Contextual insertions/deletions and computability *

Lila Kari and Gabriel Thierrin

Department of Computer Science, University of Western Ontario, N6A 5B7 London, Ontario, Canada, email: lila@csd.uwo.ca

Abstract

We investigate two generalizations of insertion and deletion of words, that have recently become of interest in the context of molecular computing. Given a pair of words (x, y) called a *context*, the (x, y)-contextual insertion of a word v into a word u is performed as follows. For each occurrence of xy as a subword in u, we include in the result of the contextual insertion the words obtained by inserting v into u, between x and y. The (x, y)-contextual deletion operation is defined in a similar way. We study closure properties of the Chomsky families under the defined operations, contextual ins-closed and del-closed languages and decidability of existence of solutions to equations involving these operations. Moreover, we prove that every Turing machine can be simulated by a system based entirely on contextual insertions and deletions.

1 Introduction

Besides being fundamental in formal language theory, the operations of insertion and deletion have recently become of interest in connection with the topic of molecular computing.

The area of molecular computing was born in 1994 when Adleman, [1], succeeded to solve an instance of the Directed Hamiltonian Path Problem solely by manipulating DNA strands. This marked the first instance where a mathematical problem could be solved by biological means and gave rise to a couple of interesting problems: a) can *any* algorithm be simulated by means of DNA manipulation, and b) is it possible, at least in theory, to design a programmable molecular computer? To answer these questions, various models of molecular computation have been proposed, and for some of these models it has been shown that the bio-operations involved can simulate the actions of a Turing machine (see, for example, [2], [4], [8], [6], [14], [17], [18]).

*This research was supported by Grant OGP0007877 of the Natural Sciences and Engineering Research Council of Canada

In this paper we focus on the formal language operations of contextual insertion and deletion. Besides their being theoretically interesting, one of the motivations for studying these operations is that they can be used as the sole primitives needed for modeling DNA computation and, moreover, they are already implementable in the laboratory. Indeed, by using available reagents and a standard technique called *PCR site-specific oligonucleotide mutagenesis* [5] one can perform insertions and deletions of nucleotide sequences. (A similar operation, substitution, has been proposed in [3] as a bio-operation necessary to simulate a universal Turing machine.) We investigate mathematical properties of contextual insertions/deletions, one of the obtained results being that we can obtain full computational power of a Turing machine by solely using these two operations.

The *contextual insertion* operation is a generalization of the catenation and insertion operations on strings and languages: words can be inserted into a string only if certain *contexts* are present. More precisely, given a set of contexts we put the condition that insertion of a word can be performed only between a pair of words in the context set. Analogously, contextual deletion allows erasing of a word only if the word is situated between a pair of words in the context set.

Section 2 deals with closure properties of the Chomsky families under contextual insertion and deletion. If the context set is finite, the families of regular and context-free languages are closed under contextual insertion and deletion with regular languages. In general, the families of context-free and context-sensitive languages are not closed under either operation.

Section 3 deals with contextual ins-closed and del-closed languages: languages with the property that the result of contextual insertion/deletion of two words in the language still belongs to the language. The contextual ins-closure/del-closure of a language L is the smallest contextual ins-closed/del-closed language containing L. Methods of constructing the contextual ins-closure and del-closure of a language are given.

Section 4 studies properties of the contextual dipolar deletion operation (an operation that deletes from a word a prefix and a suffix the catenation of which forms the word to be deleted) necessary for solving the equations studied in the next section. Section 5 considers equations of the type $L \diamond X = R$, $Y \diamond L = R$ for L, R given languages, X, Y unknowns and $\diamond$ being either contextual insertion or contextual deletion. Based on finding left and right inverses to the given operations and on a general result on these equations (see [10]), then problems of decidability of existence of solutions to these equations are solved.

Section 6 introduces the notion of an insertion-deletion scheme. We prove that the actions of every Turing machine can be simulated entirely by using contextual insertion and deletion rules, showing thus the computational completeness of molecular systems based on these operations. The proof that contextual insertions and deletions are enough to simulate the actions of a Turing machine opens thus another possible way for designing a molecular computer that uses readily available reagents and techniques.

2 Contextual insertion and deletion

In the following X will denote an alphabet, that is, a finite nonempty set. The empty word consisting of 0 letters will be denoted by λ. X^* denotes the set of all words, and X^+ the set of all nonempty words over X, while $|X|$ denotes the cardinality of X. REG, CF, CS and RE will denote respectively the families of regular, context-free, context-sensitive and recursively enumerable languages. For further formal language notions and notations the reader is referred to [15].

The insertion operation has been studied in [9] as a generalization of catenation. Given words u and v, the *insertion* of v into u consists of all words that can be obtained by inserting v in an arbitrary position of u:

$$u \leftarrow v = \{u_1 v u_2 |\ u_1 u_2 = u, u_1, u_2 \in X^*\}.$$

The above operation is too nondeterministic for modeling the type of insertions that occur during PCR site-specific oligonucleotide mutagenesis. As the name of the procedure suggests, the insertions of oligonucleotide sequences are context-sensitive. Consequently, an attempt to better model the process is to use a modified notion of insertion so that insertion of a word takes place only if a certain context is present. This can be formalized by the notion of contextual insertion, [7], [12], [13], [16].

Let $(x, y) \in X^* \times X^*$ be a pair of words called a *context*. The (x, y)-*contextual insertion* of $v \in X^*$ into $u \in X^*$ is defined as:

$$u \leftarrow_{(x,y)} v = \{u_1 x v y u_2 | u_1, u_2 \in X^*, u = u_1 x y u_2\}.$$

If the word u does not contain xy as a subword, the result of the (x, y) contextual insertion is the empty set.

If $C \subseteq X^* \times X^*$ is a set of contexts, the C-*contextual insertion* of u into v is defined as:

$$u \leftarrow_C v = \{u_1 x v y u_2 |\ (x, y) \in C, u = u_1 x y u_2, u_1, u_2 \in X^*\}.$$

If the context set C is understood, the C-contextual insertion will be called shortly *contextual insertion*. If $C = \{\lambda\} \times \{\lambda\}$, then the C-contextual insertion amounts to the usual insertion (see [9], [10]).

The C-contextual insertion of a language $L_2 \subseteq X^*$ into a language $L_1 \subseteq X^*$ can be defined in the natural way as

$$L_1 \leftarrow_C L_2 = \bigcup_{u \in L_1, v \in L_2} (u \leftarrow_C v).$$

Proposition 2.1 *If the set of contexts C is finite, REG, CF, CS are closed under C-contextual insertion.*

Proof. Let L_1, L_2 be two languages belonging to REG (respectively CF, CS) and let $C \subseteq X^* \times X^*$ be a finite set.

For each pair $(x, y) \in C$ denote

$$L_{x,y} = (X^* x\#yX^*) \cap (L_1 \amalg \{\#\}),$$

where $\amalg$ is the shuffle operation. (If $u, v \in X^*$ then $u \amalg v = u_1v_1u_2v_2u_nv_n$, $u = u_1u_2 \ldots u_n$, $v = v_1v_2 \ldots v_n$, $u_i, v_i \in X^*$, $1 \leq i \leq n$.)

Note now that

$$L = \bigcup_{(x,y)\in C} L_{x,y} = \{u_1x\#yu_2 | \ u = u_1xyu_2 \in L_1, (x,y) \in C, u_1, u_2 \in X^*\}.$$

If we consider now the substitution defined by $s(\#) = L_2 \setminus \{\lambda\}$, $s(a) = a$ for all $a \in X$, then we have

- $L_1 \leftarrow_C L_2 = s(L)$ if $\lambda \notin L_2$,
- $L_1 \leftarrow_C L_2 = s(L) \cup [L_1 \cap (\cup_{(x,y)\in C} X^*xyX^*)]$, if $\lambda \in L_2$.

The proposition now follows as REG, CF, CS are closed under shuffle, λ-free substitution, intersection with regular languages and union. □

In a manner similar to the contextual insertion, we can define the contextual deletion: deletion of a word takes place only if certain contexts are present. More precisely, let $(x, y) \in X^* \times X^*$ be a context.

The (x, y)-*contextual deletion* of $v \in X^*$ from $u \in X^*$ is defined as:

$$u \rightarrow_{(x,y)} v = \{u_1xyu_2 | u_1, u_2 \in X^*, u = u_1xvyu_2\}.$$

If $C \subseteq X^* \times X^*$ is a set of contexts, then the C-contextual deletion of v from u is

$$u \rightarrow_C v = \{u_1xyu_2 | \ (x,y) \in C, u = u_1xvyu_2, u_1, u_2 \in X^*\}.$$

The C-contextual deletion of a language $L_2 \subseteq X^*$ from a language $L_1 \subseteq X^*$ can then be defined as

$$L_1 \rightarrow_C L_2 = \bigcup_{u\in L_1, v\in L_2} (u \rightarrow_C v).$$

If $C = \{\lambda\} \times \{\lambda\}$, then the contextual deletion amounts to the usual deletion operation (see [9], [11]).

Proposition 2.2 *If L_1, L_2 are languages over X^+, L_2 a regular one, and $(x, y) \in X^* \times X^*$ is a context, then there exists a gsm g (with erasing), depending on (x, y) and L_2 such that $L_1 \rightarrow_{(x,y)} L_2 = g(L_1)$.*

Proof. Let $A = (S, X, s_A, F, P)$ be a finite deterministic automaton which recognizes L_2 and let $x = a_1a_2 \cdots a_n$, $y = b_1b_2 \cdots b_m$, $n, m \geq 0$.

Consider the gsm $g = (S', X, X, s_0, F', P')$ where s_i, s'_j, $0 \leq i \leq n$, $0 \leq j \leq m$, are new states not in S, and

$$\begin{array}{lll}
S' = S & \cup\{s_i \mid\ 0 \leq i \leq n\} \cup \{s'_j \mid\ 0 \leq j \leq m\} & \\
P' = P & \cup\{s_0 a \longrightarrow a s_0 \mid a \in X\} & (1)\\
 & \cup\{s_i a_{i+1} \longrightarrow a_{i+1} s_{i+1} \mid 0 \leq i \leq n-1\} & (2)\\
 & \cup\{s_n a \longrightarrow s \mid s_A a \longrightarrow s \in P\} \cup \{sa \longrightarrow s'_0 \mid sa \longrightarrow s_f \in P, s_f \in F\} & (3)\\
 & \cup\{s_n a \longrightarrow s'_0 \mid\ s_A a \longrightarrow s_f \in P, s_f \in F\} & \\
 & \cup\{s'_j b_{j+1} \longrightarrow b_{j+1} s'_{j+1} \mid 0 \leq j \leq m-1\} & (4)\\
 & \cup\{s'_m a \longrightarrow a s'_m \mid a \in X\} & (5)\\
F' = & \{s'_m\} &
\end{array}$$

Given a word $u_1 x v y u_2$ as an input, the gsm g works as follows. Rules (1) and (5) scan respectively the subwords u_1 and u_2. Rules (2) and (4) check the appearance of x and y in the correct order. Rules (3) and the rules of P erase the word v from the input. Note that the final state can be reached only if a final state of A is reached, that is, only if a word of L_2 occurring between x and y has been erased.

From the above explanations, it follows that $L_1 \rightarrow_{(x,y)} L_2 = g(L_1)$. □

Corollary 2.1 *If the set of context C is finite, then REG and CF are closed under C-contextual deletion with regular languages.*

Proof. Let $L_1, L_2 \subseteq X^*$ be languages, L_2 a regular one, and let C be a finite subset of $X^* \times X^*$. For each $(x, y) \in C$, according to the preceding proposition, we can construct the gsm $g_{x,y}$ with the property $L_1 \rightarrow_{(x,y)} (L_2 \setminus \{\lambda\}) = g_{x,y}(L_2)$, taking care that the sets of states $S_{x,y}$ are mutually disjoint.

We have that

- $L_1 \rightarrow_C L_2 = \bigcup_{(x,y)\in C} g_{x,y}(L_1)$ if $\lambda \notin L_2$,
- $L_1 \rightarrow_C L_2 = [\bigcup_{(x,y)\in C} g_{x,y}(L_1)] \cup [L_1 \cap (\cup_{(x,y)\in C} X^* x y X^*)]$, if $\lambda \in L_2$.

The corollary now follows as REG and CF are closed under intersection with regular languages, union and gsm mapping. □

Proposition 2.3 *There exists a finite set of contexts C such that the families of context-free and context-sensitive languages are not closed under C-contextual deletion.*

Proof. If $C = \{(\lambda, \lambda)\}$ then $L_1 \rightarrow_{(\lambda,\lambda)} L_2 = L_1 \rightarrow L_2$. The proposition now follows as CF and CS are not closed under deletion (see [11]). □

3 Contextual ins- and del-closed languages

This section studies contextual ins-closed (del-closed) languages, i.e., languages with the property that the contextual insertion (deletion) of two words in the language still belongs to the language. In order to formalize the notion of a contextual ins-closed language, we define the following auxiliary notion. Let L be a language in X^* and $C \subseteq X^* \times X^*$ be a set of contexts. The language $ins_C(L)$ is defined by:

$$ins_C(L) = \{w \in X^* | \forall u \in L, u = u_1xyu_2, (x,y) \in C \Rightarrow \ u_1xwyu_2 \in L\}$$

Intuitively, $ins_C(L)$ contains all words with the property that the result of their C-contextual insertion into words of L yields words still belonging to L. If such words do not exist, then $ins_C(L) = \emptyset$.

A language L such that $L \subseteq ins_C(L)$ is called *C-ins-closed;* L is *C-ins-closed* iff $u, v \in L$, $u = u_1xyu_2, (x,y) \in C$ implies $u = u_1xvyu_2 \in L$.

The intersection $I_C(L)$ of all C-ins-closed languages containing L is a C-ins-closed language containing L; $I_C(L)$ is called the *C-ins-closure* of L. If $L \neq \emptyset$ then $I_C(L) \neq \emptyset$.

Given a language L and set of contexts C, one can construct the C-ins-closure of L by using the iterated contextual insertion. The iterated C-contextual insertion of a language L_2 into L_1 is recursively defined as:

$$L_1 \leftarrow_C^0 L_2 = L_1$$

$$\dots$$

$$L_1 \leftarrow_C^{k+1} L_2 = (L_1 \leftarrow_C^k L_2) \leftarrow_C L_2, k \geq 0,$$

$$L_1 \leftarrow_C^* L_2 = \bigcup_{k=0}^{\infty} L_1 \leftarrow_C^k L_2.$$

Proposition 3.1 *If $C \subseteq X^* \times X^*$ is a set of contexts, the C-ins-closure of a language $L \subseteq X^*$ is $I_C(L) = L \leftarrow_C^* L$.*

Proof. "$I_C(L) \subseteq L \leftarrow_C^* L$". Obvious, as $L \leftarrow_C^* L$ is C-ins-closed and L is included in $L \leftarrow_C^* L$.

"$L \leftarrow_C^* L \subseteq I_C(L)$". We show by induction on k that $L \leftarrow_C^k L \subseteq I_C(L)$. For $k = 0$ the assertion holds, as $L \subseteq I_C(L)$.

Assume that $L \leftarrow_C^k L \subseteq I_C(L)$ and consider a word $u \in L \leftarrow_C^{k+1} L = (L \leftarrow_C^k L) \leftarrow_C L$. Then $u = u_1xvyu_2$, where $(x,y) \in C$, $v \in L$ and $u_1xyu_2 \in L \leftarrow_C^k L$. As both $L \leftarrow_C^k L$ and L are included in $I_C(L)$ and $I_C(L)$ is C-ins-closed, we deduce that $u \in I_C(L)$. The induction step, and therefore the equality, are proved. □

In order to tackle similar issues regarding the contextual deletion, we define the notion of a C-subword of L.

$$Sub_C(L) = \{w \in X^* | \ \exists u_1xwyu_2 \in L \text{ with } (x,y) \in C\}.$$

For a given language $L \subseteq X^*$ and set of contexts $C \subseteq X^* \times X^*$ define

$$del_C(L) = \{w \in Sub_C(L) | \ (x,y) \in C, u_1xwyu_2 \in L \Rightarrow$$

$$u_1xyu_2 \in L\}.$$

Intuitively, $del_C(L)$ contains all the words whose C-contextual deletion from L yields words still belonging to L.

A language L such that $L \subseteq del_C(L)$ is called *C-del-closed*; L is called *C-del-closed* iff $u, v \in L$ with $u = u_1xvyu_2, (x, y) \in C$ implies $u_1xyu_2 \in L$.

The intersection $D_C(L)$ of all C-del-closed languages containing L is a C-del-closed language containing L; $D_C(L)$ is called the *C-del-closure* of L. In the following we characterize the C-del-closure of a given language L.

Define:

$$D_0(L) = L$$

$$D_1(L) = D_0(L) \rightarrow_C (D_0(L) \cup \{\lambda\})$$

$$D_2(L) = D_1(L) \rightarrow_C (D_1(L) \cup \{\lambda\})$$

$$\ldots$$

$$D_{n+1}(L) = D_n(L) \rightarrow_C (D_n(L) \cup \{\lambda\})$$

$$\ldots$$

Note that $D_i(L) \subseteq D_{i+1}(L)$ for $i \geq 0$.

Proposition 3.2 *If $C \subseteq X^* \times X^*$ is a set of contexts, the C-del-closure of L is $D_C(L) = \bigcup_{n \geq 0} D_n(L)$.*

Proof. Clearly, $L \subseteq D_C(L)$. Let now $v \in D_C(L)$ and $u_1xvyu_2 \in D_C(L)$, with $(x, y) \in C$. Then $v \in D_i(L)$ and $u_1xvyu_2 \in D_j(L)$ for some integers $i, j \geq 0$. If $k = \max\{i, j\}$, then $v \in D_k(L)$ and $u_1xvyu_2 \in D_k(L)$. This implies $u_1xyu_2 \in D_{k+1}(L) \subseteq D_C(L)$. Therefore $D_C(L)$ is a C-del-closed language containing L.

Let T be an C-del-closed language such that $L = D_0(L) \subseteq T$. Since T is C-del-closed, if $D_k(L) \subseteq T$ then $D_{k+1}(L) \subseteq T$. By an induction argument it follows that $D_C(L) \subseteq T$. □

4 Contextual dipolar deletion

A symmetric notion of the contextual deletion is the contextual dipolar deletion: instead of deleting a word from the "middle" of another one, we delete it from its "extremities". Contextual dipolar deletion assists in obtaining characterizations of the sets $ins_C(L)$ and $del_C(L)$ associated to a language L, that were defined in the preceding section. Moreover, the contextual dipolar deletion will play an important role in Section 5, in solving certain equations involving contextual insertion and deletion operations.

Let u, v be words in X^* and $(x, y) \in X^* \times X^*$ be a context. The (x, y)-*contextual dipolar deletion* of v from u is defined as

$$u \rightleftharpoons_{(x,y)} v = \{w \in X^* | \; u = u_1xwyu_2 \text{ and } v = u_1xyu_2\}.$$

Intuitively, the (x, y)-dipolar deletion $u \rightleftharpoons_{(x,y)} v$ erases from u a prefix ending with x and a suffix starting with y, whose catenation equals v. If $x = y = \lambda$ then the (x, y)-dipolar deletion amounts to the usual *dipolar deletion* (see [11]), denoted by $u \rightleftharpoons v$.

If C is a set of contexts, then the C-contextual dipolar deletion is defined as

$$u \rightleftharpoons_C v = \{w \in X^* |\ u = u_1 x w y u_2, v = u_1 x y u_2, (x, y) \in C\}.$$

If L_1, L_2 are languages in X^* then the C-contextual dipolar deletion of L_2 from L_1 is

$$L_1 \rightleftharpoons_C L_2 = \bigcup_{u \in L_1, v \in L_2} (u \rightleftharpoons_C v).$$

We consider in the following the closure properties of the families in the Chomsky hierarchy under contextual dipolar deletion.

Proposition 4.1 *If C is a finite set of contexts, the family of regular languages is closed under C-contextual dipolar deletion.*

Proof. We show first that the proposition holds for $L_1, L_2 \subseteq X^+$ and $C = \{(x, y)\}$.

Construct the gsm $g_1 = (\{s\}, X, X \cup X' \cup X'', s, \{s\}, P)$ where $X' = \{a' |\ a \in X\}$, $X'' = \{a'' |\ a \in X\}$, if $u = a_1 a_2 \dots a_n$ then $u' = a_1' a_2' \dots a_n'$, and

$$P = \{sa \longrightarrow a's, sa \longrightarrow a''s, sa \longrightarrow a |\ a \in X\}.$$

Note that the set $g_1(L_1) \cap [(X')^* x' X^* y'' (X'')^*]$ equals

$$\{u_1' x' v y'' u_2'' |\ u = u_1 x v y u_2, u \in L_1, u_1, v, u_2 \in X^*\}.$$

If we construct now the gsm $g_2 = (\{s'\}, X, X' \cup X'', s', \{s'\}, P')$ with

$$P' = \{s'a \longrightarrow a's', s'a \longrightarrow a''s'\}$$

then one can show that

$$g_2(L_2) \cap [(X')^* x' y'' (X'')^*] = \{u_1' x' y'' u_2'' |\ u = u_1 x y u_2 \in L_2, u_1, u_2 \in X^*\}.$$

Note that the following equality holds:

$$L_1 \rightleftharpoons_{(x,y)} L_2 = [g_1(L_1) \rightleftharpoons g_2(L_2)] \cap X^*.$$

If $L_1, L_2 \subseteq X^+$ then the proposition follows as REG is closed under dipolar deletion (see [9]), intersection, union, and

$$L_1 \rightleftharpoons_C L_2 = \bigcup_{(x,y) \in C} (L_1 \rightleftharpoons_{(x,y)} L_2).$$

To complete the proof for the case $L_1, L_2 \in X^*$, note that if $\lambda \in L_2$ and $(\lambda, \lambda) \in C$ then

$$L_1 \rightleftharpoons_C L_2 = [(g_1(L_1 \setminus \{\lambda\}) \rightleftharpoons g_2(L_2 \setminus \{\lambda\})) \cap X^*] \cup L_1$$

while otherwise,

$$L_1 \rightleftharpoons_C L_2 = [g_1(L_1 \setminus \{\lambda\}) \rightleftharpoons g_2(L_2 \setminus \{\lambda\})] \cap X^*.$$

The assertion holds now for any $L_1, L_2 \in X^*$ as REG is closed under union. □

Proposition 4.2 *There exists a finite set of contexts C such that CF and CS are not closed under C-contextual dipolar deletion.*

Proof. If $C = \{(\lambda, \lambda)\}$ then $L_1 \rightleftharpoons L_2 = L_1 \rightleftharpoons_{(\lambda,\lambda)} L_2$ and CF, CS are not closed under dipolar deletion, [9]. □

The operation of C-contextual dipolar deletion enables the characterization of the sets $ins_C(L)$ and $del_C(L)$ introduced in Section 3.

Proposition 4.3 *If L is a language over X and $C \subseteq X^* \times X^*$ is a set of contexts then $ins_C(L) = (L^c \rightleftharpoons_C L)^c$ (L^c denotes the complement of the language L).*

Proof. Take $w \in ins_C(L)$. Assume, for the sake of contradiction, that $w \notin (L^c \rightleftharpoons_C L)^c$. Then $w \in L^c \rightleftharpoons_C L$, that is, there exist $(x, y) \in C$, and $u_1xwyu_2 \in L^c$, $u_1xyu_2 \in L$. We arrived at a contradiction as $u_1xyu_2 \in L$, $w \in ins_C(L)$, but $u_1xwyu_2 \notin L$.

Consider now a word $w \in (L^c \rightleftharpoons_C L)^c$. If $w \notin ins_C(L)$, there exist $u_1xyu_2 \in L$ such that $u_1xwyu_2 \in L^c$. This further implies $w \in L^c \rightleftharpoons_C L$ – a contradiction with the original assumptions about w. □

Corollary 4.1 *If $C \subseteq X^* \times X^*$ is a finite set of contexts and L is a regular language, then $ins_C(L)$ is regular.*

Proposition 4.4 *Given a language $L \subseteq X^*$ and set of contexts $C \subseteq X^* \times X^*$,*

$$del_C(L) = (L \rightleftharpoons_C L^c)^c \cap Sub_C(L).$$

Proof. Let $w \in del_C(L)$. By definition, $w \in Sub_C(L)$. Assume that $w \in L \rightleftharpoons_C L^c$. This means there exists $(x, y) \in C$, and $u_1xwyu_2 \in L$, $u_1xyu_2 \in L^c$. We arrived at a contradiction as $w \in del_C(L)$ but $u_1xyu_2 \notin L$.

For the converse inclusion, let $w \in (L \rightleftharpoons_C L^c)^c \cap Sub_C(L)$. As $w \in Sub_C(L)$, if $w \notin del_C(L)$, there exists $u_1xwyu_2 \in L$ with $u_1xyu_2 \in L^c$. This further implies that $w \in L \rightleftharpoons_C L^c$ - a contradiction. □

5 Language equations

In this section we study language equations of the type $L \diamond X = R$, $Y \diamond L = R$, where L, R are given languages and $\diamond$ denotes either a contextual insertion or a contextual deletion operation.

In the same way subtraction (an "inverse" of addition) is needed to solve numerical equations of the type $a+x=b$, solving language equations $L\diamond X = R$ involves finding an "inverse" of the language operation $\diamond$.

After defining the notion of a left inverse and right inverse of a language operation, we will use the results of [10] and of the preceding sections to construct solutions to the equations, in case they exist.

Let $\diamond$, ∇ be two binary word operations. The operation ∇ is said to be the *left inverse* (respectively *right inverse*) of the operation $\diamond$ if, for all words u, v, w over the alphabet X, the following relation holds:

$$w \in (u \diamond v) \text{ if and only if } u \in (w \nabla v).$$

(respectively, $w \in (u \diamond v)$ if and only if $v \in (u \nabla w)$.)

If $\diamond$ is a binary word (language) operation, then the reversed $\diamond$ is the operation defined by $u \diamond^r v = v \diamond u$. Recall the following results concerning solutions of language equations (see [10]):

Proposition 5.1 *Let L, R be languages over an alphabet X and $\diamond$, ∇ be two binary operations right inverses (left inverses) to each other. If the equation $L\diamond X = R$ (resp. $Y \diamond L = R$) has a solution, then also the language $R' = (L\nabla R^c)^c$ (resp. $R'' = (R^c \nabla L)^c$) is a solution. Moreover, R' (resp. R'') includes all the other solutions of the equation.*

Based on this proposition, we will be able to find solutions to the equations involving contextual insertion and deletion, provided we find the right and left inverses of these operations.

Proposition 5.2 *(a) The left inverse of the C-contextual insertion is the C-contextual deletion, while its right inverse is the reversed C-contextual dipolar deletion.*

(b) The left inverse of the C-contextual deletion is the C-contextual insertion and its right inverse is the C-contextual dipolar deletion.

Proof. (a) Let (x, y) be a context in C. The word w is in $u \leftarrow_{(x,y)} v$ iff $w = u_1xvyu_2$ with $u = u_1xyu_2$ which is equivalent to $u \in w \rightarrow_{(x,y)} v$ and also to $v \in (w \rightleftharpoons_{(x,y)} u)$. This shows that the C-contextual deletion is the left inverse of the C-contextual insertion and that reversed C-contextual dipolar deletion is the right inverse of the C-contextual insertion.

The first part of (b) follows from (a). For the second part, note that $w \in (u \rightarrow_{(x,y)} v)$ iff $u = u_1xvyu_2$ and $w = u_1xyu_2$ which is equivalent to $v \in u \rightleftharpoons_{(x,y)} w$. □

Proposition 5.3 *If C is a finite set of contexts, the problem "Does there exist a solution to the equation $L \leftarrow_C X = R$? (resp $L \rightarrow_C X = R$, $Y \leftarrow_C L = R$, $Y \rightarrow_C L = R$)" is decidable for regular languages L and R.*

Proof. It follows from Proposition 5.1, Proposition 5.2 and the fact that REG is closed under all the involved operations and their inverses (see Proposition 2.1, Corollary 2.1 and Proposition 4.1).

Moreover, in case a solution to the equation exists, then we can effectively construct a maximal solution to the equation.

The solutions are respectively

$$X_{max} = (R^c \rightleftharpoons_C L)^c \text{ for } L \leftarrow_C X = R,$$
$$Y_{max} = (R^c \rightarrow_C L)^c \text{ for } Y \leftarrow_C L = R,$$
$$X_{max} = (L \rightleftharpoons_C R^c)^c \text{ for } L \rightarrow_C X = R,$$
$$Y_{max} = (R^c \leftarrow_C L)^c \text{ for } Y \rightarrow_C L = R.$$

□

Recall that, if we choose as set of context $C = \{(\lambda, \lambda)\}$ then contextual insertion (contextual deletion, contextual dipolar deletion) amounts to ordinary (deletion, dipolar deletion). It is known (see [10]) that, if $\diamond$ denotes insertion or deletion, the problems of existence of solutions to the equations $L \diamond X = R$, $Y \diamond L = R$ are undecidable for context-free languages L and regular languages R. Consequently, in these cases, the existence of solutions will be undecidable also for the contextual versions of insertion and deletion.

6 Insertion and deletion schemes

The purpose of this section is to prove that any Turing machine can be simulated by using only context-sensitive insertions and deletions. With this in mind, we first define the notions of an insertion/deletion scheme, and then proceed to show that if a language is acceptable by a Turing machine, we can effectively construct an insdel systems that accepts the same language.

An *insertion scheme* INS is a pair $INS = (X, I)$ where X is an alphabet with $|X| \geq 2$ and $I \subseteq X^* \times X^* \times X^*$, $I \neq \emptyset$. The elements of I are denoted by $(x, z, y)_I$ with $x, y, z \in X^*$ and are called the *contextual insertion rules* of the scheme. For every word $u \in X^*$, let

$$cins_I(u) = \{v \in X^* | v \in u \leftarrow_{(x,y)} z, (x, z, y)_I \in I\}$$

(Informally, in a contextual insertion rule (x, z, y), the pair (x, y) represents the context of insertion while z is the word to be inserted.) To simplify, we can use the notation $cins(u)$ instead of $cins_I(u)$ when there is no possible ambiguity. If $L \subseteq X^*$ and I is fixed, then

$$cins(L) = \{cins(u) | u \in L\}.$$

A *deletion scheme* DEL is a pair $DEL = (X, D)$ where X is an alphabet with $|X| \geq 2$ and $D \subseteq X^* \times X^* \times X^*$, $D \neq \emptyset$. The elements of D are denoted by $(x, z, y)_D$ and are called the *contextual deletion rules* of the scheme. For every word $u \in X^*$, let

$$cdel_D(u) = \{v \in X^* | v \in u \to_{(x,y)} z, (x, z, y)_D \in D\}$$

(In a contextual deletion rule (x, z, y), the pair (x, y) represents the context of deletion while z is the word to be deleted.) To simplify, we can use the notation $cdel(u)$ instead of $cdel_D(u)$ when there is no possible ambiguity. If $L \subseteq X^*$ and D is fixed, then

$$cdel(L) = \{cdel(u) | u \in L\}.$$

An *insdel scheme* is a triple $ID = (X, I, D)$ where X is an alphabet with $|X| \geq 2$, I is a set of insertion rules and D is a set of deletion rules.

An *insdel system* ID is a quintuple:

$$ID = (X, T, I, D, \omega)$$

where X is an alphabet with $|X| \geq 2$, (X, I) is an insertion scheme, (X, D) is a deletion scheme, I, D are finite, $T \subseteq X$ is the terminal alphabet, and $\omega \in X^+$ is a fixed word called the axiom of the insdel system.

If $u \in X^*$ and $v \in cins(u) \cup cdel(u)$, then v is said to be directly ID-derived from u and this derivation is denoted by $u \Longrightarrow v$. The sequence of direct derivations:

$$u_1 \Longrightarrow u_2 \Longrightarrow \ldots \Longrightarrow u_k, k \geq 1$$

is denoted by $u_1 \Longrightarrow^* u_k$ and u_k is said to be derived from u_1.

The *language* $L_g(ID)$ *generated* by the insdel system ID is the set:

$$L_g(ID) = \{v \in T^* | \omega \Longrightarrow^* v \text{ where } \omega \text{ is the axiom}\}$$

and analogously we can define the *language* $L_a(ID)$ *accepted* by the insdel system as

$$L_a(ID) = \{v \in T^* | \ v \Longrightarrow^* \omega, \text{ where } \omega \text{ is the axiom}\}$$

Recall that, [15], a rewriting system $(S, X \cup \{\#\}, F)$ is called a *Turing machine* iff the following conditions are satisfied.

(i) S and $X \cup \{\#\}$ (with $\# \notin X$ and $X \neq \emptyset$) are two disjoint alphabets referred to as the *state* and *tape* alphabet.

(ii) Elements $s_0 \in S$, $\flat \in X$, and a subset $S_f \subseteq S$ are specified, namely, the *initial state*, the *blank symbol*, and the *final state set*. A subset $V_f \subseteq X$ is specified as the *final alphabet*.

(iii) The productions in F are of the forms

(1) $s_i a \longrightarrow s_j b$ overprint
(2) $s_i ac \longrightarrow as_j c$ move right
(3) $s_i a\# \longrightarrow as_j \flat \#$ move right and extend workspace
(4) $cs_i a \longrightarrow s_j ca$ move left
(5) $\#s_i a \longrightarrow \#s_j \flat a$ move left and extend the workspace

where $s_i, s_j \in S$ and $a, b, c \in X$. Furthermore, for each $s_i, s_j \in S$ and $a \in X$, F either contains no productions (2) and (3) (resp. (4) and (5)) or else contains both (2) and (3) (respectively (4), (5)) for every $c \in X$. For no $s_i \in S$ and $a \in X$, the word $s_i a$ is a subword of the left side in two productions of the forms (1), (3) and (5).

We say that a word sw, where $s \in S$ and $w \in (X \cup \{\#\})^*$ is *final* iff w does not begin with a letter a such that sa is a subword of the left side of some production in F. The language *accepted* by a Turing machine TM is defined by

$$L(TM) = \{w \in V_f^* \mid \ \#s_0 w\# \Longrightarrow^* \#w_1 s_f w_2\# \text{ for some } s_f \in S_f,$$
$$w_1, w_2 \in X^* \text{ such that } s_f w_2\# \text{ is final}\}$$

where $\Longrightarrow$ denotes derivation according to the rewriting rules (1) – (5) of the Turing machine. A language is *acceptable* by a Turing machine iff $L = L(TM)$ for some TM. It is to be noted that TM is *deterministic*: at each step of the rewriting process, at most one production is applicable.

Proposition 6.1 *If a language is acceptable by a Turing machine TM, then there exists an insdel system ID accepting the same language.*

Proof. Let TM be a Turing machine $TM = (S, X \cup \{\#\}, F)$ as described above. We will construct an insdel system $ID = (N, T, I, D, Y_0)$ such that the language accepted by the insdel system is $L_a(ID) = L(TM)$. The alphabet of ID is $N = S \cup X \cup \{\#\} \cup \{O, L, R, Y_0, Y_1, Y_2\}$, where O, L, R, Y_0, Y_1, Y_2 are new symbols not appearing in $S \cup X$. The terminal alphabet is $T = V_f$, the axiom is Y_0, and the contextual insertion and deletion rules are defined as follows.

(a) For each rule of the Turing machine TM, insertion and deletion rules are added to the insdel system in the following fashion, where a, b, c are letters in X, $x, y \in X \cup X^2 \cup X^3 \cup \{\#\}$, $z \in X^2 \cup \{\#\}X$, and $r, t \in X^*$:

a1. For each *rule (1)* $s_i a \longrightarrow s_j b$ (overprint) of F, we add to the insdel system the rules $(xs_i a, s_j Ob, y)_I$, $(x, s_i a, s_j Oby)_D$ and $(zs_j, O, by)_D$.

Hence, if $u = \#rxs_i ayt\#$, then rule *(1)* of TM can be simulated by the following derivation in ID:

$$\#rxs_i ayt\# \Longrightarrow \#rxs_i as_j Obyt\# \Longrightarrow \#rxs_j Obyt\# \Longrightarrow \#rxs_j byt\#$$

a2. For each *rule (2)* $s_i ac \longrightarrow as_j c$ (move right) of F, we add to ID the rules $(xs_i a, s_j R, cy)_I$, $(x, s_i, as_j Rcy)_D$ and $(zas_j, R, cy)_D$.

Hence, if $u = \#rxs_iacyt\#$, then rule *(2)* of TM can be simulated by the following derivation in ID:

$$\#rxs_iacyt\# \Longrightarrow \#rxs_ias_jRcyt\# \Longrightarrow \#rxas_jRcyt\# \Longrightarrow \#rxas_jcyt\#.$$

a3. For each *rule (3)* $s_ia\# \longrightarrow as_j \flat \#$ (move right and extend workspace) of F, we add to ID the rules $(xs_ia, s_jR \flat, \#)_I$, $(x, s_i, as_jR \flat \#)_D$, and $(xas_j, R, \flat \#)_D$.

If $u = \#rxs_ia\#$, then rule *(3)* of TM can be simulated by the following derivation in ID:

$$\#rxs_ia\# \Longrightarrow \#rxs_ias_jR \flat \# \Longrightarrow \#rxas_jR \flat \# \Longrightarrow \#rxas_j \flat \#.$$

a4. For each *rule (4)* $cs_ia \longrightarrow s_jca$ (move left) of F, we add to ID the rules $(x, s_jL, cs_iay)_I$, $(xs_jLc, s_i, ay)_D$, $(xs_j, L, cay)_D$.

Hence, if $u = \#rxcs_iayt\#$, then rule *(4)* of TM can be simulated by the following derivation in ID:

$$\#rxcs_iayt\# \Longrightarrow \#rxs_jLcs_iayt\# \Longrightarrow \#rxs_jLcayt\# \Longrightarrow \#rxs_jcayt\#.$$

a5. For each *rule (5)* $\#s_ia \longrightarrow \#s_j \flat a$ (move left and extend workspace) of F we add to ID the rules $(\#, s_jL \flat, s_iay)_I$, $(\#s_jL \flat, s_i, ay)_D$, $(\#s_j, L, \flat ay)_D$.

Hence, if $u = \#s_iayt\#$ then rule *(5)* of TM can be simulated by the following derivation in ID:

$$\#s_iayt\# \Longrightarrow \#s_jL \flat s_iayt\# \Longrightarrow \#s_jL \flat ayt\# \Longrightarrow \#s_j \flat ayt\#.$$

In addition to the rules above, that simulate the rewriting of the Turing machine by insertions and deletion rules, we introduce the following rules **(b)**:

$$\begin{array}{ll}
(b1) & (\lambda, \#s_0, b)_I, (b, \#, \lambda)_I \\
(b2) & (s_f, Y_1, a)_I, (s_f, Y_1, \#)_I \\
(b3) & (c, s_f, Y_1)_D, (\#, s_f, Y_1)_D \\
(b4) & (Y_1, b, c)_D, (Y_1, b, \#)_D \\
(b5) & (b, Y_2, Y_1\#)_I, (\#, Y_2, Y_1\#)_I \\
(b6) & (Y_2, Y_1, \#)_D \\
(b7) & (b, c, Y_2)_D, (\#, b, Y_2)_D \\
(b8) & (\#, Y_0, Y_2\#)_I \\
(b9) & (\lambda, \#, Y_0)_D, (Y_0, Y_2\#, \lambda)_D \\
(b10) & (\lambda, \#s_0\#, \lambda)_I, (\#s_f, Y_2, \#)_I, (\#, s_f, Y_2\#)_D
\end{array}$$

where s_f ranges over S_f, b, c range over X, and for each s_f, a ranges over such elements of X that s_fa is final. It can now be verified that $L_a(ID) = L(TM)$.

Indeed, if $w \in L(TM)$ then $w \in T^*$ and there exists a derivation

$$\#s_0w\# \Longrightarrow^* \#w_1s_fw_2\# \qquad (*)$$

for some $s_f \in S_f$, $w_1, w_2 \in X^*$, $s_f w_2\#$ final. To show that $w \in L_a(ID)$ we must find a derivation $w \Longrightarrow^* Y_0$ according to the rules of ID. If $w \neq 1$, such a derivation is the following:

$$w \overset{(b1)}{\Longrightarrow} \#s_0 w\# \overset{(a)}{\Longrightarrow}{}^* \#w_1 s_f w_2\# \overset{(b2)}{\Longrightarrow} \#w_1 s_f Y_1 w_2\# \overset{(b3)}{\Longrightarrow} \#w_1 Y_1 w_2\# \overset{(b4)}{\Longrightarrow}$$

$$\#w_1 Y_1\# \overset{(b5)}{\Longrightarrow} \#w_1 Y_2 Y_1\# \overset{(b6)}{\Longrightarrow} \#w_1 Y_2\# \overset{(b7)}{\Longrightarrow} \#Y_2\# \overset{(b8)}{\Longrightarrow} \#Y_0 Y_2\# \overset{(b9)}{\Longrightarrow} Y_0,$$

where $\overset{(a)}{\Longrightarrow}{}^*$ represents a simulation of the derivation (*) of the Turing machine by the rules **(a)**.

If $w = \lambda$, the required derivation is the following:

$$\lambda \overset{(b10)}{\Longrightarrow} \#s_0\# \overset{(b10)}{\Longrightarrow} \#Y_2\# \overset{(b8)}{\Longrightarrow} \#Y_0 Y_2\# \overset{(b9)}{\Longrightarrow} Y_0.$$

Assume, conversely that $w \in L_a(ID)$. If $w = \lambda$ there is a derivation according to ID from $\#s_f\#$ to Y_0 where $s_f \in S_f$ and $s_f = s_0$. This implies that $\lambda \in L(TM)$. If $w \neq \lambda$ then, according to the way the rules **(a)** were constructed, there is a derivation according to ID from

$$\#w_1 s_f a w_2'\#, \qquad s_f \in S_f, a \in X, w_1, w_2' \in X^*, s_f a \text{ final}, \qquad (**)$$

to Y_0, and also a derivation from w to $(**)$, according to rules (b). This implies that $w \in L(TM)$. $\square$

References

[1] L.Adleman. Molecular computation of solutions to combinatorial problems. *Science* vol.266, Nov.1994, 1021–1024.

[2] L.Adleman. On constructing a molecular computer. *ftp: /ftp/pub/csinfo/papers/adleman/molecular_computer.ps.*

[3] D.Beaver. A universal molecular computer. *Proceedings of the DIMACS workshop on DNA-based computing*, Princeton, April 1995.

[4] D.Boneh, R.Lipton, C.Dunworth, J.Sgall. On the computational power of DNA. *http://www.cs.princeton.edu/˜ dabo.*

[5] C.W.Dieffenbach, G.S.Dveksler, Eds. *A laboratory manual*, Cold Spring Harbor Laboratory Press, 1995, 581-621.

[6] R.Freund, L.Kari, G.Paun. DNA computing based on splicing: the existence of universal computers. Technical Report 185-2/FR-2/95, TU Wien, Institute for Computer Languages, 1995, also *http://www.csd.uwo.ca/˜lkari.*

[7] B.S.Galiukschov. Semicontextual grammars (in Russian). *Mat.logica i mat. ling.*, Kalinin Univ., 1981, 38-50.

[8] T.Head. Formal language theory and DNA: an analysis of the generative capacity of recombinant behaviors. *Bulletin of Mathematical Biology*, 49(1987), 737-759.

[9] L.Kari. *On insertions and deletions in formal languages.* PhD thesis, University of Turku, Finland, 1991.

[10] L.Kari. On language equations with invertible operations. *Theoretical Computer Science*, 132(1994), 129-150.

[11] L.Kari. Deletion operations: closure properties. *International Journal of Computer Mathematics*, 52(1994), 23-42.

[12] G.Paun. On semicontextual grammars. *Bull. Math. Soc. Sci. Math. Roumanie*, 28(76), 1984, 63-68.

[13] G.Paun. Two theorems about Galiukschov grammars. *Kybernetika*, 21(1985), 360-365.

[14] P.Rothemund. A DNA and restriction enzyme implementation of Turing machines. Abstract at *http://www.ugcs.caltech.edu/˜pwkr/oett.html.*

[15] A.Salomaa. *Formal Languages.* Academic Press, New York, 1973.

[16] C.C.Squier. Semicontextual grammars: an example. *Bull. Math. Soc. Sci. Math. Roumanie*, 32(80), 1988, 167-170.

[17] W.Smith, A.Schweitzer. DNA computers in vitro and in vivo. NEC Technical Report, 3/20/95.

[18] E.Winfree. On the computational power of DNA annealing and ligation. *http://dope/caltech.edu/winfree/DNA.html.*