that the only thing the rules 2–10 can do is to simulate a derivation of G_0. The new rules 11–13 can only simulate a leftmost application of rules of the form $A \to a$, as follows by (3) and the above explanation. Any sentential form obtained before an application of rule 14 will produce the empty set by right quotient with R, and can therefore be discounted. If, after an application of the rule 14, all nonterminals are marked, then the result of the right quotient with R is a correct string from $L(G_0)$. Otherwise again an empty set results.

□

A proof can be easily adapted to use the *left quotient* instead of the right quotient if we replace all the axioms and rules by their mirror images. In the case of rules this also means that the left context is replaced by a mirror image of the right context, and vice versa.

4 Application in DNA computing

It is known that contextual insertion and/or deletion of in DNA frequently occurs in certain cell types. For instance, it is essential for operations with plasmides; is occurs also during a transfer between micronucleus and macronucleus in cilliates. Therefore this operation has been formalized within the DNA computing framework. The following definition is due to [7].

Definition 6 *An* insdel system *is a construct* $\gamma = (V, T, A, R)$, *where* V *is an alphabet,* $T \subseteq V$, A *is a finite language over* V, *and* R *is a finite set of triples of the form* $(u, \alpha/\beta, v)$, *where* $u, v \in V^*, (\alpha, \beta) \in (V^+ \times \{\lambda\}) \cup (\{\lambda\} \times V^+)$.

The elements of T are *terminal symbols*, those of A are *axioms*, the triples in R are *insertion-deletion rules.* The meaning of $(u, \lambda/\beta, v)$ is that β can be inserted in between u and v; the meaning of $(u, \alpha/\lambda, v)$ is that α can be deleted from the context (u, v). Stated otherwise, $(u, \lambda/\beta, v)$ corresponds to the rewriting rule $uv \to u\beta v$, and $(u, \alpha/\lambda, v)$ corresponds to the rewriting rule $u\alpha v \to uv$. These rules are applied in the manner usual in insertion and deletion grammars. The generated language is the set of all terminal words which can be obtained from the axiom set A by an application of the rules in R.

Furthermore, there is a hierarchy of the insdel systems due to the length of the context in rules. The *weight* of an insdel system $\gamma = (V, T, A, R)$ is a four-tuple $(n, m; p, q)$, where

$$
\begin{aligned}
&n = \max\{|\beta| \mid (u, \lambda/\beta, v) \in R\},\\
&m = \max\{|u| \mid (u, \lambda/\beta, v) \in R \text{ or } (v, \lambda/\beta, u) \in R\},\\
&p = \max\{|\alpha| \mid (u, \alpha/\lambda, v) \in R\},\\
&q = \max\{|u| \mid (u, \alpha/\lambda, v) \in R \text{ or } (v, \alpha/\lambda, u) \in R\}.
\end{aligned}
$$

The expression $INS_n^m DEL_p^q$, for $n, m, p, q \geq 0$, denotes the family of languages generated by insdel systems of weight $(n', m'; p', q')$ such that $n' \leq n$, $m' \leq m, p' \leq p, q' \leq q$.

Several universality results has been shown for insdel systems in [7]. For instance, the classes $INS_1^1 DEL_2^0$ and $INS_1^2 DEL_1^1$ both equal to the class of recursively enumerable languages. For the insdel systems without deletion rules, however, only results analogous to those at [6] have been shown. Therefore we can improve Theorem 6.10 in [7] as follows.

Theorem 7 *For each recursively enumerable language L there exists a morphism h, a weak coding g and a language $L_1 \in INS_3^3 DEL_0^0$ such that $L = g(h^{-1}(L_1))$.*

The statement follows immediately by Theorem 4 and its proof. Similarly our Theorem 5 improves Corollary 6.1 in [7] as follows.

Corollary 8 *There exists a regular language R with the following property: for each recursively enumerable language L there is a language $L' \in INS_3^3 DEL_0^0$ such that $L = L' \longrightarrow_{\mathrm{rq}} R$.*

Finally, Corollary 3 in [6] together with Corollary 6.2 in [7] can be strengthened as follows.

Corollary 9 *All families S_n and all families $INS_m^n DEL_0^0$, $m, n \geq 3$, are incomparable with each family F where $LIN \subseteq F \subset RE$ and F is closed under weak codings and inverse morphisms, or under left/right quotient with regular languages.*

5 Conclusion

We have characterized the class of recursively-enumerable languages by using pure insertion grammars, filtered via an inverse morphism and a weak coding, or by a right quotient. Notice that both constructions in Theorems 4 and 5 are effectively computable. This is not the case e.g. in [6] where the construction leading to Corollary 2 uses a non-computable axiom set.

Our results improve previously known lower bounds on the necessary size of context in universal insertion grammars. We have shown that insertion grammars with context of the length $n \geq 3$ are universal generators in the above explained sense. It has been shown in [6] that with the length of context $n \leq 1$, the languages generated using the above mentioned filters belong to the context-free class. For context of the length $n = 2$, it is only known that the class S_2 is strictly contained in CS but incomparable with CF, and its characterization remains an *open problem.*

Acknowledgements

This research was supported by the Canada Research Chair Grant and NSERC Discovery Grant to L.K., and by the Grant Agency of Czech Republic, grant No. 201/02/P079 to P.S.

References

[1] M. Daley, L. Kari, G. Gloor, R. Siromoney, Circular contextual insertions/ deletions with applications to biomolecular computation. *Proceedings of SPIRE'99, 6th International Symposium on String Processing and Information Retrieval*, Cancun, Mexico, IEEE Computer Society Press, Los Alamitos, California, 1999, 47-54.

[2] B. S. Galiukschov, Semicontextual grammars (in Russian). *Mat. Logica i Mat. Ling.* (1981), 38–50.

[3] L.Kari, G.Thierrin, Contextual insertions/ deletions and computability. *Information and Computation* **131-1** (1996), 47–61.

[4] M. Madhu, K. Krithivasan, A. S. Reddy, On Characterizing Recursively Enumerable Languages by Insertion Grammars. Technical report of IIIT, Hyderabad, and submitted for journal publication.

[5] S. Marcus, Contextual grammars. *Rev. Roum. Math. Pures Appl.* **14** (1969), 1525–1534.

[6] C. Martin-Vide, Gh. Păun, A. Salomaa, A characterization of recursively enumerable languages by means of insertion grammars. Theoretical Computer Science, 205(1–2):195–205, 1998.

[7] G. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms* (Springer-Verlag, Berlin, 1998).

[8] A. Salomaa, G. Rozenberg (Eds.), *Handbook of Formal Languages.* Springer-Verlag, Berlin, 1997.