On Some Properties of Controlled Shuffle

Karina Chuda[1,*,†], Branislav Rovan[1,†]

[1]Department of Computer Science, Faculty of Mathematics, Physics and Informatics, Comenius University, Mlynská dolina F1, 842 48 Bratislava, Slovakia

Abstract

Several approaches to use a number of grammars to jointly generate a word were already studied in the literature (e.g., Cooperating distributed grammar systems, Parallel communicating grammar systems). When considering words in languages generated by such systems of grammars, it is possible to identify subwords generated by a particular grammar. Based on the known operation of shuffle, a new operation on languages - the controlled shuffle - is introduced and studied in this paper. The motivation came from the study of usefulness of additional information in problem solving in the formalism of grammars. Unlike the ordinary shuffle operation, it allows to shuffle the words in a predefined way. It is shown that families of languages closed under controlled shuffle are also closed under basic language operations. Moreover, properties of the closure of the basic language families in the Chomsky hierarchy under controlled shuffle are presented.

Keywords

formal language, controlled shuffle, operation on languages, closure property

1. Introduction

This work is motivated by the study of usefulness of information in problem solving. The information is useful if it helps to solve a problem easier. So far, it was studied in the formalism of automata. A problem, a language L_{prob}, is solved by an automaton A_{prob} accepting this language. A supplementary information (advice) allows to assume the input word satisfies some property (belongs to an advice language L_{adv} accepted by an automaton A_{adv}) so, for determining whether the input word belongs to L_{prob}, it suffices to check for fewer properties by a simpler automaton A_{new} accepting L_{new}. Thus the task can be viewed as a decomposition of an automaton A_{prob} into two simpler automata A_{new} and A_{adv} such that $L_{prob} = L_{new} \cap L_{adv}$ (see, e.g., [9] for motivation and results). In [1], the study of usefulness of information using grammars is introduced. The solution to a problem L_{prob} is given by a grammar G_{prob}. A supplementary information provided enables a simpler grammar G_{new} to generate just parts of the word in L_{prob} the remaining parts to be filled in by the advice grammar G_{adv}. Thus, to obtain a word in L_{prob}, some words in L_{new} and L_{adv} are shuffled in an appropriate way. Here, the general shuffle operation known from the literature may result in words not in the original L_{prob}. This leads to introducing a new operation - the controlled shuffle - studied in this paper. It allows to specify how two grammars can cooperate on producing a word consisting of parts generated by each of the grammars.

The idea of using several grammars to jointly generate a word already appeared in the literature. Cooperating distributed grammar systems [6] and Parallel communicating grammar systems [8] (among others) are grammar systems where several grammars are used to jointly generate a word. In the words generated by such grammar systems, subwords generated by individual grammars can be identified. The new binary operation - controlled shuffle - also keeps this property. In contrast to the ordinary shuffle, parameters of the controlled shuffle, called the control symbols, determine how the subwords are shuffled to create a single resulting word. By generating these control symbols, the grammar G_{new} indicates positions in the word generated at which it expects parts of the word generated by G_{adv} to be inserted to obtain a word in L_{prob}. Similarly, the control symbols generated by G_{adv} indicate how the word generated

ITAT 2026: Information Technologies - Applications and Theory, September 25-29, 2026, Biele Karpaty, Slovakia

*Corresponding author.

†These authors contributed equally.

karina.chuda@fmph.uniba.sk (K. Chuda); rovan@dcs.fmph.uniba.sk (B. Rovan)

0000-0002-0015-403X (B. Rovan)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-09

CEUR-WS.org/Vol-4323/paper20.pdf

is to be split in parts to fit in the places indicated in the word generated by G_{new}. Clearly, only some pairs of words are suitable to fulfil the expectations of both grammars. The notion of compatibility of pairs of words shall be explained later. The role of the control symbols, a placeholder, a separator, and an empty, can be assigned to any triple of pairwise distinct symbols not occurring in the alphabet of L_{prob}.

It is to be noted that several approaches (using different motivation) to limit the result of the general shuffle operation appeared in the literature. In [7], a generalization of the rewrite step in grammars is considered limiting the result of the shuffle of two words to be in a given regular language. In [5], the result of the shuffle is determined by explicitly specifying the number of letters to be taken from the first word, then the number of letters taken from the second word, and again the number of letters from the first word, etc., using the notion of a trajectory (in the two dimensional grid). The shuffle on trajectories was shown to be able to simulate various types on restrictions on shuffle. It might be of interest to explore the relation between the shuffle on trajectories and the controlled shuffle studied in this paper.

Following the definition of the controlled shuffle in Section 2, the properties of the controlled shuffle are studied in Section 3. It is shown that the controlled shuffle is a powerful operation. It is shown that a family of languages closed under controlled shuffle is (under mild assumptions) also closed under the usual basic language operations. Finally, closures of basic language families in the Chomsky hierarchy under controlled shuffle are investigated in Section 4.

Standard notations of the formal languages theory are used throughout the paper, for instance the notations introduced in [4, 3]. To abbreviate the presentation and to enhance the readability of the text, some conventions are kept unless stated otherwise.

The control symbols are always assumed to be pairwise distinct. The following dedicated symbols are used as the control symbols, the placeholder denoted by $\sqcup$, the separator denoted by $\top$, and the empty denoted by $\ni$. The dedicated control symbols are omitted from the notation of related notions. The use of other symbols in the role of control symbols is explicitly mentioned.

All (words and) languages are assumed to be over the alphabet Σ. It is convenient to assume that the control symbols are in Σ.

The alphabet Σ_0 denotes the set of symbols in Σ which are not control symbols, more precisely $\Sigma_0 = \Sigma - \{\sqcup, \top, \ni\}$. (In the simulations of operations, it is assumed that the control symbols do not occur in the argument languages. This can be achieved by using new symbols that do not occur in the argument languages.) The alphabet Σ_0 is assumed to have k_0 elements a_i for $1 \leq i \leq k_0$.

In the process of defining the controlled shuffle, a set of special symbols $_$, $\llcorner$, $\lrcorner$, $\ulcorner$, and $\urcorner$ is used. These special symbols are assumed to be pairwise distinct and not in Σ_0. The alphabet Σ_M denotes Σ_0 together with the special symbols, more precisely $\Sigma_M = \Sigma_0 \cup \{_, \llcorner, \lrcorner, \ulcorner, \urcorner\}$.

It shall often be useful to mark the symbols in Σ_0. For each a in Σ_0 a new symbol a' is used. The $'$ notation shall be used in a natural way. Thus, for $\Sigma_0 = \left\{a_1, a_2, a_3, \ldots, a_{k_0}\right\}$, the alphabet of primed symbols is $\Sigma_0{}' = \left\{a_1{}', a_2{}', a_3{}', \ldots, a_{k_0}{}'\right\}$. Similarly, for a word w in $\Sigma_0{}^*$, the word w' is obtained by using the primed version of the symbols in w. Naturally, for a language L, the language L' consists of the primed versions of the words in L.

2. Controlled shuffle

The way the controlled shuffle operates can be explained by describing the roles of the control symbols. The placeholder $\sqcup$ in one word is replaced by a substring over Σ_0 from the other word. The separator $\top$ is used to separate neighboring substrings. The empty $\ni$ stands in for the empty string. The control symbols vanish in the shuffling process. A simple example of the controlled shuffle follows. It is further used in a step by step explanation of the controlled shuffle. The word in the first row and the word in the second row are shuffled into the word in the third row.

$$\begin{array}{l} ab\ \sqcup\ \ \sqcup\ \ni \\ \sqcup\ cac\ \top\ a\ \sqcup \\ \hline ab\ cac\ \ a \end{array}$$

The controlled shuffle can be explained in two steps. A pair of words can be shuffled only in case they are compatible. Thus the first step in the explanation describes the compatibility relation. The second step in the explanation describes the controlled shuffle itself.

Images of words by an a-transducer as defined, e.g., in [3] shall be used to explain the compatibility. It is a nondeterministic finite state transducer with accepting states. Observe that an a-transducer can be applied to any language however only input words leading to an accepting state have a nonempty output.

The image of a word by the a-transducer M (depicted below) is a language of words without the control symbols also containing the new pairwise distinct symbols ␣, ⌞, ⌟, ⌜, ⌝ not in Σ_0. The words of the language consist of possibly empty subwords not containing the control symbols enclosed between ⌜ and ⌝ and of possibly empty subwords of variable length containing only symbols ␣ enclosed between ⌞ and ⌟.

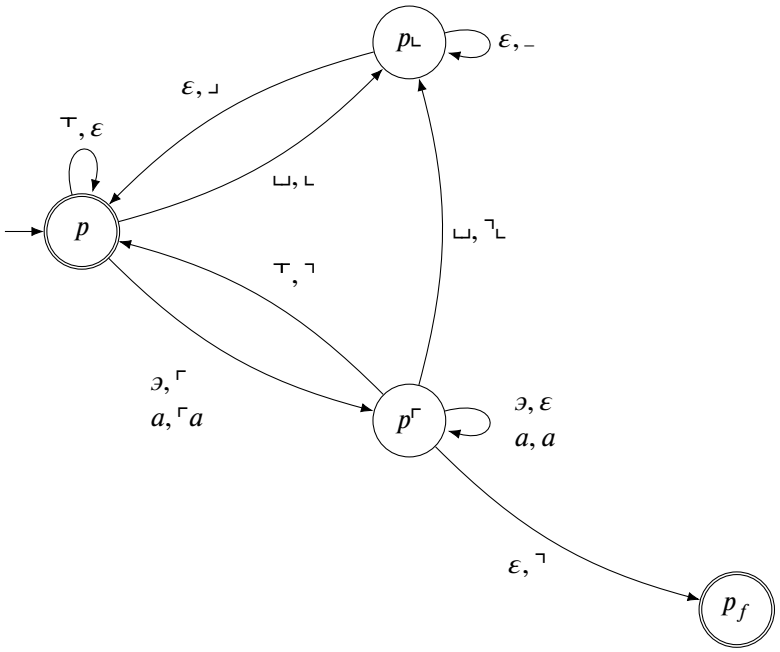
The images of words from the previous example by M are as follows.

$M\,(ab␣␣ə) = \left\{ ⌜\, a\, b\, ⌝\, ⌞\, ␣^{i_1}\, ⌟\, ⌞\, ␣^{i_2}\, ⌟\, ⌜\, ⌝ \middle| i_1, i_2 \in \mathbb{N} \right\}$

$M\,(␣cac⊤a␣) = \left\{ ⌞\, ␣^{i_1}\, ⌟\, ⌜\, c\, a\, c\, ⌝\, ⌜\, a\, ⌝\, ⌞\, ␣^{i_2}\, ⌟ \middle| i_1, i_2 \in \mathbb{N} \right\}$

The a-transducer M has states p, $p_⌞$, $p^⌜$, p_f where the state p is the initial state and the states p and p_f are final states. If the a-transducer is in the state p (and in the state p_f) then the output consists of subwords enclosed either between ⌞ and ⌟ or between ⌜ and ⌝. If the a-transducer is in the state $p_⌞$ then the possibly empty subword containing only symbols ␣ delimited by ⌞ from the left is produced. If the a-transducer is in the state $p^⌜$ then the possibly empty subword not containing the control symbols delimited by ⌜ from the left is produced. At the end of an input word ending with a non control symbol, the a-transducer can enter the final state p_f.

The a-transducer M, formally defined in Notation 2.1, can be given by a diagram having the transition arrows labeled with pairs of input symbol and output word. The diagram is simplified, the symbol a stands for all a distinct from the control symbols and two transitions sharing the beginning state and the ending state are depicted simply by one arrow with two label pairs.



Notation 2.1. Denote by M the a-transducer $\left(Q, \Sigma, \Delta, H, q_0, F\right)$ such that the set of states $Q = \left\{p, p^⌜, p_⌞, p_f\right\}$, the input alphabet is Σ, the output alphabet $\Delta = \Sigma_M$ (see Σ_M defined in Section 1), the transitions $H = \{(p, ⊤, \varepsilon, p), (p, ␣, ⌞, p_⌞), (p, ə, ⌜, p^⌜)\} \cup \left\{(p, a, ⌜a, p^⌜) \middle| a \in \Sigma_0\right\} \cup \{(p_⌞, \varepsilon, ␣, p_⌞), (p_⌞, \varepsilon, ⌟, p)\} \cup \{(p^⌜, ⊤, ⌝, p), (p^⌜, ␣, ⌝⌞, p_⌞), (p^⌜, ə, \varepsilon, p^⌜)\} \cup \left\{(p^⌜, a, a, p^⌜) \middle| a \in \Sigma_0\right\} \cup \left\{\left(p^⌜, \varepsilon, ⌝, p_f\right)\right\}$, the initial state $q_0 = p$, and the set of final states $F = \left\{p, p_f\right\}$.

Note that the image of a word by the nondeterministic a-transducer M is a (nonempty) language such that the only difference between its words is the number of symbols ␣ to which a particular occurrence of

$\sqcup$ is mapped (due to the nondeterministic behavior of the state $q_{\llcorner}$). A pair of words u and v is compatible if and only if the lengths of the subwords consisting of symbols $_$ enclosed between $\llcorner$ and $\lrcorner$ in the outputs $\mathbb{u}_v$ and $\mathbb{v}_u$ by M for the inputs v and u can be chosen in a way that any subword enclosed between $\llcorner$ and $\lrcorner$ in one of the outputs is of the same length as the corresponding subword enclosed between $\ulcorner$ and $\urcorner$ in the other of the outputs. Note that if it is possible to choose such lengths then the $\mathbb{u}_v$ and $\mathbb{v}_u$ are unique since the length of a subword enclosed between $\llcorner$ and $\lrcorner$ in one of the words is determined by the length of the the corresponding subword enclosed between $\ulcorner$ and $\urcorner$ in the other word.

While the translation of a word u by the a-transducer M is almost deterministic in the sense that the words in the image $M(u)$ of u by M differ only in the lengths of the sequences of symbols $_$ (as explained above), distinct words u_1 and u_2 can have the same image $M(u_1) = M(u_2)$ by the a-transducer M (due to the handling of the symbols $\top$ and $\ni$ by M).

The words given in the example are compatible.

$\ulcorner a\ b \urcorner \llcorner _\ _\ _ \lrcorner \llcorner _ \lrcorner \ulcorner \urcorner \in M$ ($ab\ \sqcup\ \ \sqcup\ \ni$)

$\llcorner _\ _\ _ \lrcorner \ulcorner c\ a\ c \urcorner \ulcorner a \urcorner \llcorner \lrcorner \in M$ ($\sqcup\ cac\ \top\ a\ \sqcup$)

To give an exact definition of the compatibility, it is useful to introduce the following alphabet of paired symbols, with the pairs written in columns, and projections to each of the levels.

Notation 2.2. Denote by Π the alphabet $\left\{ \begin{bmatrix} \ulcorner \\ \llcorner \end{bmatrix}, \begin{bmatrix} \llcorner \\ \ulcorner \end{bmatrix}, \begin{bmatrix} \urcorner \\ \lrcorner \end{bmatrix}, \begin{bmatrix} \lrcorner \\ \urcorner \end{bmatrix} \right\} \cup \left\{ \begin{bmatrix} a \\ _ \end{bmatrix} \middle| a \in \Sigma_0 \right\} \cup \left\{ \begin{bmatrix} _ \\ a \end{bmatrix} \middle| a \in \Sigma_0 \right\}$.

Denote by π_1 the homomorphism from Π^* to ${\Sigma_M}^*$ such that $\pi_1\left(\begin{bmatrix} a \\ b \end{bmatrix}\right) = a$ for each $\begin{bmatrix} a \\ b \end{bmatrix} \in \Pi$.

Denote by π_2 the homomorphism from Π^* to ${\Sigma_M}^*$ such that $\pi_2\left(\begin{bmatrix} a \\ b \end{bmatrix}\right) = b$ for each $\begin{bmatrix} a \\ b \end{bmatrix} \in \Pi$.

Definition 2.3. Let u an v be words.

The *compatibility word* $\mathbb{w}_{uv}$ *of* u *and* v is the word over Π such that $\pi_1\left(\mathbb{w}_{uv}\right)$ is a word in $M(u)$ and $\pi_2\left(\mathbb{w}_{uv}\right)$ is a word in $M(v)$.

The words u and v are *compatible*, denoted by $u \perp v$, if there exists the compatibility word $\mathbb{w}_{uv}$ of u and v.

As mentioned above, if the words u and v are compatible then there exist unique output $\mathbb{u}_v$ for the input u by M and unique output $\mathbb{v}_u$ for the input v by M stacking up to the compatibility word $\mathbb{w}_{uv}$. In other words, if the words u and v are compatible then there exist unique output $\mathbb{u}_v$ for the input u and unique output $\mathbb{v}_u$ of v such that there exists the compatibility word $\mathbb{w}_{uv}$ over Π with $\pi_1\left(\mathbb{w}_{uv}\right) = \mathbb{u}_v$ and $\pi_2\left(\mathbb{w}_{uv}\right) = \mathbb{v}_u$. Thus $\mathbb{w}_{uv}$ is uniquely determined by u and v if u and v are compatible and there exists no such word if u and v are not compatible.

Although the compatibility word $\mathbb{w}_{uv}$ is uniquely determined for any pair of compatible words u and v, distinct pairs of compatible words, say the pair u_1 and v_1 and the pair u_2 and v_2, may yield the same compatibility word $\mathbb{w}_{u_1v_1} = \mathbb{w}_{u_2v_2}$, since distinct words can have the same image by M as discussed above.

The compatibility word of the words given in the example used above follows.

$\begin{bmatrix} \ulcorner \\ \llcorner \end{bmatrix} \begin{bmatrix} a \\ _ \end{bmatrix} \begin{bmatrix} b \\ _ \end{bmatrix} \begin{bmatrix} \urcorner \\ \lrcorner \end{bmatrix} \begin{bmatrix} \llcorner \\ \ulcorner \end{bmatrix} \begin{bmatrix} _ \\ c \end{bmatrix} \begin{bmatrix} _ \\ a \end{bmatrix} \begin{bmatrix} _ \\ c \end{bmatrix} \begin{bmatrix} \lrcorner \\ \urcorner \end{bmatrix} \begin{bmatrix} \llcorner \\ \ulcorner \end{bmatrix} \begin{bmatrix} _ \\ a \end{bmatrix} \begin{bmatrix} \lrcorner \\ \urcorner \end{bmatrix} \begin{bmatrix} \ulcorner \\ \llcorner \end{bmatrix} \begin{bmatrix} \urcorner \\ \lrcorner \end{bmatrix}$

To explain the controlled shuffle itself, a homomorphism $\mathfrak{h}$ on the compatibility word is used.

Notation 2.4. Denote by $\mathfrak{h}$ the homomorphism from Π^* to $(\Sigma - \{\sqcup, \top, \ni\})^*$ such that $\mathfrak{h}\left(\begin{bmatrix} \ulcorner \\ \llcorner \end{bmatrix}\right) = \varepsilon$, $\mathfrak{h}\left(\begin{bmatrix} \llcorner \\ \ulcorner \end{bmatrix}\right) = \varepsilon$, $\mathfrak{h}\left(\begin{bmatrix} \urcorner \\ \lrcorner \end{bmatrix}\right) = \varepsilon$, $\mathfrak{h}\left(\begin{bmatrix} \lrcorner \\ \urcorner \end{bmatrix}\right) = \varepsilon$, $\mathfrak{h}\left(\begin{bmatrix} a \\ _ \end{bmatrix}\right) = a$ for each $a \in \Sigma_0$, and $\mathfrak{h}\left(\begin{bmatrix} _ \\ a \end{bmatrix}\right) = a$ for each $a \in \Sigma_0$.

Note that the homomorphism $\mathfrak{h}$ shall be used on compatible words only.

The image under $\mathfrak{h}$ of the compatibility word of the given example is as follows.

$\mathfrak{h}\left(\begin{bmatrix} \ulcorner \\ \llcorner \end{bmatrix} \begin{bmatrix} a \\ _ \end{bmatrix} \begin{bmatrix} b \\ _ \end{bmatrix} \begin{bmatrix} \urcorner \\ \lrcorner \end{bmatrix} \begin{bmatrix} \llcorner \\ \ulcorner \end{bmatrix} \begin{bmatrix} _ \\ c \end{bmatrix} \begin{bmatrix} _ \\ a \end{bmatrix} \begin{bmatrix} _ \\ c \end{bmatrix} \begin{bmatrix} \lrcorner \\ \urcorner \end{bmatrix} \begin{bmatrix} \llcorner \\ \ulcorner \end{bmatrix} \begin{bmatrix} _ \\ a \end{bmatrix} \begin{bmatrix} \lrcorner \\ \urcorner \end{bmatrix} \begin{bmatrix} \ulcorner \\ \llcorner \end{bmatrix} \begin{bmatrix} \urcorner \\ \lrcorner \end{bmatrix}\right) = abcaca$

Using the homomorphism $\mathfrak{h}$ on the compatibility word, the controlled shuffle can thus be defined as the image of the compatibility word under $\mathfrak{h}$.

The words from the example shuffle to the following word.

$$ab\sqcup\sqcup\mathfrak{d}\sqcup\!\sqcup c a c^{\top} a\sqcup = \mathfrak{h}\left(\begin{bmatrix}\ulcorner\\ \llcorner\end{bmatrix}\begin{bmatrix}a\\ _\end{bmatrix}\begin{bmatrix}b\\ _\end{bmatrix}\begin{bmatrix}\urcorner\\ \lrcorner\end{bmatrix}\begin{bmatrix}\llcorner\\ \ulcorner\end{bmatrix}\begin{bmatrix}_\\ c\end{bmatrix}\begin{bmatrix}_\\ a\end{bmatrix}\begin{bmatrix}_\\ c\end{bmatrix}\begin{bmatrix}\lrcorner\\ \urcorner\end{bmatrix}\begin{bmatrix}\llcorner\\ \ulcorner\end{bmatrix}\begin{bmatrix}_\\ a\end{bmatrix}\begin{bmatrix}\lrcorner\\ \urcorner\end{bmatrix}\begin{bmatrix}\ulcorner\\ \llcorner\end{bmatrix}\begin{bmatrix}\urcorner\\ \lrcorner\end{bmatrix}\right) = abcaca$$

Definition 2.5. Let u and v be compatible words. The *controlled shuffle* $u\sqcup\!\sqcup v$ *of words* u *and* v is the word $u\sqcup\!\sqcup v = \mathfrak{h}\left(\mathbb{w}_{uv}\right)$ over Σ_0 for each $u \perp v$. It is not defined otherwise.

Let L_1 and L_2 be languages. The *controlled shuffle* $L_1\sqcup\!\sqcup L_2$ *of languages* L_1 *and* L_2 is the language $L_1\sqcup\!\sqcup L_2 = \left\{u\sqcup\!\sqcup v \middle| u \in L_1, v \in L_2, u \perp v\right\}$ over Σ_0.

A purely algebraic definition of the controlled shuffle (without an a-transducer) can be found in [1].

3. Properties of controlled shuffle

It is shown that the controlled shuffle is a powerful operation. In particular, a family of languages closed under controlled shuffle is (under mild assumptions) also closed under the usual basic language operations. This follows from the fact that the controlled shuffle can be used to simulate several operations on languages, namely concatenation, union, intersection, homomorphism, and substitution. In the simulations, the original languages are first replaced by auxiliary languages derived from the original languages. These languages are derived by simple means such as mappings by elementary homomorphisms (renaming symbols) or simple operations (concatenation and union with a singleton language) up to an exception discussed later. In particular, the derived languages are in the Chomsky family of the original languages. The presented simulations vary in difficulty. The simulation of concatenation and of union require one controlled shuffle while the simulation of intersection (being the hardest), homomorphism, and substitution use a sequence of controlled shuffles depending on the size of the underlying alphabets.

Since the controlled shuffle requires a non empty string to be matched with $\sqcup$, a language $L_{\mathfrak{d},\varepsilon}$ constructed from the respective language L is often used instead of L itself.

Notation 3.1. Let L be a language over Σ_0 and let $h_{\varepsilon,\mathfrak{d}}$ be the homomorphism from $\left(\Sigma_0 \cup \{\mathfrak{d}\}\right)^*$ to ${\Sigma_0}^*$ such that $h_{\varepsilon,\mathfrak{d}}(\mathfrak{d}) = \varepsilon$ and $h_{\varepsilon,\mathfrak{d}}(a) = a$ for each $a \in \Sigma_0$.

Denote by $w_{\mathfrak{d},\varepsilon}$ any word such that $w_{\mathfrak{d},\varepsilon}$ is not ε and $h_{\varepsilon,\mathfrak{d}}\left(w_{\mathfrak{d},\varepsilon}\right) = w$.

Denote by $L_{\mathfrak{d},\varepsilon}$ any language such that ε is not in $L_{\mathfrak{d},\varepsilon}$ and $h_{\varepsilon,\mathfrak{d}}\left(L_{\mathfrak{d},\varepsilon}\right) = L$.

If $w = \varepsilon$ then the role of $w_{\mathfrak{d},\varepsilon}$ can be played by any word from the preimage of w under $h_{\varepsilon,\mathfrak{d}}$ except w itself and by no other word, hence it can be any word from $\{\mathfrak{d}\}^+$. If $w \neq \varepsilon$ then the role of $w_{\mathfrak{d},\varepsilon}$ can be played by any word from the preimage of w under $h_{\varepsilon,\mathfrak{d}}$ including w itself and by no other word. For instance, $ab_{\mathfrak{d},\varepsilon}$ can be chosen to be not only ab, but also $ab\mathfrak{d}$ or any other word from $\{\mathfrak{d}\}^* a \{\mathfrak{d}\}^* b \{\mathfrak{d}\}^*$.

The most straightforward is the simulation of concatenation.

Notation 3.2. Let L_1 and L_2 be languages over Σ_0. Denote by $L_{1\,2}$ the language $L_{1\,2} = L_{1\mathfrak{d},\varepsilon}\{\sqcup\} \sqcup\!\sqcup \{\sqcup\} L_{2\mathfrak{d},\varepsilon}$.

Theorem 3.3. Let L_1 and L_2 be languages over Σ_0. Then $L_1 L_2 = L_{1\,2}$.

Proof. Take the assumptions from Theorem 3.3 and Notation 3.2.

Let w be in $L_{1\,2}$. Hence there exist $u_{\mathfrak{d},\varepsilon}$ in $L_{1\mathfrak{d},\varepsilon}$ and $v_{\mathfrak{d},\varepsilon}$ in $L_{2\mathfrak{d},\varepsilon}$ such that $w = u_{\mathfrak{d},\varepsilon}\sqcup\!\sqcup\sqcup v_{\mathfrak{d},\varepsilon}$. Furthermore, there exist u in L_1 and v in L_2 such that $h_{\varepsilon,\mathfrak{d}}\left(u_{\mathfrak{d},\varepsilon}\right) = u$ and $h_{\varepsilon,\mathfrak{d}}\left(v_{\mathfrak{d},\varepsilon}\right) = v$. Since u is in L_1 and v is in L_2, uv is in $L_1 L_2$. Thus $w = u_{\mathfrak{d},\varepsilon}\sqcup\!\sqcup\sqcup v_{\mathfrak{d},\varepsilon} = h_{\varepsilon,\mathfrak{d}}\left(u_{\mathfrak{d},\varepsilon}\right) h_{\varepsilon,\mathfrak{d}}\left(v_{\mathfrak{d},\varepsilon}\right) = uv$ is in $L_1 L_2$.

Let w be in $L_1 L_2$. Hence there exist u in L_1 and v in L_2 such that $w = uv$. Furthermore, there exist $u_{\mathfrak{d},\varepsilon}$ in $L_{1\mathfrak{d},\varepsilon}$ such that $h_{\varepsilon,\mathfrak{d}}\left(u_{\mathfrak{d},\varepsilon}\right) = u$ and $v_{\mathfrak{d},\varepsilon}$ in $L_{2\mathfrak{d},\varepsilon}$ such that $h_{\varepsilon,\mathfrak{d}}\left(v_{\mathfrak{d},\varepsilon}\right) = v$. The words $u_{\mathfrak{d},\varepsilon}\sqcup$ and $\sqcup v_{\mathfrak{d},\varepsilon}$ are compatible therefore $u_{\mathfrak{d},\varepsilon}\sqcup\!\sqcup\sqcup v_{\mathfrak{d},\varepsilon}$ is in $L_{1\,2}$. Thus $w = uv = h_{\varepsilon,\mathfrak{d}}\left(u_{\mathfrak{d},\varepsilon}\right) h_{\varepsilon,\mathfrak{d}}\left(v_{\mathfrak{d},\varepsilon}\right) = u_{\mathfrak{d},\varepsilon}\sqcup\!\sqcup\sqcup v_{\mathfrak{d},\varepsilon}$ is in $L_{1\,2}$. □

The simulation of union is slightly more involved, but similar to the concatenation case.

Notation 3.4. Let L_1 and L_2 be languages over Σ_0. Denote by $L_{1\cup 2}$ the language $L_{1\cup 2} = \left(L_{1\textschwa,\varepsilon}\,\{\sqcup\} \cup \{\sqcup\textschwa\}\right) ⧢ \left(L_{2\textschwa,\varepsilon}\,\{\sqcup\} \cup \{\sqcup\textschwa\}\right)$.

Theorem 3.5. Let L_1 and L_2 be languages over Σ_0. Then $L_1 \cup L_2 = L_{1\cup 2}$.

Proof. Take the assumptions from Theorem 3.5 and Notation 3.4.

Observe that $u_{\textschwa,\varepsilon}\sqcup$ is only compatible with $\sqcup\textschwa$ for each $u_{\textschwa,\varepsilon}$ in $L_{1\textschwa,\varepsilon}$ and $v_{\textschwa,\varepsilon}\sqcup$ is only compatible with $\sqcup\textschwa$ for each $v_{\textschwa,\varepsilon}$ in $L_{2\textschwa,\varepsilon}$ and that $\sqcup\textschwa$ and $\sqcup\textschwa$ are not compatible.

Let w be in $L_{1\cup 2}$. There exists either $u_{\textschwa,\varepsilon}$ in $L_{1\textschwa,\varepsilon}$ such that $w = u_{\textschwa,\varepsilon}\sqcup ⧢ \sqcup\textschwa$ or $v_{\textschwa,\varepsilon}$ in $L_{2\textschwa,\varepsilon}$ such that $w = \sqcup\textschwa ⧢ v_{\textschwa,\varepsilon}\sqcup$. In the first case, there exists u in L_1 such that $h_{\varepsilon,\textschwa}\left(u_{\textschwa,\varepsilon}\right) = u$. Since u is in L_1, u is in $L_1 \cup L_2$. Thus $w = u_{\textschwa,\varepsilon}\sqcup ⧢ \sqcup\textschwa = h_{\varepsilon,\textschwa}\left(u_{\textschwa,\varepsilon}\right) = u$ is in $L_1 \cup L_2$. The second case is proven analogously.

Let w be in $L_1 \cup L_2$. There exists either u in L_1 such that $w = u$ or v in L_2 such that $w = v$. In the first case, there exists $u_{\textschwa,\varepsilon}$ in $L_{1\textschwa,\varepsilon}$ such that $h_{\varepsilon,\textschwa}\left(u_{\textschwa,\varepsilon}\right) = u$. The words $u_{\textschwa,\varepsilon}\sqcup$ and $\sqcup\textschwa$ are compatible therefore $u_{\textschwa,\varepsilon}\sqcup ⧢ \sqcup\textschwa$ is in $L_{1\cup 2}$. Thus $w = u = h_{\varepsilon,\textschwa}\left(u_{\textschwa,\varepsilon}\right) = u_{\textschwa,\varepsilon}\sqcup ⧢ \sqcup\textschwa$ is in $L_{1\cup 2}$. The second case is proven analogously. □

The simulations of intersection, homomorphism, and substitution proceed using several controlled shuffles, about one controlled shuffle is needed for each symbol from the underlying alphabet. During the simulation, the already processed symbols and the yet to be processed symbols have to be distinguished. One way to deal with the task is to mark all symbols at the beginning of the simulation and to erase the marks on the symbols as they are processed.

While marking the symbols is just a technical solution to prevent mismatch in processed and unprocessed symbols, the following considerations explain the core of the presented simulations of intersection, homomorphism, and substitution.

Two transformations of a given language L are considered with focus on subwords consisting of symbols $\sqcup$. Such subword shall be called $\sqcup$ maximal if it is delimited by symbols other than $\sqcup$ or the beginning, resp. the end of the word. Note that a word may contain several $\sqcup$ maximal subwords possibly having different lengths. First, for a language L, the symbol $\sqcup$, a positive integer n, and a word v, the language derived from L in the following way is considered. Any word in L containing a $\sqcup$ maximal subword of length not divisible by n is discarded. Any word in L containing $\sqcup$ maximal subwords of possibly various lengths divisible by n is transformed. In the transformation, each $\sqcup$ maximal subword, which is necessarily $\sqcup^{kn}$ for some $k \in \mathbb{N}$, is replaced by v^k while the rest of the word is kept without change. Second, given a language L, the symbol $\sqcup$, a positive integer n, and a language Λ, the language derived from L in the same way as before is considered except that $\sqcup^{kn}$ is replaced by Λ^k.

The concept is formalized using a symbol α. In the simulations, special cases are used where α is identified either with v or with $\sqcup^n$. The case ($n = 2$ and) $\alpha = v$ is used in the simulation of intersection where $\sqcup^2$ is replaced by α. The case $\alpha = \sqcup^n$ implying $n = 1$ is used in the simulation of homomorphism where α is replaced by v and in the simulation of substitution where α is replaced by Λ. While the intended use in the simulation of intersection might not be obvious, in the case of the homomorphism and substitution simulations, the replacement of α by the respective v or Λ has clearly a lot in common with a homomorphism or substitution mapping each but one symbol to itself or to the language which contains the mapped symbol as the only word, respectively. The general case is presented to prevent discussing three similar results using nearly identical definitions and proofs.

To formalize the concept, the symbol α mentioned and the homomorphisms $h_{\sqcup^n,\alpha}$ and $h_{v,\alpha}$ are used. The symbol α is mapped to $\sqcup^n$ by $h_{\sqcup^n,\alpha}$ and to v by $h_{v,\alpha}$ while each of the remaining symbols is mapped to itself by both homomorphisms. In terms of the homomorphisms, the language L is transformed to the image under $h_{v,\alpha}$ of the preimage under $h_{\sqcup^n,\alpha}$ of L. Replacing the word v by the language Λ, the homomorphism $h_{v,\alpha}$ has to be replaced by the substitution $\tau_{\Lambda,\alpha}$ such that α is mapped to Λ and each of the remaining symbols is mapped to the language which contains this symbol as the only word. The language L is then transformed to the image under $\tau_{\Lambda,\alpha}$ of the preimage under $h_{\sqcup^n,\alpha}$ of L.

Notation 3.6. Let L be a language over $\Sigma_0 \cup \{\sqcup\}$, let α be a symbol not in Σ_0, let $n \in \mathbb{N} - \{0\}$, and let $h_{\sqcup^n,\alpha}$ be the homomorphism from $\left(\Sigma_0 \cup \{\alpha\}\right)^*$ to $\left(\Sigma_0 \cup \{\sqcup\}\right)^*$ such that $h_{\sqcup^n,\alpha}(\alpha) = \sqcup^n$ and $h_{\sqcup^n,\alpha}(a) = a$ for each $a \in \Sigma_0$.

Let υ be a word over Δ and let $h_{\upsilon,\alpha}$ be the homomorphism from $\left(\Sigma_0 \cup \{\alpha\}\right)^*$ to $\left(\Sigma_0 \cup \Delta\right)^*$ such that $h_{\upsilon,\alpha}(\alpha) = \upsilon$ and $h_{\upsilon,\alpha}(a) = a$ for each $a \in \Sigma_0$. Denote by $\mathcal{T}_{\upsilon,\sqcup^n}(L)$ the language $\mathcal{T}_{\upsilon,\sqcup^n}(L) = \left\{h_{\upsilon,\alpha}(w) \middle| w \in \left(\Sigma_0 \cup \{\alpha\}\right)^*, h_{\sqcup^n,\alpha}(w) \in L\right\}$.

Let Λ be a language over Δ and let $\tau_{\Lambda,\alpha}$ be the substitution from $(\Sigma \cup \{\alpha\})^*$ to $\mathcal{P}\left(\left(\Sigma_0 \cup \Delta\right)^*\right)$ such that $\tau_{\Lambda,\alpha}(\alpha) = \Lambda$ and $\tau_{\Lambda,\alpha}(a) = \{a\}$ for each $a \in \Sigma_0$. Denote by $\mathcal{T}_{\Lambda,\sqcup^n}(L)$ the language $\mathcal{T}_{\Lambda,\sqcup^n}(L) = \bigcup\left(\left\{\tau_{\Lambda,\alpha}(w) \middle| w \in \left(\Sigma_0 \cup \{\alpha\}\right)^*, h_{\sqcup^n,\alpha}(w) \in L\right\}\right)$.

Note that the language $\mathcal{T}_{\upsilon,\sqcup^n}(L)$ can be obtained by the controlled shuffle from the given language L and a (regular) language depending only on the integer n, the symbol $\sqcup$, and the word υ thus $\mathcal{T}_{\upsilon,\sqcup^n}(L)$ can be used in simulations by the controlled shuffle. The main idea of obtaining $\mathcal{T}_{\upsilon,\sqcup^n}(L)$ by the controlled shuffle is to treat the symbol $\sqcup$ as the placeholder. For the controlled shuffle to work as intended, two assumptions have to be met. First, the symbol $\sqcup$ cannot appear in υ which is added to the assumptions below. Second, the word υ cannot be the empty word ε therefore $\upsilon_{\ni,\varepsilon}$ is used instead of υ. The controlled shuffle has to provide the desired outcome for each word in L. However, an arbitrary word in an arbitrary language L is hard to control therefore all the control comes from the other shuffled language $\left\{{}^\top\upsilon_{\ni,\varepsilon}{}^\top({}^\top\ni^\top)^{n-1}, \sqcup\right\}^*$ (in the string ${}^\top\upsilon_{\ni,\varepsilon}{}^\top({}^\top\ni^\top)^{n-1}$, single inner symbols ${}^\top$ and one outer symbol ${}^\top$ would suffice, the additional symbols ${}^\top$ just simplify the reasoning). In any word from the language $\left\{{}^\top\upsilon_{\ni,\varepsilon}{}^\top({}^\top\ni^\top)^{n-1}, \sqcup\right\}^*$, the strings free from the control symbols placeholder $\sqcup$ and separator ${}^\top$ come in n-tuples (uninterrupted by $\sqcup$) consisting of $\upsilon_{\ni,\varepsilon}$ and $n-1$ copies of $\ni$ therefore a compatible word in L must in turn contain placeholders $\sqcup$ in n-tuples. Thus a word in L containing a maximal sequence of symbols $\sqcup$ of length not kn for some $k \in \mathbb{N}$ is not compatible with any word in $\left\{{}^\top\upsilon_{\ni,\varepsilon}{}^\top({}^\top\ni^\top)^{n-1}, \sqcup\right\}^*$ and is discarded. Moreover, each n-tuple of placeholders $\sqcup$ in a compatible word in L is replaced by the n-tuple of strings consisting of $\upsilon_{\ni,\varepsilon}$ and $n-1$ copies of $\ni$ vanishing in the shuffling process resulting in a string υ, in other words, any maximal string $\sqcup^{kn}$ is replaced by a string υ^k for each $k \in \mathbb{N}$. Considering the language Λ instead of the word υ, the controlled shuffle produces $\mathcal{T}_{\Lambda,\sqcup^n}(L)$ in L and the language $\left(\left(\{{}^\top\}\,\Lambda_{\ni,\varepsilon}\,\{{}^\top\}\,\{{}^\top\ni^\top\}^{n-1}\right) \cup \{\sqcup\}\right)^*$ (which need not be regular and belongs to the same Chomsky family as Λ). In each word from the language $\left(\left(\{{}^\top\}\,\Lambda_{\ni,\varepsilon}\,\{{}^\top\}\,\{{}^\top\ni^\top\}^{n-1}\right) \cup \{\sqcup\}\right)^*$, the strings not containing the control symbols placeholder $\sqcup$ and separator ${}^\top$ come in n-tuples then consisting of any word in $\Lambda_{\ni,\varepsilon}$ and $n-1$ copies of $\ni$ vanishing in the shuffling process. Each maximal string $\sqcup^{kn}$ is then replaced by a string of k not necessarily identical words in Λ.

Notation 3.7. Let L be a language over $\Sigma_0 \cup \{\sqcup\}$, let Δ be an alphabet such that $\sqcup$, ${}^\top$, and $\ni$ are not in Δ, and let $n \in \mathbb{N} - \{0\}$.

Let υ be a word over Δ. Denote by $L_{\mathcal{T}\upsilon,\sqcup^n}$ the language $L_{\mathcal{T}\upsilon,\sqcup^n} = L \text{⧢} \left\{{}^\top\upsilon_{\ni,\varepsilon}{}^\top({}^\top\ni^\top)^{n-1}, \sqcup\right\}^*$.

Let Λ be a language over Δ. Denote by $L_{\mathcal{T}\Lambda,\sqcup^n}$ the language $L_{\mathcal{T}\Lambda,\sqcup^n} = L \text{⧢} \left(\left(\{{}^\top\}\,\Lambda_{\ni,\varepsilon}\,\{{}^\top\}\,\{{}^\top\ni^\top\}^{n-1}\right) \cup \{\sqcup\}\right)^*$.

Lemma 3.8. Let L be a language over $\Sigma_0 \cup \{\sqcup\}$, let Δ be an alphabet such that $\sqcup$ is not in Δ, and let $n \in \mathbb{N} - \{0\}$.

Let υ be a word over Δ. Then $\mathcal{T}_{\upsilon,\sqcup^n}(L) = L_{\mathcal{T}\upsilon,\sqcup^n}$.

Let Λ be a language over Δ. Then $\mathcal{T}_{\Lambda,\sqcup^n}(L) = L_{\mathcal{T}\Lambda,\sqcup^n}$.

Proof. Take the assumptions from Lemma 3.8, Notation 3.7, and Notation 3.6.

First $L_{\mathcal{T}\Lambda,\sqcup^n} = \mathcal{T}_{\Lambda,\sqcup^n}(L)$ is proven then $L_{\mathcal{T}\upsilon,\sqcup^n} = \mathcal{T}_{\upsilon,\sqcup^n}(L)$ follows as a corollary.

The statement $L_{\mathcal{T}\Lambda,\sqcup^n} = \mathcal{T}_{\Lambda,\sqcup^n}(L)$ is proven as follows.

Let w be in $L_{\mathcal{T}\upsilon,\sqcup^n}$. Then there exist compatible words u in L and υ in $\left(\left(\{{}^\top\}\,\Lambda_{\ni,\varepsilon}\,\{{}^\top\}\,\{{}^\top\ni^\top\}^{n-1}\right) \cup \{\sqcup\}\right)^*$ such that $w = u \text{⧢} \upsilon$. Since υ is in $\left(\left(\{{}^\top\}\,\Lambda_{\ni,\varepsilon}\,\{{}^\top\}\,\{{}^\top\ni^\top\}^{n-1}\right) \cup \{\sqcup\}\right)^*$, there exist uniquely determined $k \in \mathbb{N}$ and $\delta_i \in \mathbb{N}$ for each $0 \leq i \leq k$ and words υ_i in Λ (thus υ_i not containing $\sqcup$, ${}^\top$, and $\ni$) for each $1 \leq i \leq k$ such that $\upsilon = \sqcup^{\delta_0}{}^\top\upsilon_{1\ni,\varepsilon}{}^\top({}^\top\ni^\top)^{n-1}\sqcup^{\delta_1}{}^\top\upsilon_{2\ni,\varepsilon}{}^\top({}^\top\ni^\top)^{n-1}\sqcup^{\delta_2}{}^\top\upsilon_{3\ni,\varepsilon}{}^\top({}^\top\ni^\top)^{n-1}\ldots\sqcup^{\delta_k}$ (thus $\upsilon_{i\ni,\varepsilon} \neq \varepsilon$ not containing $\sqcup$ and ${}^\top$). Since u and υ are compatible and u is in L over $\Sigma_0 \cup \{\sqcup\}$, there exist words u_i over Σ_0 for each $0 \leq i \leq k$ such that $u = u_0\sqcup^n u_1 \sqcup^n u_2 \sqcup^n \ldots u_k$ where $u_i = \varepsilon$ if $\delta_i = 0$ and $u_i \neq \varepsilon$ if $\delta_i \neq 0$ for each

$0 \leq i \leq k$. Since u_i is over Σ_0 for each $0 \leq i \leq k$ and $\top$ is not in Σ_0, if $\delta_i \neq 0$ then $\delta_i = 1$ for each $0 \leq i \leq k$. Let $u_\alpha = u_0 \alpha u_1 \alpha u_2 \alpha \dots u_k$. It holds that u_α is over $\Sigma_0 \cup \{\alpha\}$ and $h_{\sqcup^n,\alpha}(u_\alpha) = u$ is in L therefore $w = u \text{⧢} v = u_0 v_1 u_1 v_2 u_2 v_3 \dots u_k$ is in $\tau_{\Lambda,\alpha}(u_\alpha)$ thus in $\mathcal{T}_{\Lambda,\sqcup^n}(L)$.

For the converse, let w be in $\mathcal{T}_{\Lambda,\sqcup^n}(L)$. Therefore there exists u_α over $\Sigma_0 \cup \{\alpha\}$ such that $h_{\sqcup^n,\alpha}(u_\alpha)$ is in L and w is in $\tau_{\Lambda,\alpha}(u_\alpha)$. Since u_α is over $\Sigma_0 \cup \{\alpha\}$, there exist uniquely determined $k \in \mathbb{N}$ and words u_i over Σ_0 for each $0 \leq i \leq k$ such that $u_\alpha = u_0 \alpha u_1 \alpha u_2 \alpha \dots u_k$. Since w is in $\tau_{\Lambda,\alpha}(u_\alpha)$, there exist words v_i in Λ (thus v_i not containing $\sqcup$, $\top$, and $\ni$) for each $1 \leq i \leq k$ such that $w = u_0 v_1 u_1 v_2 u_2 v_3 \dots u_k$. Let $u = h_{\sqcup^n,\alpha}(u_\alpha) = u_0 \sqcup^n u_1 \sqcup^n u_2 \sqcup^n \dots u_k$. Let $v = \sqcup^{\delta_0} \top v_{1\ni,\varepsilon} \top (\top \ni \top)^{n-1} \sqcup^{\delta_1} \top v_{2\ni,\varepsilon} \top (\top \ni \top)^{n-1} \sqcup^{\delta_2} \top v_{3\ni,\varepsilon} \top (\top \ni \top)^{n-1} \dots \sqcup^{\delta_k}$ (thus $v_{i\ni,\varepsilon} \neq \varepsilon$ not containing $\sqcup$ and $\top$) where $\delta_i = 0$ if $u_i = \varepsilon$ and $\delta_i = 1$ if $u_i \neq \varepsilon$ for each $0 \leq i \leq k$. The word v is in $\left(\left(\{\top\} \Lambda_{\ni,\varepsilon} \{\top\} \{\top \ni \top\}^{n-1}\right) \cup \{\sqcup\}\right)^*$ and u and v are compatible. Since u is in L and v is in $\left(\left(\{\top\} \Lambda_{\ni,\varepsilon} \{\top\} \{\top \ni \top\}^{n-1}\right) \cup \{\sqcup\}\right)^*$, $w = u_0 v_1 u_1 v_2 u_2 v_3 \dots v_k = u \text{⧢} v$ is in $L_{\mathcal{T}\Lambda,\sqcup^n}$.

The statement $L_{\mathcal{T}v,\sqcup^n} = \mathcal{T}_{v,\sqcup^n}(L)$ can be proven as a corollary of the statement $L_{\mathcal{T}\Lambda,\sqcup^n} = \mathcal{T}_{\Lambda,\sqcup^n}(L)$ for $\Lambda = \{v\}$.

Let v be a word over Δ. It holds that $L_{\mathcal{T}v,\sqcup^n} = L \text{⧢} \left\{\top v_{\ni,\varepsilon} \top (\top \ni \top)^{n-1}, \sqcup\right\}^* = L \text{⧢} \left(\left(\{\top\} \left\{v_{\ni,\varepsilon}\right\} \{\top\} \{\top \ni \top\}^{n-1}\right) \cup \{\sqcup\}\right)^* = L_{\mathcal{T}\{v\},\sqcup^n} = \mathcal{T}_{\{v\},\sqcup^n}(L) = \bigcup\left(\left\{\tau_{\{v\},\alpha}(w) \middle| w \in \left(\Sigma_0 \cup \{\alpha\}\right)^*, h_{\sqcup^n,\alpha}(w) \in L\right\}\right) = \left\{h_{v,\alpha}(w) \middle| w \in \left(\Sigma_0 \cup \{\alpha\}\right)^*, h_{\sqcup^n,\alpha}(w) \in L\right\} = \mathcal{T}_{v,\sqcup^n}(L)$.

For a more detailed argument, it suffices to replace the language Λ by the word v and the substitution $\tau_{\Lambda,\alpha}$ by the homomorphism $h_{v,\alpha}$ in Notation 3.6 in all detail in the proof. □

The condition on $\sqcup$ not to appear in v and Λ causes that, in some cases, $\mathcal{T}_{v,\sqcup^n}(L)$ and $\mathcal{T}_{\Lambda,\sqcup^n}(L)$ cannot be constructed in this way. These cases include, for instance, the case $v = \alpha = \sqcup^n$, namely constructing L from L. More importantly, the cases $v = \alpha = \sqcup$ which would simplify the intersection simulation and $\sqcup = \alpha \in \Sigma_0$ which might simplify the homomorphism and substitution simulation are also affected.

The intersection simulation case is used with ($n = 2$ and) $v = \alpha$ implying $h_{v,\alpha} = \mathrm{id}$. Therefore $\mathcal{T}_{\alpha,\sqcup^2}(L) = \left\{w \middle| w \in \left(\Sigma_0 \cup \Delta\right)^*, h_{\sqcup^2,\alpha}(w) \in L\right\}$. Furthermore $\alpha = v$ is over Σ_0 while $\sqcup$ is not in Σ_0 hence α and $\sqcup$ must be distinct. The parameter $n = 2$ is important later on as it indicates that sequences of symbols $\sqcup$ of even length are halved, more precisely the string $\sqcup^{2k}$ is replaced by the string α^k.

The homomorphism simulation case is used with $\sqcup^n = \alpha$ implying $h_{\sqcup^n,\alpha} = \mathrm{id}$ and $n = 1$. Therefore $\mathcal{T}_{v,\alpha}(L) = \left\{h_{v,\alpha}(w) \middle| w \in \left(\Sigma_0 \cup \{\sqcup\}\right)^*, w \in L\right\}$. Furthermore $\alpha = \sqcup$ is not in Σ_0.

Similarly, the substitution simulation case is also used with $\sqcup^n = \alpha$ implying $h_{\sqcup^n,\alpha} = \mathrm{id}$ and $n = 1$. Therefore $\mathcal{T}_{\Lambda,\alpha}(L) = \bigcup\left(\left\{\tau_{\Lambda,\alpha}(w) \middle| w \in \left(\Sigma_0 \cup \{\sqcup\}\right)^*, w \in L\right\}\right)$. Furthermore $\alpha = \sqcup$ is not in Σ_0.

It follows from the above lemma that the controlled shuffle can be used on L over $\Sigma_0 \cup \{\sqcup\}$ to replace n-tuples of the placeholder $\sqcup$ by a desired word or language. In the simulations, the lemma is iteratively used so that several (say k) symbols are in a sequence assigned the role of the placeholder $\sqcup_i$ for each $1 \leq i \leq k$. To reflect this, another transformation of L is defined determined by three k-tuples, namely $\vec{\sqcup}$ of pairwise distinct symbols, $\vec{n}$ of positive integers, and $\vec{v}$ of words or $\vec{\Lambda}$ of languages, respectively.

Notation 3.9. Let L be a language over $\Sigma_0 \cup \Psi$, let $k \in \mathbb{N}$, let Δ_i be alphabets for each $1 \leq i \leq k$ and $\Delta = \bigcup\left(\left\{\Delta_i \middle| 1 \leq i \leq k\right\}\right)$, let Ψ be an alphabet such that $|\Psi| = k$ and $\Psi = \left\{\sqcup_i \middle| 1 \leq i \leq k\right\}$, and $\Sigma_0 \cap \Psi = \emptyset$, let $\vec{\sqcup} = \left(\sqcup_1, \sqcup_2, \sqcup_3, \dots, \sqcup_k\right)$, let Γ be an alphabet such that $|\Gamma| = k$, $\Gamma = \left\{\alpha_i \middle| 1 \leq i \leq k\right\}$, and $\Sigma_0 \cap \Gamma = \emptyset$ and α_i is not from $\left(\Psi - \left\{\sqcup_i\right\}\right) \cup \left(\Delta - \Delta_i\right)$, let $\vec{\alpha} = \left(\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_k\right)$, let $\vec{n} = \left(n_1, n_2, n_3, \dots, n_k\right)$ such that $n_i \in \mathbb{N} - \{0\}$ for each $1 \leq i \leq k$, let $\overrightarrow{\sqcup^n} = \left({\sqcup_1}^{n_1}, \dots, {\sqcup_i}^{n_i}, \dots, {\sqcup_k}^{n_k}\right)$ and $\overrightarrow{\sqcup^n, \alpha} = \left({\sqcup_1}^{n_1}, \alpha_1, \dots, {\sqcup_i}^{n_i}, \alpha_i, \dots, {\sqcup_k}^{n_k}, \alpha_k\right)$, and let $h_{\overrightarrow{\sqcup^n,\alpha}}$ be the homomorphism from $\left(\Sigma_0 \cup \Gamma\right)^*$ to $\left(\Sigma_0 \cup \Psi\right)^*$ such that $h_{\overrightarrow{\sqcup^n,\alpha}}(\alpha_i) = {\sqcup_i}^{n_i}$ for each $\alpha_i \in \Gamma$ and $h_{\overrightarrow{\sqcup^n,\alpha}}(a) = a$ for each $a \in \Sigma_0$.

Let $\vec{v} = \left(v_1, v_2, v_3, \dots, v_k\right)$ such that v_i is a word over Δ_i for each $1 \leq i \leq k$, let $\overrightarrow{v, \alpha} = \left(v_1, \alpha_1, \dots, v_i, \alpha_i, \dots, v_k, \alpha_k\right)$, let $h_{\overrightarrow{v,\alpha}}$ be the homomorphism from $\left(\Sigma_0 \cup \Gamma\right)^*$ to $\left(\Sigma_0 \cup \Delta\right)^*$ such that $h_{\overrightarrow{v,\alpha}}(\alpha_i) = v_i$ for each $\alpha_i \in \Gamma$ and $h_{\overrightarrow{v,\alpha}}(a) = a$ for each $a \in \Sigma_0$, and

let $\overrightarrow{v, \sqcup^n} = \left(v_1, {\sqcup_1}^{n_1}, \ldots, v_i, {\sqcup_i}^{n_i}, \ldots, v_k, {\sqcup_k}^{n_k}\right)$. Denote by $\mathcal{T}_{\overrightarrow{v,\sqcup^n}}(L)$ the language $\mathcal{T}_{\overrightarrow{v,\sqcup^n}}(L) = \left\{ h_{\overrightarrow{v,\alpha}}(w) \middle| w \in \left(\Sigma_0 \cup \Gamma\right)^*, h_{\overrightarrow{\sqcup^n,\alpha}}(w) \in L \right\}$.

Let $\overrightarrow{\Lambda} = \left(\Lambda_1, \Lambda_2, \Lambda_3, \ldots, \Lambda_k\right)$ such that Λ_i is a language over Δ_i for each $1 \leq i \leq k$, let $\overrightarrow{\Lambda, \alpha} = \left(\Lambda_1, \alpha_1, \ldots, \Lambda_i, \alpha_i, \ldots, \Lambda_k, \alpha_k\right)$, let $\tau_{\overrightarrow{\Lambda,\alpha}}$ be the substitution from $\left(\Sigma_0 \cup \Gamma\right)^*$ to $\mathcal{P}\left(\left(\Sigma_0 \cup \Delta\right)^*\right)$ such that $\tau_{\overrightarrow{\Lambda,\alpha}}\left(\alpha_i\right) = \Lambda_i$ for each $\alpha_i \in \Gamma$ and $\tau_{\overrightarrow{\Lambda,\alpha}}(a) = \{a\}$ for each $a \in \Sigma_0$, and let $\overrightarrow{\Lambda, \sqcup^n} = \left(\Lambda_1, {\sqcup_1}^{n_1}, \ldots, \Lambda_i, {\sqcup_i}^{n_i}, \ldots, \Lambda_k, {\sqcup_k}^{n_k}\right)$. Denote by $\mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L)$ the language $\mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L) = \bigcup\left(\left\{ \tau_{\overrightarrow{\Lambda,\alpha}}(w) \middle| w \in \left(\Sigma_0 \cup \Gamma\right)^*, h_{\overrightarrow{\sqcup^n,\alpha}}(w) \in L \right\}\right)$.

The transformation $\mathcal{T}_{\overrightarrow{v,\sqcup^n}}(L)$ of L for $k = \left|\overrightarrow{v, \sqcup^n}\right|$ can be obtained in a series of k controlled shuffles each of them used to transform an interim language L_{i-1} to another interim language $L_i = \mathcal{T}_{v_i,{\sqcup_i}^{n_1}}\left(L_{i-1}\right)$ starting with $L_0 = L$ and ending with $L_k = \mathcal{T}_{\overrightarrow{v,\sqcup^n}}(L)$. Recall that each of the controlled shuffles needs the respective symbol $\sqcup_i$ to be absent from v_i $(1 \leq i \leq k)$ to work as intended, in other words, the controlled shuffles in general need $\Psi \cap \Delta = \varnothing$ which is added to the assumptions in what follows. Similarly, the transformation $\mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L)$ of L for $k = \left|\overrightarrow{\Lambda, \sqcup^n}\right|$ can be obtained by k consecutive controlled shuffles transforming L_{i-1} to $L_i = \mathcal{T}_{\Lambda_i,{\sqcup_i}^{n_1}}\left(L_{i-1}\right)$ with $L_0 = L$ and $L_k = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L)$. Again, the assumption $\Sigma \cap \Delta = \varnothing$ is required in general.

Notation 3.10. Let L be a language over $\Sigma_0 \cup \Psi$, let $k \in \mathbb{N}$, let Δ be an alphabet, let Ψ be an alphabet such that $|\Psi| = k$ and $\Psi = \left\{\sqcup_i \middle| 1 \leq i \leq k\right\}$, and $\Sigma_0 \cap \Psi = \varnothing$, let $\vec{\sqcup} = \left(\sqcup_1, \sqcup_2, \sqcup_3, \ldots, \sqcup_k\right)$, let Δ be an alphabet such that $\Psi \cap \Delta = \varnothing$, and let $\vec{n} = \left(n_1, n_2, n_3, \ldots, n_k\right)$ such that $n_i \in \mathbb{N} - \{0\}$ for each $1 \leq i \leq k$.

Let $\vec{v} = \left(v_1, v_2, v_3, \ldots, v_k\right)$ such that v_i is a word over Δ for each $1 \leq i \leq k$ and let $L_0 = L$ and $L_i = \mathcal{T}_{v_i,{\sqcup_i}^{n_i}}\left(L_{i-1}\right)$ for each $1 \leq 1 \leq k$. Denote by $L_{\mathcal{T}\overrightarrow{v,\sqcup^n}}$ the language $L_{\mathcal{T}\overrightarrow{v,\sqcup^n}} = L_k$.

Let $\overrightarrow{\Lambda} = \left(\Lambda_1, \Lambda_2, \Lambda_3, \ldots, \Lambda_k\right)$ such that Λ_i is a language over Δ for each $1 \leq i \leq k$ and let $L_0 = L$ and $L_i = \mathcal{T}_{\Lambda_i,{\sqcup_i}^{n_i}}\left(L_{i-1}\right)$ for each $1 \leq 1 \leq k$. Denote by $L_{\mathcal{T}\overrightarrow{\Lambda,\sqcup^n}}$ the language $L_{\mathcal{T}\overrightarrow{\Lambda,\sqcup^n}} = L_k$.

Lemma 3.11. Let L be a language over $\Sigma_0 \cup \Psi$, let $k \in \mathbb{N}$, let Δ be an alphabet, let Ψ be an alphabet such that $|\Psi| = k$ and $\Psi = \left\{\sqcup_i \middle| 1 \leq i \leq k\right\}$, and $\Sigma_0 \cap \Psi = \varnothing$, let $\vec{\sqcup} = \left(\sqcup_1, \sqcup_2, \sqcup_3, \ldots, \sqcup_k\right)$, let Δ be an alphabet such that $\Psi \cap \Delta = \varnothing$, and let $\vec{n} = \left(n_1, n_2, n_3, \ldots, n_k\right)$ such that $n_i \in \mathbb{N} - \{0\}$ for each $1 \leq i \leq k$.

Let $\vec{v} = \left(v_1, v_2, v_3, \ldots, v_k\right)$ such that v_i is a word over Δ for each $1 \leq i \leq k$. Then $\mathcal{T}_{\overrightarrow{v,\sqcup^n}}(L) = L_{\mathcal{T}\overrightarrow{v,\sqcup^n}}$.

Let $\overrightarrow{\Lambda} = \left(\Lambda_1, \Lambda_2, \Lambda_3, \ldots, \Lambda_k\right)$ such that Λ_i is a language over Δ for each $1 \leq i \leq k$. Then $\mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L) = L_{\mathcal{T}\overrightarrow{\Lambda,\sqcup^n}}$.

Proof. Take the assumptions from Lemma 3.11, Notation 3.10, and Notation 3.9.

Again, first $L_{\mathcal{T}\overrightarrow{\Lambda,\sqcup^n}} = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L)$ is proven then $L_{\mathcal{T}\overrightarrow{v,\sqcup^n}} = \mathcal{T}_{\overrightarrow{v,\sqcup^n}}(L)$ follows as a corollary.

The statement $L_{\mathcal{T}\overrightarrow{\Lambda,\sqcup^n}} = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L)$ is proven as follows.

In correspondence with the interim languages, interim alphabets, i-tuples, and interim mappings are defined. The defined alphabets and i-tuples are $\Psi_i = \left\{\sqcup_1, \sqcup_2, \sqcup_3, \ldots, \sqcup_i\right\}$ for each $0 \leq i \leq k$, $\vec{\sqcup}_i = \left(\sqcup_1, \sqcup_2, \sqcup_3, \ldots, \sqcup_i\right)$ for each $0 \leq i \leq k$, $\Gamma_i = \left\{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_i\right\}$ for each $0 \leq i \leq k$, $\vec{\alpha}_i = \left(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_i\right)$ for each $0 \leq i \leq k$, $\vec{n}_i = \left(n_1, n_2, n_3, \ldots, n_i\right)$ for each $0 \leq i \leq k$, $\Delta_{\cup i} = \Delta_1 \cup \Delta_2 \cup \Delta_3 \cup \cdots \cup \Delta_i$ for each $0 \leq i \leq k$, and $\overrightarrow{\Lambda}_i = \left(\Lambda_1, \Lambda_2, \Lambda_3, \ldots, \Lambda_i\right)$ for each $0 \leq i \leq k$. The defined mappings are the homomorphisms $h_{\overrightarrow{\sqcup^n,\alpha_i}}$ from $\left(\Sigma_0 \cup \Gamma_i\right)^*$ to $\left(\Sigma_0 \cup \Psi_i\right)^*$ such that $h_{\overrightarrow{\sqcup^n,\alpha_i}}(\alpha) = h_{\overrightarrow{\sqcup^n,\alpha}}(\alpha)$ for each $\alpha \in \Gamma_i$ and $h_{\overrightarrow{\sqcup^n,\alpha_i}}(a) = a$ for each $a \in \Sigma_0$ for each $0 \leq i \leq k$ and the substitutions $\tau_{\overrightarrow{\Lambda,\alpha_i}}$ from $\left(\Sigma_0 \cup \Gamma_i\right)^*$ to $\mathcal{P}\left(\left(\Sigma_0 \cup \Delta_{\cup i}\right)^*\right)$ such that $\tau_{\overrightarrow{\Lambda,\alpha_i}}(\alpha) = \tau_{\overrightarrow{\Lambda,\alpha}}(\alpha)$ for each $\alpha \in \Gamma_i$ and $\tau_{\overrightarrow{\Lambda,\alpha_i}}(a) = \{a\}$ for each $a \in \Sigma_0$ for each $0 \leq i \leq k$. Another mappings are the homomorphisms $h_{{\sqcup_i}^{n_i},\alpha_i}$ from $\left(\left(\Sigma_0 \cup \Delta_{\cup i-1}\right) \cup \left\{\alpha_i\right\}\right)^*$ to $\left(\left(\Sigma_0 \cup \Delta_{\cup i-1}\right) \cup \left\{\sqcup_i\right\}\right)^*$ such that $h_{{\sqcup_i}^{n_i},\alpha_i}\left(\alpha_i\right) = {\sqcup_i}^{n_i}$ and $h_{{\sqcup_i}^{n_i},\alpha_i}(a) = a$ for each $a \in \Sigma_0 \cup \Delta_{\cup i-1}$ for each $1 \leq i \leq k$ and the substitutions $\tau_{\Lambda_i,\alpha_i}$ from $\left(\left(\Sigma_0 \cup \Delta_{\cup i-1}\right) \cup \left\{\alpha_i\right\}\right)^*$ to $\mathcal{P}\left(\left(\Sigma_0 \cup \Delta_{\cup i}\right)^*\right)$ such that $\tau_{\Lambda_i,\alpha_i}\left(\alpha_i\right) = \Lambda_i$ and $\tau_{\Lambda_i,\alpha_i}(a) = \{a\}$ for each $a \in \Sigma_0 \cup \Delta_{\cup i-1}$ for each $1 \leq i \leq k$.

The main statement that $L_{\mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}} = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L)$ follows from an auxiliary statement that $L_i = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}_i}(L)$ for each $0 \leq i \leq k$.

The auxiliary statement that $L_i = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}_i}(L)$ for each $0 \leq i \leq k$ is proven by induction on i.

The induction basis $L_0 = L = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}_0}(L)$ holds.

By the induction hypothesis, it holds that $L_{i-1} = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}_{i-1}}(L)$.

The induction step is proven in two parts.

Let w be from L_i. There exists a word w_{α_i} over $\left(\Sigma_0 \cup \Delta_{\cup i-1}\right) \cup \left\{\alpha_i\right\}$ such that $h_{\sqcup_i{}^{n_i},\alpha_i}\left(w_{\alpha_i}\right)$ is from L_{i-1} and w is from $\tau_{\Lambda_i,\alpha_i}\left(w_{\alpha_i}\right)$. Since w_{α_i} is over $\left(\Sigma_0 \cup \Delta_{\cup i-1}\right) \cup \left\{\alpha_i\right\}$ there exist uniquely determined $k' \in \mathbb{N}$ and words $w_{2i'}$ over $\Sigma_0 \cup \Delta_{\cup i-1}$ for each $0 \leq i' \leq k'$ such that $w_{\alpha_i} = w_0\alpha_i w_2\alpha_i w_4\alpha_i \ldots w_{2k'}$. Since w is from $\tau_{\Lambda_i,\alpha_i}\left(w_{\alpha_i}\right)$, $w = w_0w_1w_2 \ldots w_{2k'}$ where $w_{2i'-1}$ is from $\tau_{\Lambda_i,\alpha_i}\left(\alpha_i\right) = \Lambda_i = \tau_{\overrightarrow{\Lambda,\alpha_i}}\left(\alpha_i\right)$ for each $1 \leq i' \leq k'$. Since $h_{\sqcup_i{}^{n_i},\alpha_i}\left(w_{\alpha_i}\right) = w_0\sqcup_i{}^{n_i}w_2\sqcup_i{}^{n_i}w_4\sqcup_i{}^{n_i} \ldots w_{2k'}$ is from $L_{i-1} = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}_{i-1}}(L)$ there exists a word $w_{\vec{\alpha}_{i-1}}$ over $\Sigma_0 \cup \Gamma_{i-1}$ such that $h_{\overrightarrow{\sqcup^n,\alpha_{i-1}}}\left(w_{\vec{\alpha}_{i-1}}\right)$ is from L and $h_{\sqcup_i{}^{n_i},\alpha_i}\left(w_{\alpha_i}\right)$ is from $\tau_{\overrightarrow{\Lambda,\alpha_{i-1}}}\left(w_{\vec{\alpha}_{i-1}}\right)$. Since $h_{\sqcup_i{}^{n_i},\alpha_i}\left(w_{\alpha_i}\right)$ is from $\tau_{\overrightarrow{\Lambda,\alpha_{i-1}}}\left(w_{\vec{\alpha}_{i-1}}\right)$, it holds that $w_{\vec{\alpha}_{i-1}} = u_0\sqcup_i{}^{n_i}u_1\sqcup_i{}^{n_i}u_2\sqcup_i{}^{n_i} \ldots u_{k'}$ such that $w_{2i'}$ is from $\tau_{\overrightarrow{\Lambda,\alpha_{i-1}}}\left(u_{i'}\right) = \tau_{\overrightarrow{\Lambda,\alpha_i}}\left(u_{i'}\right)$ for each $0 \leq i' \leq k'$. Let $w_{\vec{\alpha}_i} = u_0\alpha_i u_1\alpha_i u_2\alpha_i \ldots u_{k'}$. It holds that w is from $\tau_{\overrightarrow{\Lambda,\alpha_i}}\left(w_{\vec{\alpha}_i}\right)$. It holds that $w_{\vec{\alpha}_i}$ is over $\Sigma_0 \cup \Gamma_i$ and $h_{\overrightarrow{\sqcup^n,\alpha_i}}\left(w_{\vec{\alpha}_i}\right)$ is from L therefore $\tau_{\overrightarrow{\Lambda,\alpha_i}}\left(w_{\vec{\alpha}_i}\right)$ is a subset of $\mathcal{T}_{\overrightarrow{\upsilon,\sqcup^n}_i}(L)$. Since w is from $\tau_{\overrightarrow{\Lambda,\alpha_i}}\left(w_{\vec{\alpha}_i}\right)$ and $\tau_{\overrightarrow{\Lambda,\alpha_i}}\left(w_{\vec{\alpha}_i}\right)$ is a subset of $\mathcal{T}_{\overrightarrow{\upsilon,\sqcup^n}_i}(L)$, it follows that w is from $\mathcal{T}_{\overrightarrow{\upsilon,\sqcup^n}_i}(L)$.

For the converse, let w be from $\mathcal{T}_{\overrightarrow{\upsilon,\sqcup^n}_i}(L)$. There exists a word $w_{\vec{\alpha}_i}$ over $\Sigma_0 \cup \Gamma_i$ such that $h_{\overrightarrow{\sqcup^n,\alpha_i}}\left(w_{\vec{\alpha}_i}\right)$ is from L and w is from $\tau_{\overrightarrow{\Lambda,\alpha_i}}\left(w_{\vec{\alpha}_i}\right)$. Since $w_{\vec{\alpha}_i}$ is over $\Sigma_0 \cup \Gamma_i$ there exist uniquely determined $k' \in \mathbb{N}$ and words $u_{i'}$ over $\Sigma_0 \cup \Gamma_{i-1}$ for each $0 \leq i' \leq k'$ such that $w_{\vec{\alpha}_i} = u_0\alpha_i u_1\alpha_i u_2\alpha_i \ldots u_{k'}$. Since w is from $\tau_{\overrightarrow{\Lambda,\alpha_i}}\left(w_{\vec{\alpha}_i}\right)$, $w = w_0w_1w_2 \ldots w_{2k'}$ such that $w_{2i'}$ is from $\tau_{\overrightarrow{\Lambda,\alpha_i}}\left(u_{i'}\right) = \tau_{\overrightarrow{\Lambda,\alpha_{i-1}}}\left(u_{i'}\right)$ for each $0 \leq i' \leq k'$ and $w_{2i'-1}$ is from $\tau_{\overrightarrow{\Lambda,\alpha_i}}\left(\alpha_i\right) = \Lambda_i = \tau_{\Lambda_i,\alpha_i}\left(\alpha_i\right)$ for each $1 \leq i' \leq k'$. Let $w_{\vec{\alpha}_{i-1}} = u_0\sqcup_i{}^{n_i}u_1\sqcup_i{}^{n_i}u_2\sqcup_i{}^{n_i} \ldots u_{k'}$. It holds that $w_{\vec{\alpha}_{i-1}}$ is over $\Sigma_0 \cup \Gamma_{i-1}$ and $h_{\overrightarrow{\sqcup^n,\alpha_{i-1}}}\left(w_{\vec{\alpha}_{i-1}}\right) = h_{\overrightarrow{\sqcup^n,\alpha_i}}\left(w_{\vec{\alpha}_i}\right)$ is from L therefore $\tau_{\overrightarrow{\Lambda,\alpha_{i-1}}}\left(w_{\vec{\alpha}_{i-1}}\right)$ is a subset of $\mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}_{i-1}}(L) = L_{i-1}$. Let $w_{\alpha_i} = w_0\alpha_i w_2\alpha_i w_4\alpha_i \ldots w_{2k'}$. It holds that w_{α_i} is over $\left(\Sigma_0 \cup \Delta_{\cup i-1}\right) \cup \left\{\alpha_i\right\}$ and $h_{\sqcup_i{}^{n_i},\alpha_i}\left(w_{\alpha_i}\right) = w_0\sqcup_i{}^{n_i}w_2\sqcup_i{}^{n_i}w_4\sqcup_i{}^{n_i} \ldots w_{2k'}$ is from $\tau_{\overrightarrow{\Lambda,\alpha_{i-1}}}\left(w_{\vec{\alpha}_{i-1}}\right)$ thus from L_{i-1} therefore $\tau_{\Lambda_i,\alpha_i}\left(w_{\alpha_i}\right)$ is a subset of L_i. Since w is from $\tau_{\Lambda_i,\alpha_i}\left(w_{\alpha_i}\right)$ and $\tau_{\Lambda_i,\alpha_i}\left(w_{\alpha_i}\right)$ is a subset of L_i, it follows that w is from L_i.

Thus $L_i = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}_i}(L)$ for each $0 \leq i \leq k$.

It follows from the auxiliary statement that $L_k = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}_k}(L)$.

Since $\vec{n}_k = \vec{n}$ and $\overrightarrow{\Lambda}_k = \overrightarrow{\Lambda}$, the main statement follows, namely $L_{\mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}} = L_k = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}_k}(L) = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L)$.

The statement $L_{\mathcal{T}_{\overrightarrow{\upsilon,\sqcup^n}}} = \mathcal{T}_{\overrightarrow{\upsilon,\sqcup^n}}(L)$ can be proven as a corollary of the statement $L_{\mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}} = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L)$.

Let $\vec{\upsilon} = \left(\upsilon_1, \upsilon_2, \upsilon_3, \ldots, \upsilon_k\right)$ such that υ_i is a word over Δ for each $1 \leq i \leq k$. It holds that $L_{\mathcal{T}_{\overrightarrow{\upsilon,\sqcup^n}}} = L_{\mathcal{T}_{\overrightarrow{\{\upsilon\},\sqcup^n}}} = \mathcal{T}_{\overrightarrow{\{\upsilon\},\sqcup^n}}(L) = \mathcal{T}_{\overrightarrow{\upsilon,\sqcup^n}}(L)$.

For a more detailed argument, it suffices to replace the languages Λ_i by the words υ_i for each $1 \leq i \leq k$ and the substitution $\tau_{\overrightarrow{\Lambda,\alpha}}$ by the homomorphism $h_{\overrightarrow{\upsilon,\alpha}}$ in Notation 3.9 in all detail in the proof. □

In the simulations, an extremal case is used, namely the case where $\Sigma_0 = \varnothing$ thus L is a language over Ψ which corresponds to transformations $\mathcal{T}_{\overrightarrow{\upsilon,\sqcup^n}}(L)$ and $\mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L)$ of L such that the string $\sqcup_i{}^{n_i}$ is transformed to the word υ_i or the language Λ_i, respectively, for each symbol $\sqcup_i$ from the underlying alphabet Ψ. It follows from above that each from the languages $\mathcal{T}_{\overrightarrow{\upsilon,\sqcup^n}}(L)$ and $\mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L)$ can be obtained from L over Ψ

in a series of k controlled shuffles for $k = |\Psi|$. The discussed special cases can be used in the simulations of intersection, homomorphism, and substitution.

In this extremal case, the language L is entirely over Ψ therefore the underlying alphabet Ψ of L can be renamed to Σ_0. To obtain the transformations $\mathcal{T}_{\overrightarrow{v,\sqcup^n}}(L)$ or $\mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L)$ of L by the controlled shuffle, the alphabets (now) Σ_0 and Δ cannot have common elements. Since it is desired to keep the resulting languages $\mathcal{T}_{\overrightarrow{v,\sqcup^n}}(L)$ or $\mathcal{T}_{\overrightarrow{\Lambda,\sqcup^n}}(L)$ over Σ_0, the underlying alphabet of L can again be renamed from Σ_0 to ${\Sigma_0}'$ (leaving the name Σ_0 for the former alphabet Δ) hence the marked versions of the original languages are used in the simulations.

Let $\vec{n} = (2, 2, 2, \dots, 2)$ and $\vec{v} = \vec{\alpha}$. Then $h_{\overrightarrow{v,\alpha}} = \mathrm{id}$ and $h_{\overrightarrow{\sqcup^2,\alpha}}\left(\alpha_i\right) = \sqcup_i{}^2$ for each $1 \leq i \leq k_0$. The language $\mathcal{T}_{\overrightarrow{\alpha,\sqcup^2}}(L) = \left\{ w \middle| w \in {\Sigma_0}^*, h_{\overrightarrow{\sqcup^2,\alpha}}(w) \in L \right\}$. The language $\mathcal{T}_{\overrightarrow{\alpha,\sqcup^2}}(L)$ is the preimage of L under $h_{\overrightarrow{\sqcup^2,\alpha}}$ (words in L consisting of doubled symbols are carried over to $\mathcal{T}_{\overrightarrow{\alpha,\sqcup^2}}(L)$ with the double symbols replaced by corresponding single symbols while other words in L are discarded).

Let $\overrightarrow{\sqcup^n} = \vec{\alpha}$. Then $h_{\overrightarrow{\sqcup^n,\alpha}} = \mathrm{id}$, $\vec{n} = (1, 1, 1, \dots, 1)$, and $h_{\overrightarrow{v,\alpha}}\left(\alpha_i\right) = v_i$ for each $1 \leq i \leq k$. The language $\mathcal{T}_{\overrightarrow{v,\alpha}}(L) = \left\{ h_{\overrightarrow{v,\alpha}}(w) \middle| w \in {\Sigma_0}'^*, w \in L \right\}$. If v_i is the image of α_i under the same homomorphism h for each $1 \leq i \leq k$, the language $\mathcal{T}_{\overrightarrow{v,\alpha}}(L)$ is the image of L under h.

Let once again $\overrightarrow{\sqcup^n} = \vec{\alpha}$. Then $h_{\overrightarrow{\sqcup^n,\alpha}} = \mathrm{id}$, $\vec{n} = (1, 1, 1, \dots, 1)$, and $\tau_{\overrightarrow{\Lambda,\alpha}}\left(\alpha_i\right) = \Lambda_i$ for each $1 \leq i \leq k$. The language $\mathcal{T}_{\overrightarrow{\Lambda,\alpha}}(L) = \bigcup \left(\left\{ \tau_{\overrightarrow{\Lambda,\alpha}}(w) \middle| w \in {\Sigma_0}'^*, w \in L \right\} \right)$. If Λ_i is the image of α_i under the same substitution τ for each $1 \leq i \leq k$, the language $\mathcal{T}_{\overrightarrow{\Lambda,\alpha}}(L)$ is the image of L under τ.

The simulation of the intersection by the controlled shuffle is the most involved one. As already discussed in detail above, marked languages are used instead of the original languages. The simulation of the intersection is realized in two phases. In the first phase, an interlace of the languages is constructed consisting essentially from words of even length with the odd symbols being the symbols of a word from the first of the languages and the even symbols being the symbols of a word of the same length from the second of the languages. Observe, that a word is from the intersection if and only if there is a word in the interlace with each symbol of the original word doubled. Therefore, in the second phase, any word from the interlace of the languages which originates from two copies of the same word is transformed to the very word and any word from the interlace of the languages which originates from a pair of distinct words is discarded resulting in the intersection of the original languages. The second phase is done in a series of steps as explained in the lemma above.

As mentioned above, any word from the interlace $L_1 \sqcup\!\sqcap L_2$ of L_1 and L_2 is composed from words u in L_1 and v in L_2 of the same length alternating the symbols in u and v starting with a symbol in u and finishing with a symbol in v.

Notation 3.12. Let u and v be words over Σ_0 (resp. $\Sigma_0 \cup \{\sqcup\}$) of the same length k. Denote by $u \sqcup\!\sqcap v$ the word $u \sqcup\!\sqcap v = a_1 b_1 a_2 b_2 a_3 b_3 \dots a_k b_k$ such that a_i and b_i are symbols for each $1 \leq i \leq k$ and $u = a_1 a_2 a_3 \dots a_k$ and $v = b_1 b_2 b_3 \dots b_k$.

Let L_1 and L_2 be languages over Σ_0 (resp. $\Sigma_0 \cup \{\sqcup\}$). Denote by $L_1 \sqcup\!\sqcap L_2$ the language $L_1 \sqcup\!\sqcap L_2 = \left\{ u \sqcup\!\sqcap v | u \in L_1, v \in L_2, |u| = |v| \right\}$.

The following method shows how to construct $L_1 \sqcup\!\sqcap L_2$ for L_1 and L_2 over Σ_0 using the controlled shuffle. The languages L_1 and L_2 are first interlaced with $\{\sqcup\}^*$ from the right and from the left, respectively, resulting in $L_1 \sqcup\!\sqcap \{\sqcup\}^*$ and $\{\sqcup\}^* \sqcup\!\sqcap L_2$. These special cases of interlacing are realized by two simple homomorphisms. The general case resulting in the interlace of (arbitrary) languages is then realized by the controlled shuffle. In particular, the languages $L_1 \sqcup\!\sqcap \{\sqcup\}^*$ and $\{\sqcup\}^* \sqcup\!\sqcap L_2$ are shuffled into $L_1 \sqcup\!\sqcap L_2$.

Notation 3.13. Let L_1 and L_2 be languages over Σ_0 and let $h_{\mathrm{id}\,\sqcup}$ and $h_{\sqcup\,\mathrm{id}}$ be homomorphisms from ${\Sigma_0}^*$ to $\left(\Sigma_0 \cup \{\sqcup\}\right)^*$ such that $h_{\mathrm{id}\,\sqcup}(a) = a\sqcup$ for each $a \in \Sigma_0$ and $h_{\sqcup\,\mathrm{id}}(a) = \sqcup a$ for each $a \in \Sigma_0$. Denote by $L_{1\sqcup\!\sqcap 2}$ the language $L_{1\sqcup\!\sqcap 2} = h_{\mathrm{id}\,\sqcup}\left(L_1\right) \text{⧢} h_{\sqcup\,\mathrm{id}}\left(L_2\right)$.

Lemma 3.14. Let L_1 and L_2 be languages over Σ_0. Then $L_1 \sqcup\!\sqcap L_2 = L_{1\sqcup\!\sqcap 2}$.

Proof. Take the assumptions from Lemma 3.14, Notation 3.13, and Notation 3.12.

Let w be in $L_{1\sqcup\sqcap 2}$. There exist u in L_1 and v in L_2 such that $w = h_{\mathrm{id}\sqcup}(u) \sqcup\!\sqcup h_{\sqcup\,\mathrm{id}}(v)$. Thus the words $h_{\mathrm{id}\sqcup}(u)$ and $h_{\sqcup\,\mathrm{id}}(v)$ are compatible therefore $|u| = |v|$. Since $|u| = |v|$, $u\sqcup\!\sqcap v$ is in $L_1\sqcup\!\sqcap L_2$. Let $|u| = k = |v|$ and $u = a_1a_2a_3 \dots a_k$ and $v = b_1b_2b_3 \dots b_k$. It follows that $w = h_{\mathrm{id}\sqcup}(u) \sqcup\!\sqcup h_{\sqcup\,\mathrm{id}}(v) = a_1\sqcup a_2\sqcup a_3\sqcup \dots a_k\sqcup\!\sqcup\sqcup b_1\sqcup b_2\sqcup b_3 \dots \sqcup b_k = a_1b_1a_2b_2a_3b_3 \dots a_kb_k = u\sqcup\!\sqcap v$ is in $L_1\sqcup\!\sqcap L_2$.

Let w be in $L_1\sqcup\!\sqcap L_2$. There exist u in L_1 and v in L_2 such that $w = u\sqcup\!\sqcap v$. Thus $|u| = |v|$ therefore $h_{\mathrm{id}\sqcup}(u)$ and $h_{\sqcup\,\mathrm{id}}(v)$ are compatible. Since $h_{\mathrm{id}\sqcup}(u)$ and $h_{\sqcup\,\mathrm{id}}(v)$ are compatible, $h_{\mathrm{id}\sqcup}(u) \sqcup\!\sqcup h_{\sqcup\,\mathrm{id}}(v)$ is in $L_{1\sqcup\sqcap 2}$. Let $|u| = k = |v|$ and $u = a_1a_2a_3 \dots a_k$ and $v = b_1b_2b_3 \dots b_k$. It follows that $w = u\sqcup\!\sqcap v = a_1b_1a_2b_2a_3b_3 \dots a_kb_k = a_1\sqcup a_2\sqcup a_3\sqcup \dots a_k\sqcup\!\sqcup\sqcup b_1\sqcup b_2\sqcup b_3 \dots \sqcup b_k = h_{\mathrm{id}\sqcup}(u) \sqcup\!\sqcup h_{\sqcup\,\mathrm{id}}(v)$ is in $L_1\sqcup\!\sqcap L_2$. □

To obtain $L_1 \cap L_2$ from the interlace $L_1\sqcup\!\sqcap L_2$ of L_1 and L_2, any word $u\sqcup\!\sqcap v$ in $L_1\sqcup\!\sqcap L_2$ for $u \neq v$ has to be excluded and any word $u\sqcup\!\sqcap v$ in $L_1\sqcup\!\sqcap L_2$ for $u = w = v$ has to be transformed into the original word w. In other words, a word consisting of doubled symbols only has to be transformed info a word where each doubled symbol is halved and any other word has to be discarded. Both goals can be achieved by using the transformation of a language L into the language $\mathcal{T}_{\underrightarrow{v,\sqcup^n}}(L)$ as discussed before, namely the special case with $\vec{n} = (2, 2, 2, \dots, 2)$ and $\vec{v} = \vec{\alpha}$ transforming L into $\mathcal{T}_{\underrightarrow{\alpha,\sqcup^2}}(L) = \left\{ w \middle| w \in {\Sigma_0}^*, h_{\underrightarrow{\sqcup^2,\alpha}}(w) \in L \right\}$.

To have the possibility to obtain $\mathcal{T}_{\underrightarrow{\alpha,\sqcup^2}}(L)$ by the controlled shuffle, instead the language $L_1\sqcup\!\sqcap L_2$ its marked version $\left(L_1\sqcup\!\sqcap L_2\right)'$ is used. The marking can be done at any moment prior the use of the transformation. However, since the languages used in the simulation of the intersection $L_1 \cap L_2$ are desired to stay in the Chomsky family of L_1 and L_2, the languages L_1 and L_2 are marked by the (non erasing) homomorphism h_t at the very beginning of the whole process thus the marked languages ${L_1}'$ and ${L_2}'$ are used from the very beginning resulting then in the language $\left(L_1\sqcup\!\sqcap L_2\right)' = {L_1}'\sqcup\!\sqcap {L_2}'$.

Notation 3.15. Let L_1 and L_2 be languages over Σ_0 and let $\vec{\sqcup} = \left({a_1}', {a_2}', {a_3}', \dots, {a_{k_0}}'\right)$, $\vec{n} = (2, 2, 2, \dots, 2)$, and $\vec{v} = \left(a_1, a_2, a_3, \dots, a_{k_0}\right)$. Denote by $L_{1\cap 2}$ the language $L_{1\cap 2} = \mathcal{T}_{\underrightarrow{v,\sqcup^n}}\left({L_1}'\sqcup\!\sqcap {L_2}'\right)$.

Theorem 3.16. Let L_1 and L_2 be languages. Then $L_1 \cap L_2 = L_{1\cap 2}$.

Proof. Take the assumptions from Theorem 3.16 and Notation 3.15.

Let w be in $L_{1\cap 2}$. Therefore w is over Σ_0 and $w'\sqcup\!\sqcap w' = h_{\left({a_1}'^2, a_1, \dots, {a_i}'^2, a_i, \dots, {a_{k_0}}'^2, a_{k_0}\right)}(w)$ is in ${L_1}'\sqcup\!\sqcap {L_2}'$. If follows that w is in L_1 and in L_2 hence w is in $L_1 \cap L_2$.

Let w be in $L_1 \cap L_2$. Therefore w is in L_1 and in L_2. It follows that w is over Σ_0 and $h_{\left({a_1}'^2, a_1, \dots, {a_i}'^2, a_i, \dots, {a_{k_0}}'^2, a_{k_0}\right)}(w) = w'\sqcup\!\sqcap w'$ is in ${L_1}'\sqcup\!\sqcap {L_2}'$ thus w is in $L_{1\cap 2}$. □

The homomorphisms used in the simulation of the intersection by the controlled shuffle themselves can be simulated by the controlled shuffle as shown bellow. The simulation of the intersection enhanced by the simulation of the homomorphisms can be considered more complicated, since the number of the used controlled shuffles rises from $k_0 + 1$ to $3k_0 + 1$ with k_0 being the size of the underlying alphabet. On the other hand, the enhanced simulation can be considered simpler, since the input languages for the simulation are only the original languages L_1 and L_2 and regular languages without using homomorphisms at all.

The simulation of a homomorphism by the controlled shuffle also uses a marked language L_t instead of the original language L. Therefore the homomorphism used to mark the language must be dealt with as well. However, a homomorphism suitable for marking can be simulated by the controlled shuffle directly (without the use of another marked language) as the domain and codomain of a marking homomorphism already do not share a symbol. Therefore the simulation of a homomorphism marking the language is skipped (as it is the same as the general case without the use of a marked language) and the general case is presented instead.

Notation 3.17. Let h be a homomorphism from ${\Sigma_0}^*$ to ${\Sigma_0}^*$, let L be a language over Σ_0, and let $\vec{\sqcup} = \left({a_1}', {a_2}', {a_3}', \dots, {a_{k_0}}'\right)$, $\vec{n} = (1, 1, 1, \dots, 1)$, and $\vec{v} = \left(h\left(a_1\right), h\left(a_2\right), h\left(a_3\right), \dots, h\left(a_{k_0}\right)\right)$. Denote by L_h the language $L_h = \mathcal{T}_{\underrightarrow{v,\sqcup^n}}\left(L'\right)$.

Theorem 3.18. Let h be a homomorphism from ${\Sigma_0}^*$ to ${\Sigma_0}^*$ and let L be a language over Σ_0. Then $h(L) = L_h$.

Proof. Take the assumptions from Theorem 3.18 and Notation 3.17.
The language $L_h = \mathcal{T}_{\left(h(a_1),{a_1}',\ldots,h(a_i),{a_i}',\ldots,h\left(a_{k_0}\right),{a_{k_0}}'\right)}\left(L'\right) = \left\{ h_{\left(h(a_1),{a_1}',\ldots,h(a_i),{a_i}',\ldots,h\left(a_{k_0}\right),{a_{k_0}}'\right)}(w) \middle| w \in {\Sigma_0}'^*, w \in L' \right\} = \left\{ h(w) \middle| w \in {\Sigma_0}^*, w \in L \right\} = h(L).$ □

The simulation of a substitution by the controlled shuffle is extremely similar to the simulation of a homomorphism except for one point which is significantly different (even from all other presented simulations by the controlled shuffle). A marked language L' of the original language L is subsequently shuffled with languages which are not simple regular languages anymore. It can be shown that, in general, the languages belong to the same Chomsky family as the images of the symbols from the underlying alphabet under the simulated substitution.

Notation 3.19. Let τ be a substitution from ${\Sigma_0}^*$ to $\mathcal{P}\left({\Sigma_0}^*\right)$, let L be a language over Σ_0, and let $\vec{\sqcup} = \left({a_1}', {a_2}', {a_3}', \ldots, {a_{k_0}}'\right)$, $\vec{n} = (1, 1, 1, \ldots, 1)$, and $\overrightarrow{\Lambda} = \left(\tau\left(a_1\right), \tau\left(a_2\right), \tau\left(a_3\right), \ldots, \tau\left(a_{k_0}\right)\right)$. Denote by L_τ the language $L_\tau = \mathcal{T}_{\overrightarrow{\Lambda,\sqcup^{\vec{n}}}}\left(L'\right)$.

Theorem 3.20. Let τ be a substitution from ${\Sigma_0}^*$ to $\mathcal{P}\left({\Sigma_0}^*\right)$ and let L be a language over Σ_0. Then $\tau(L) = L_\tau$.

Proof. Take the assumptions from Theorem 3.20 and Notation 3.19.
The language $L_\tau = \mathcal{T}_{\left(\tau(a_1),{a_1}',\ldots,\tau(a_i),{a_i}',\ldots,\tau\left(a_{k_0}\right),{a_{k_0}}'\right)}\left(L'\right) = \bigcup\left(\left\{ \tau_{\left(\tau(a_1),{a_1}',\ldots,\tau(a_i),{a_i}',\ldots,\tau\left(a_{k_0}\right),{a_{k_0}}'\right)}(w) \middle| w \in {\Sigma_0}'^*, w \in L' \right\} \right) = \bigcup\left(\left\{ \tau(w) \middle| w \in {\Sigma_0}^*, w \in L \right\}\right) = \tau(L).$ □

4. Closure of Chomsky families under controlled shuffle

Motivated by the above observations, the question arises how strong the tool of controlled shuffle really is. One way to approach the question is to investigate the closure properties of the Chomsky families under controlled shuffle. In what follows, the results for the Chomsky families are presented.

Theorem 4.1. The family $\mathcal{R}$ of regular languages is closed under controlled shuffle.

Proof. Let L_1 and L_2 be regular languages. The family $\mathcal{R}$ is closed under a-transducer mapping therefore $L_{1M} = M\left(L_1\right)$ and $L_{2M} = M\left(L_2\right)$ are regular languages thus there exist finite automata $A_{1M} = \left(Q_1, \Sigma_1, \delta_1, q_{01}, F_1\right)$ accepting L_{1M} and $A_{2M} = \left(Q_2, \Sigma_2, \delta_2, q_{02}, F_2\right)$ accepting L_{2M}. Let $A_\perp = \left(Q, \Pi, \delta, q_0, F\right)$ be the finite automaton such that $Q = Q_1 \times Q_2$, $\begin{bmatrix} p_{11} \\ p_{12} \end{bmatrix} \in \delta\left(\begin{bmatrix} p_{01} \\ p_{02} \end{bmatrix}, \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}\right)$ if $p_{11} \in \delta\left(p_{01}, a_1\right)$ and $p_{12} \in \delta\left(p_{02}, a_2\right)$, $q_0 = \begin{bmatrix} q_{01} \\ q_{02} \end{bmatrix}$, and $F = F_1 \times F_2$. It can be shown that $A_\perp$ accepts the language $L_\perp = \left\{ w \middle| w \in \Pi^*, \pi_1(w) \in L_{1M}, \pi_2(w) \in L_{2M} \right\}$ of compatibility words therefore the language $L_\perp$ of compatibility words is regular. The family $\mathcal{R}$ is closed under homomorphism therefore $L = \mathfrak{h}\left(L_\perp\right)$ is regular. Since $L_1 ⧢ L_2 = L$, $L_1 ⧢ L_2$ is a regular language. □

Example 4.2. Let $L_1 = \{a^n \sqcup a^n \sqcup | n \in \mathbb{N} - \{0\}\}$ and let $L_2 = \{\sqcup b^n \sqcup b^n | n \in \mathbb{N} - \{0\}\}$. Clearly, L_1 and L_2 are context free while $L_1 ⧢ L_2 = \{a^m b^n a^m b^n | m, n \in \mathbb{N} - \{0\}\}$ is not.

Theorem 4.3. The family $\mathscr{L}_{CF}$ of context free languages is not closed under controlled shuffle.

Proof. It suffices to take the languages from the above example. □

Theorem 4.4. The family $\mathscr{L}_{ECS}$ of extended context sensitive languages is not closed under controlled shuffle.

Proof. Rather then giving an example of a pair of extended context sensitive languages L_1 and L_2 such that $L_1 ш L_2$ is not extended context sensitive, the statement of the theorem follows from more general observations. As mentioned above, the controlled shuffle can be used to simulate homomorphism, in particular, in can be used to simulate symbol erasing homomorphism. The image of a language L under the $э$ erasing homomorphism can be obtained as $L ш \{\varepsilon, ␣\}$. Since $\{\varepsilon, ␣\}$ is an extended context sensitive language for any symbol $␣$, the image $L ш \{\varepsilon, ␣\}$ of any extended context sensitive language L by the $э$ erasing homomorphism for any symbol $э$ can be obtained by the controlled shuffle of extended context sensitive languages. Since $\mathscr{L}_{ECS}$ is not closed under symbol erasing homomorphism, the statement of the theorem follows (it is also a direct corollary of the Theorem 4.7 dealing with the closure of $\mathscr{L}_{ECS}$ under controlled shuffle). □

Theorem 4.5. The family $\mathscr{L}_{RE}$ of recursively enumerable languages is closed under controlled shuffle.

Proof. The proof for $\mathscr{L}_{RE}$ is analogous to the proof for $\mathcal{R}$ except the construction of a multitape Turing machine $A_\perp$ accepting the language $L_\perp$ of compatibility words from the standard Turing machines A_{1M} and A_{2M} accepting the respective translations L_{1M} and L_{2M} of L_1 and L_2 by the a-transducer M. Besides the input tape containing the compatibility word candidate w, $A_\perp$ has the working tapes 1 and 2. The computation of $A_\perp$ consist of two phases. In the first phase, $A_\perp$ reads the word w from the input tape and copies the upper part $\pi_1(w)$ of w on the working tape 1 and the lower part $\pi_2(w)$ of w on the working tape 2. In the second phase, $A_\perp$ simulates the computation of A_{1M} on the working tape 1 and the computation of A_{2M} on the working tape 2 and accepts if both A_{1M} and A_{2M} accepted. □

Since the family $\mathscr{L}_{CF}$ of context free languages and the family $\mathscr{L}_{ECS}$ of extended context sensitive languages are not closed under the controlled shuffle, the question arises, what are the closures of these families under controlled shuffle.

Theorem 4.6. It holds that $\overline{\mathscr{L}_{CF}}^{ш} = \mathscr{L}_{RE}$.

Proof. The proof consist of two parts, first, $\overline{\mathscr{L}_{CF}}^{ш}$ is proven to be a subfamily of $\mathscr{L}_{RE}$, then, $overline\mathscr{L}_{CF}{}^{ш}$ is proven to contain $\mathscr{L}_{RE}$.

To prove that $\overline{\mathscr{L}_{CF}}^{ш}$ is a subfamily of $\mathscr{L}_{RE}$, it suffices to observe that $\mathscr{L}_{CF}$ is a subfamily of $\mathscr{L}_{RE}$ and $\mathscr{L}_{RE}$ are closed under controlled shuffle.

To prove that $\overline{\mathscr{L}_{CF}}^{ш}$ contains $\mathscr{L}_{RE}$, Lemma 4.7.1 in [3] essentially stating that a superfamily of $\mathscr{L}_{CF}$ containing $h\left(L_1 \cap L_2\right)$ for each L_1 and L_2 in $\mathscr{L}_{CF}$ and each homomorphism h is also a superfamily of $\mathscr{L}_{RE}$ is used.

Since $\overline{\mathscr{L}_{CF}}^{ш}$ is the closure of $\mathscr{L}_{CF}$, $\overline{\mathscr{L}_{CF}}^{ш}$ is a superfamily of $\mathscr{L}_{CF}$.

To prove that $\overline{\mathscr{L}_{CF}}^{ш}$ is a superfamily of $\mathscr{L}_{RE}$, it remains to show that $\overline{\mathscr{L}_{CF}}^{ш}$ contains $h\left(L_1 \cap L_2\right)$ for each L_1 and L_2 in $\mathscr{L}_{CF}$ and each homomorphism h. It is shown by arguing that $h\left(L_1 \cap L_2\right)$ can be obtained from context free languages (repeatedly) using only controlled shuffle.

Let L_1 and L_2 over Σ_0 be in $\mathscr{L}_{CF}$ and let h be a homomorphism from $\Sigma_0{}^*$ to $\Sigma_0{}^*$.

The languages used in the construction of $h\left(L_1 \cap L_2\right)$ by controlled shuffle are the languages $L_1 ⊓ \{␣\}^*$ and $\{␣\}^* ⊓ L_2$ in $\mathscr{L}_{CF}$ since they are obtained from L_1 and L_2, respectively, by homomorphism, the languages $\left\{{}^\top a_i{}' {}^\top {}^\top э {}^\top, a_i\right\}^*$ even in $\mathcal{R}$ for each $1 \leq i \leq k$, and the languages $\left\{{}^\top h\left(a_i\right)_{э,\varepsilon}{}^\top, a_i{}'\right\}^*$ also in $\mathcal{R}$ for each $1 \leq i \leq k$.

The language $h\left(L_1 \cap L_2\right)$ is constructed in two steps, first, $L_1{}' \cap L_2{}' = \left(L_1 \cap L_2\right)'$ is constructed from L_1 and L_2 and, then, $h\left(L_1 \cap L_2\right)$ is constructed form $\left(L_1 \cap L_2\right)'$.

The first construction proceeds as follows. The languages L_1 and L_2 can be used as the marked languages for the languages $L_1{}'$ and $L_2{}'$, respectively, to obtain $L_1{}' \cap L_2{}' = \mathcal{T}_{\overrightarrow{v,␣^n}}\left(L_1 ⊓ L_2\right)$ for $\vec{␣} = \left(a_1, a_2, a_3, \ldots, a_{k_0}\right)$, $\vec{n} = (2,2,2,\ldots,2)$, and $\vec{v} = \left(a_1{}', a_2{}', a_3{}', \ldots, a_{k_0}{}'\right)$. The language $\mathcal{T}_{\overrightarrow{v,␣^n}}\left(L_1 ⊓ L_2\right)$

can be obtained by using only controlled shuffle on the languages $L_1 ⨅ \{⨿\}^*$ and $\{⨿\}^* ⨅ L_2$ and $\left\{ {}^{\top} a_i' {}^{\top\top} ə^{\top}, a_i \right\}^*$ for each $1 \leq i \leq k$.

The second construction proceeds as follows. The language $\left(L_1 \cap L_2\right)'$ can be used as the marked language for $L_1 \cap L_2$ to obtain $h\left(L_1 \cap L_2\right) = \mathcal{T}_{\overrightarrow{v,⨿^{\vec{n}}}}\left(\left(L_1 \cap L_2\right)'\right)$ for $\vec{⨿} = \left(a_1', a_2', a_3', \ldots, a_{k_0}'\right)$, $\vec{n} = (1, 1, 1, \ldots, 1)$, and $\vec{v} = \left(h\left(a_1\right), h\left(a_2\right), h\left(a_3\right), \ldots, h\left(a_{k_0}\right)\right)$. The language $\mathcal{T}_{\overrightarrow{v,⨿^{\vec{n}}}}\left(\left(L_1 \cap L_2\right)'\right)$ can be obtained by using only controlled shuffle on the languages $\left(L_1 \cap L_2\right)'$ and $\left\{ {}^{\top} h\left(a_i\right)_{ə,\varepsilon} {}^{\top}, a_i' \right\}^*$ for each $1 \leq i \leq k$. □

The previous theorem also implies that $\overline{\mathscr{L}_{ECS}}^{⨊} = \mathscr{L}_{RE}$. However, it can be shown that one application of controlled shuffle suffices to obtain any language in $\mathscr{L}_{RE}$ from a suitable language in $\mathscr{L}_{ECS}$.

Theorem 4.7. It holds that $⨊\left(\mathscr{L}_{ECS}\right) = \mathscr{L}_{RE}$.

Proof. The statement is proven by proving that for every L over Σ_0 in $\mathscr{L}_{RE}$, there exists a language $L_{ə,\varepsilon}$ over $\Sigma \cup \{ə\}$ in $\mathscr{L}_{ECS}$ such that $L_{ə,\varepsilon} ⨊_{⨿^{\top}ə} \{\varepsilon, ⨿\} = L$ (simulating the $ə$ erasing homomorphism). Note that, since $L_{ə,\varepsilon}$ is even in $\mathscr{L}_{CS}$, a controlled shuffle with the singleton language $\{⨿\}$ is sufficient to obtain L.

A standard construction can be used to construct $L_{ə,\varepsilon}$ in $\mathscr{L}_{ECS}$, for instance the preimage under the $ə$ erasing homomorphism $h_{ə,\varepsilon}$ of L with removed ε. □

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] K. Chuda, Usefulness of Additional Information Formalized by Context Free Languages, PhD. Thesis, Comenius University Bratislava, 2026.

[2] K. Chuda, B. Rovan, Advice Usefulness in Framework of Context Free Grammars, (in preparation).

[3] S. Ginsburg, Algebraic and Automata-Theoretic Properties of Formal Languages, North-Holland, 1975.

[4] J. E. Hopcroft, R. Motwani, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Pearson, 2013.

[5] A. Mateescu, G. Rozenberg, A. Salomaa, Shuffle on Trajectories: Syntactic Constraints, Theoretical Computer Science 197 (1998) 1-56.

[6] R. Meersman, G. Rozenberg, Cooperating Grammar Systems, in: Mathematical Foundations of Computer Science 1978, Cham, 2006, pp. 364-373.

[7] G. Păun, G. Rozenberg, A. Salomaa, Grammars Based on the Shuffle Operation, Journal of Universal Computer Science 1 (1995) 67-82.

[8] G. Păun, L. Santean, Parallel Communicating Grammar Systems: The Regular Case, Analele Universității București, Matematică-Informatică 37 (1989) 55-63.

[9] B. Rovan, Š. Sádovský, On Usefulness of Information: Framework and NFA Case, in: H.-J. Böckenhauer, D. Komm, W. Unger (Eds.), Adventures Between Lower Bounds and Higher Altitudes, Cham, 2018, pp. 85-99.