The Pumping Lemma for Context-Free Languages is Undecidable

Hermann Gruber[1], Markus Holzer[2], and Christian Rauch[2]

[1] Planerio GmbH, Theresienhöhe 11A, 80538 München
h.gruber@planerio.de
[2] Institut für Informatik, Universität Giessen
Arndtstr. 2, 35392 Giessen, Germany
{holzer,christian.rauch}@informatik.uni-giessen.de

Abstract. Recently, the computational complexity of the Pumping-Problem, that is, for a given finite automaton A and a value p, deciding whether the language $L(A)$ satisfies a previously fixed regular pumping lemma w.r.t. the value p, was considered in [H. Gruber and M. Holzer and C. Rauch. The Pumping Lemma for Regular Languages is Hard. *CIAA 2023*, pp. 128-140.]. Here we generalize the Pumping-Problem by investigating Bar-Hillel's context-free pumping lemma instead. It turns out that for context-free languages, the Pumping-Problem for Bar-Hillel's pumping lemma is undecidable. When restricted to regular languages, the problem under consideration becomes decidable.

1 Introduction

Since the beginning of automata and formal language theory, researchers have studied pumping and iteration properties of formal languages to gain better insights into the computational complexity and expressive power of various types of language accepting or generating mechanisms. It is well known that not all formal language families obey pumping properties as, e.g., context-sensitive or Type-0 languages. Hence, satisfying a particular pumping property gives certain information about the structure of the language family, and is very often used to show that a particular language does not belong to the language family in question. For instance, Bar-Hillel's lemma [3] applied to the language $L = \{\, a^n b^n c^n \mid n \geq 0 \,\}$ shows that this language is *not* context free. In fact, the literature on pumping properties is far-reaching, with very different applications, see, e.g., [13], where language families are defined *via* pumping properties, [20] with a focus on pumping with the additional requirement that repeating a sub-word is allowed only if it is done a minimal number of times, or [15], which investigates regular pumping of Turing machine languages and learnability, just to mention a few.

We continue our research on the Pumping-Problem initiated in [7]. This is the problem of deciding for an automaton A (or a grammar G, respectively) and a value p, whether the language $L(A)$ (the set $L(G)$, respectively) satisfies a previously fixed pumping lemma w.r.t. the value p. For finite automata and Kozen's [16] and Jaffe's [14] pumping lemmata for regular languages, it has been shown that this problem is already intractable for DFAs, namely coNP-complete—this is the case for both pumping lemmata. This is quite remarkable, as it is a rare example of a finite automaton problem where the studied property becomes intractable for a single deterministic device. Jaffe's pumping property turns out to be more complex for NFAs, namely PSPACE-complete, while for Kozen's lemma it is shown to be coNP-hard and contained in Π_2^{P} for nondeterministic finite state devices. Furthermore, analysis of these problems has led to the conclusion that they are inapproximable unless the Exponential Time Hypothesis (ETH) fails. Since regular languages also satisfy context-free pumping lemmata, e.g., Bar-Hillel's pumping lemma [3], the natural question is how complicated it is to decide whether a regular or a context-free language satisfies a given context-free pumping lemma w.r.t. the pumping parameter involved? This is the starting point of the current paper.

Here we study the complexity of the Pumping-Problem for regular, linear context-free, and context-free languages w.r.t. Bar-Hillel's pumping lemma [3] and its variants for regular [18] and linear context-free language [12]. Before investigating these problems in detail, we present some basic properties of the minimal pumping constants w.r.t. these pumping lemmata. It is shown that in almost all cases the Pumping-Problem is undecidable, except when we consider regular languages represented by finite automata or right-linear grammars, where the problem becomes decidable. A more detailed analysis shows that the Pumping-Problem for context-free languages w.r.t. Bar Hillel's pumping lemma is complete for the Π_1^0-level of the arithmetical hierarchy. Observe that every language in Π_1^0 is the complement of a recursively enumerable language. A summary of the results obtained is given in Table 1.

Language family	Pumping w.r.t. value p		
	regular (Lem. 3)	linear context-free	context-free (Lem. 1)
REG	decidable		
LIN	undecidable (Π_1^0)		
CFL			

Table 1. Decidability status of the Pumping-Problem for different language families and pumping lemmata.

The paper is organized as follows: in the next section we introduce the necessary notations for context-free and regular languages and two special pumping lemmata for these language families. One of these pumping lemmata is the well known Bar-Hillel lemma [3] for context-free languages. The Pumping-Problem is then studied. First, the relation between the pumping constants induced by Bar-Hillel's lemma and its variants for regular and linear context-free languages is investigated. It is then shown that the PUMPING-PROBLEM for regular languages is decidable, for each of the pumping lemmata under consideration. We conclude with a summary and topics for further investigation.

2 Preliminaries

We assume the reader to be familiar with the basic notions on grammars and languages as contained in [12]. In particular, a *context-free grammar* (CFG) is a 4-tuple $G = (N, T, P, S)$, where N and T are disjoint alphabets of *nonterminals* and *terminals*, respectively, $S \in N$ is the *axiom*, and P is a finite set of *productions* of the form $A \to \alpha$, where $A \in N$ and $\alpha \in (N \cup T)^*$. As usual, the transitive closure of the derivation relation $\Rightarrow_G$ is written as $\Rightarrow_G^*$. If there is no danger of confusion, we simply write $\Rightarrow$ ($\Rightarrow^*$, respectively) instead of $\Rightarrow_G$ ($\Rightarrow_G^*$, respectively). The *language generated* by G is defined as

$$L(G) = \{\, w \in T^* \mid S \Rightarrow_G^* w \,\}.$$

We also consider the following restrictions of context-free grammars: (i) a context-free grammar is said to be *linear context-free* (LIN) if the productions are of the form $A \to \alpha$, where $A \in N$ and $\alpha \in T^*(N \cup \{\varepsilon\})T^*$—here ε refers to the *empty word*, and (ii) a context-free grammar is said to be *right-linear* or *regular* (REG) if the productions are of the form $A \to \alpha$, where $A \in N$ and $\alpha \in T^*(N \cup \{\varepsilon\})$.

The following pumping lemma for context-free languages can be found in [12, page 125, Lemma 6.1] and is a variant of the well-known Bar-Hillel pumping lemma [3, page 154, Theorem 4.1], see also [6, page 84, Lemma 3.1.1]. Observe that the original Bar-Hillel pumping lemma uses two pumping constants, one for the length of the word and the other for the sub-word that can be pumped.

Lemma 1. *Let L be a context-free language over Σ. Then, there is a constant p (depending on L) such that the following holds: If $z \in L$ and $|z| \geq p$, then there are words $u, v, w, x, y \in \Sigma^*$ such that $z = uvwxy$, $|vx| \geq 1$, $|vwx| \leq p$, and $uv^t wx^t y \in L$ for $t \geq 0$—it is then said that v and x can be (simultaneously)* pumped *in z.*

For a context-free language L, let $\texttt{mpcf}(L)$ denote the minimal number p satisfying the conditions of Lemma 1. Let us give a small example:

Example 2. Let $p \geq 2$. Consider the linear context-free grammar

$$G_p = (N, T, P_p, S)$$

with nonterminals $N = \{S\}$, terminals $T = \{a, b\}$, and the set of productions P_p containing the rules

$$S \to a^{p-1}Sb \mid \varepsilon.$$

It is easy to see that G_p generates the language $\{\, a^{n(p-1)}b^n \mid n \geq 0 \,\}$. By inspection, the sub-words a^{p-1} and b can be pumped in any word of $L(G_p)$, but any shorter word cannot since otherwise the number of a's and b's is no longer well correlated anymore. Hence, the minimal pumping constant w.r.t. Lemma 1 is equal to p. Thus, $\mathtt{mpcf}(T_p) = p$, where T_p refers to the language generated by the linear context-free grammar G_p. $\square$

The previous example shows that already for linear context-free grammars, with a single nonterminal, the minimal pumping constant can be arbitrarily large w.r.t. Lemma 1. The pumping lemma for linear-context free languages reads like the pumping lemma for context-free languages, but with one exception: instead of $|vwx| \leq p$, now the condition $|uvxy| \leq p$ is required—see [12, page 143, Exercise 6.11]. For a linear context-free language L let $\mathtt{mplin}(L)$ denote the minimal number p satisfying the conditions of the pumping lemma for linear context-free languages. Note that $\mathtt{mplin}(T_p) = p$ holds.

When considering regular languages, we usually use finite automata instead of right-linear grammars. A *nondeterministic finite automaton* (NFA) is a quintuple $A = (Q, \Sigma, \cdot, q_0, F)$, where Q is the finite set of *states*, Σ is the finite set of *input symbols*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states*, and the *transition function* $\cdot$ maps $Q \times \Sigma$ to 2^Q. Here 2^Q refers to the powerset of Q. The *language accepted* by the NFA A is defined as

$$L(A) = \{\, w \in \Sigma^* \mid (q_0 \cdot w) \cap F \neq \emptyset \,\},$$

where the transition function is recursively extended to a mapping $Q \times \Sigma^* \to 2^Q$ in the usual way. An NFA A is said to be *deterministic* (DFA) if $|q \cdot a| = 1$ for all $q \in Q$ and $a \in \Sigma$. In this case we simply write $q \cdot a = p$ instead of $q \cdot a = \{p\}$.

The syntactic monoid for a given language $L \subseteq \Sigma^*$, is defined by the *syntactic congruence* $\equiv_L$ over Σ^* where $v_1 \equiv_L v_2$ if and only if $uv_1w \in L \iff uv_2w \in L$ for every $u, w \in \Sigma^*$. Then the *syntactic monoid* is the quotient monoid $M(L) = \Sigma^*/\equiv_L$, where the concatenation of equivalence classes $[u]_{\equiv_L} \cdot [v]_{\equiv_L} = [uv]_{\equiv_L}$ serves as the monoid operation. The syntactic monoid of a regular language L is the smallest monoid recognizing the language under consideration (with respect to the division relation) and it is isomorphic to the transformation monoid of the minimal deterministic finite automaton accepting L. Here a language $L \subseteq \Sigma^*$ is *recognizable* if and only if there exists a finite monoid M, a morphism $\varphi_L : \Sigma^* \to M$, and a subset $N \subseteq M$ such that $L = \varphi_L^{-1}(N)$, which in turn is equivalent to the regularity (acceptance by a finite state machine) of L. For an n-state DFA (NFA, respectively) the size of the syntactic monoid is at most n^n (2^{n^2}, respectively).

The pumping lemma for regular languages, which can be found in [18, page 119, Lemma 8], [4, page 252, Folgerung 5.4.10], and [12, page 56, Lemma 6.1], reads as follows:

Lemma 3. *Let L be a regular language over Σ. Then there is a constant p (depending on L) such that the following holds: If $w \in L$ and $|w| \geq p$, then there are words $x \in \Sigma^*$, $y \in \Sigma^+$, and $z \in \Sigma^*$ such that $w = xyz$, $|xy| \leq p$, and $xy^tz \in L$ for $t \geq 0$. It is said that y can be* pumped *in w.*

Similarly to context-free languages, for a regular language L let $\texttt{mpl}(L)$ denote the smallest number p that satisfies the above statement. A more relaxed version of the previous lemma can be found in [16, page 70, Theorem 11.1], where the condition $|xy| \leq p$ is *not* required. We call this variant *Kozen's pumping lemma.* Obviously, Lemma 3 implies Kozen's pumping lemma. For a regular language L, the smallest value p that satisfies Kozen's pumping lemma is denoted $\texttt{mpc}(L)$. Then we have

$$\texttt{mpc}(L) \leq \texttt{mpl}(L) \leq \texttt{sc}(L),$$

where $\texttt{sc}(L)$ is the number of states of the minimal deterministic finite automaton (DFA) accepting L, as shown in [5]. For further properties of $\texttt{mpc}$ and $\texttt{mpl}$, see [5, 7, 10, 11].

3 The Language-Pumping-Problem

Recently in [7], the computational complexity of the pumping problem for regular languages w.r.t. Jaffe's [14] and Kozen's [16] pumping lemmata was investigated. It turned out that this problem is already intractable for DFAs and becomes PSPACE-complete for NFAs. Both pumping lemmata don't require any *upper* bound on the length of the pumped sub-word v or uv in $z = uvw$. The problem under consideration is defined as follows:

Language-Pumping-Problem or for short Pumping-Problem:
- Fixed: Particular pumping lemma, such as Lemma 1.
- Input: an accepting or generating device A for a formal language family such as, e.g., the regular languages, and a natural number p, i.e., an encoding $\langle A, 1^p\rangle$.
- Output: Yes, if and only if the statement from a previously fixed pumping lemma holds for the language $L(A)$ w.r.t. the value p.

Thus, the Language-Pumping-Problem is a natural host for different problem variants by considering different pumping lemmata and different formal language families. Here, we are particularly interested in the family of regular and context-free languages and some of their pumping lemmata as mentioned above.

Before investigating these problems in detail, we present some simple properties of the minimal pumping constants w.r.t. Lemma 1 and 3.

Theorem 4. *Let L be a regular language. Then $\texttt{mpcf}(L) \leq \texttt{mpl}(L)$ and moreover $\texttt{mplin}(L) \leq \texttt{mpl}(L)$.*

Proof. We only prove the first relation $\texttt{mpcf}(L) \leq \texttt{mpl}(L)$, which is immediate, because any pumpable decomposition of a word z w.r.t. Lemma 3 can be read as a pumpable decomposition w.r.t. Lemma 1. To this end consider a pumpable decomposition of $z = uvw$ with $|uv| \leq \texttt{mpl}(L)$ and $|v| \geq 1$ according to Lemma 3. Then define $u' = u$, $v' = v$, $w' = \varepsilon$, $x' = \varepsilon$, and $y' = w$. Obviously $z = u'v'w'x'y'$ and is a valid pumpable decomposition w.r.t. Lemma 1. This proves the stated claim. □

Recall, that from [5] is known that $\texttt{mpl}(L) \leq \texttt{sc}(L)$, for any regular language L. Here $\texttt{sc}(L)$ can be replaced by $\texttt{nsc}(L)$ as mentioned in [7]. For context-free languages we find the following situation, which follows from the proof of the pumping lemma given in [8, Chapter 6]:

Theorem 5. *Let L be generated by the context-free grammar $G = (N, T, P, S)$. Set $n := |N|$ and $m := \max\{\, 2, |\alpha| \mid A \to \alpha \in P \,\}$, then $\texttt{mpcf}(L) \leq m^{2n+3}$.* $\square$

If the given context-free grammar is in Chomsky normal form,[3] the proof in [12] yields the bound $\texttt{mpcf}(L) \leq 2^n$. Notice, however, that the conversion to normal form incurs a size blow-up in the worst case [17]. For linear context-free languages, the next theorem applies:

Theorem 6. *Let L be a* linear *context-free language generated by the linear context-free grammar $G = (N, T, P, S)$. Set $n := |N|$ and similarly as above define $m := \max\{\, |\alpha| \mid A \to \alpha \in P \,\}$, then $\texttt{mplin}(L) \leq (m-1) \cdot n + 2$.*

Proof. Consider a derivation $S \Rightarrow^* z$ of G generating a word $z \in L$, with $|z| \geq (m-1) \cdot n + 2$. (In case there is no such word z, then the pumping condition is trivially satisfied). Observe that the derivation must have at least $n + 1$ steps: the last derivation step can generate at most m terminal symbols, and every other derivation step can generate at most $m - 1$ terminal symbols each. Within n steps, the grammar G can thus generate only terminal words of length at most $(m-1)n + 1 < |z|$.

We find a decomposition $z = uvwxy$ that meets the needs as follows. Let A denote the first variable in the derivation that appears twice. Then the derivation can be written as

$$\underbrace{S \Rightarrow^* uAy \Rightarrow^* uvAxy}_{\leq n+1 \text{ steps}} \Rightarrow^* uvwxy.$$

Then $uvAxy$ is generated in at most $n + 1$ steps, each of which generates one variable and at most $m - 1$ terminal symbols, hence $|uvxy| \leq (m-1)(n+1)$. In case $|vx| = 0$, we can cut out the part $uAy \Rightarrow^* uvAxy$ from the derivation and recursively find a suitable decomposition for the shorter derivation. Finally, it is immediate that $A \Rightarrow^* v^i A x^i$ for all $i \geq 0$, so $z = uvwxy$ is a decomposition with the desired properties. This proves the stated upper bound. $\square$

The relation between $\texttt{mpcf}(L)$ and $\texttt{mplin}(L)$ for regular and linear context-free languages L is subject to further research.

3.1 Decidability of Context-Free Pumping for Regular Languages

The main result of this section is that the pumping problem for regular languages w.r.t. Lemma 1 is decidable. The statement reads as follows:

[3] A context-free grammar $G = (N, T, P, S)$ is in *Chomsky normal form* if every production is either of the form $A \to a$ or $A \to BC$ or $S \to \varepsilon$, for $A, B, C \in N$ and $a \in T$.

Theorem 7. *Given a finite automaton A and a natural number p, it is decidable whether for the language $L(A)$ the statement of Lemma 1 holds for the value p.*

Before we come to the proof of this result, let us take a closer look at the pumping lemma for regular languages, as stated in Lemma 3.

Theorem 8. *Let L be a regular language over Σ that is accepted by an n-state finite automaton and $p \leq n$. If for every $w \in L$ with $p \leq |w| \leq n + |M(L)|$, there are words $x \in \Sigma^*$, $y \in \Sigma^+$, and $z \in \Sigma^*$ such that $w = xyz$, $|xy| \leq p$, and $xy^tz \in L$ for $t \geq 0$, then Lemma 3 is satisfied w.r.t. the value p.*

Proof. It suffices to show that for every word w with $|w| > n + |M(L)|$ there are words $x \in \Sigma^*$, $y \in \Sigma^+$, and $z \in \Sigma^*$ such that $w = xyz$, $|xy| \leq p$, and $xy^tz \in L$ for $t \geq 0$. Since $|xy| \leq p$ and $p \leq n$, by assumption the pumping of y only appears in the prefix of length at most n of w. Hence we decompose w into $w = uv$ such that $|u| = n$ and replace v by the shortest word v' that is equivalent w.r.t. the syntactic monoid of L, i.e., $\varphi_L(v') = \varphi_L(v)$ for the syntactic morphism $\varphi_L : \Sigma^* \to M(L)$ and there is no shorter word than v' that satisfies the above equality. Then uv' is of length at most $n + |M(L)|$ and

$$w = uv \in L \iff uv' \in L.$$

By the precondition of the implication there are words $x \in \Sigma^*$, $y \in \Sigma^+$, and $z \in \Sigma^*$ such that $uv' = xyz$, $|xy| \leq p$, and $xy^tz \in L$ for $t \geq 0$. By construction xy entirely lies within u, while z may contain letters from the right end of u followed by the whole word v'. Thus, we can write $z = z_1z_2$ with $z_1 = (xy)^{-1}u$ and $z_2 = v'$. But then we can use this decomposition to construct a y-pumpable decomposition for the original word w we started from, by using the words x, y, and z_1v without changing the acceptance of the word

$$xy^tz = xy^tz_1z_2 \in L \iff xy^tz_1v \in L,$$

for $t \geq 0$, because $z_2 = v'$ and v belong to the same equivalence class w.r.t. the syntactic congruence $\equiv_L$ of the language L. $\square$

With the help of the previous theorem we can now show that the regular pumping-problem for regular languages is decidable.

Theorem 9. *Given a finite automaton A and a natural number p, it is decidable whether for the language $L(A)$ the statement of Lemma 3 holds for the value p.*

Proof. Let n be the number of states of the automaton A. A Turing machine M that decides the problem in question works as follows: first M constructs a list of all words of length at least p and at most $n + 2^{n^2}$, that belong to the language $L(A)$. Observe that $n^n \leq 2^{n^2}$ and therefore $|M(L)| \leq 2^{n^2}$ holds. If this list is empty, then the Turing machine halts and accepts, because in this case $L(A)$ is a finite language, whose longest word is shorter than p. Next assume that the constructed list is *not* empty. Then M cycles through all words z in the list and tries to find a valid decomposition $z = uvw$ according to Lemma 3 with value p such that v can be pumped in z. If the machine M does not find such a decomposition it halts and rejects. Otherwise it continues with the next word in the list. In case there is no next word in the list, that is, by cycling trough all words in the list M has found a p-length valid pumpable decomposition of each word, the Turing machine halts and accepts, since by Theorem 8 the problem instance requires a positive answer. $\square$

The proof of Theorem 9 relies heavily on Theorem 8, which does not obviously generalize to context-free pumping lemmata, because the simultaneous pumping of sub-words (context-free pumping) can occur at any position in a given word—and not necessarily only in a prefix of bounded length, as it does for the regular pumping lemma in question (Lemma 3). Thus, proving that the context-free pumping problem for regular languages is decidable requires a different proof strategy, which we develop in the next proof.

Proof (of Theorem 7). Recall, that A is a finite automaton with input alphabet Σ. Let p be natural number. We want to decide whether Lemma 1 holds for the regular language $L(A)$ with value p. First let us take a closer look at word decompositions used in Lemma 1 with value p when applied to a regular language. Let $L := L(A)$ and z be any word in L (not necessarily of length at least p). Any decomposition of z into $u, v, w, x, y \in \Sigma^*$ such that $z = uvwxy$, $|vx| \geq 1$, $|vwx| \leq p$, and $uv^t wx^t y \in L$ for $t \geq 0$, can be described by a five-tuple

$$(u', v, w, x, y'),$$

where u' (respectively y') is any representative for the Myhill-Nerode equivalence class $[u']_{\equiv_L}$ (respectively $[y']_{\equiv_L}$). For such a 5-tuple (induced by a word z and its decomposition), it is easy to see that any word z' with the property $z' \in [u']_{\equiv_L} \cdot vwx \cdot [y']_{\equiv_L}$, obeys a v-x-pumpable decomposition $z' = uvwxy$ with $u \in [u']_{\equiv_L}$ and $y \in [y']_{\equiv_L}$ satisfying $|vx| \geq 1$ and $|vwx| \leq p$ such that $uv^t wx^t y \in L$ for $t \geq 0$. Observe that the representatives of the equivalence classes in the first and last component can be chosen as a word of length at most 2^{n^2} each, because the syntactic monoid has at most 2^{n^2} elements. If this is the case, we call the corresponding five-tuple *valid*.

Let $D_{L,p}$ refer to the set of all valid five-tuples. It is easy to see that membership in $D_{L,p}$ is decidable. On input (u, v, w, x, y) a Turing machine first verifies the length requirements on these words, i.e., $|u|, |y| \leq 2^{n^2}$, $|vwx| \leq p$, and $|vx| \geq 1$. If the length requirements are fulfilled, then for the linear-context-free grammar $G = (\{S, A\}, \Sigma, P, S)$ with the production set $P = \{\, S \to uAy, A \to vAx \mid w \,\}$, which generates the language $\{\, uv^t wx^t y \mid t \geq 0 \,\}$, it is verified whether $L(G) \subseteq L$ holds. This can be done by checking $L(G) \cap \overline{L}$ for being empty. If this is the case, the Turing machine halts and accepts the input (u, v, w, x, y). Otherwise, the Turing machine halts and rejects. Here $\overline{L}$ refers to the complement of L, i.e., $\overline{L} := \Sigma^* \setminus L$. Since all the necessary checks on the linear context-free grammar G described above can be decided, the whole algorithm decides membership in $D_{L,p}$.

Next we construct an NFA for the language

$$P := \bigcup_{(u,v,w,x,y) \in D_{L,p}} [u]_{\equiv_L} \cdot vwx \cdot [y]_{\equiv_L}.$$

This can be easily done by cycling through all elements of $D_{L,p}$. It is easy to see that P describes all words in L that can be pumped according to Lemma 1 w.r.t. the value p. Finally we build an automaton for the language $L \setminus P$. Then we consider two cases:

1. $L \setminus P$ contains a word w of length at least p. Then the pumping lemma for context-free languages (Lemma 1) w.r.t. the value p does not hold for L. This is witnessed by w. Hence, the input A and p has to be rejected.
2. $L \setminus P$ does not contain any word of length at least p. Then the pumping lemma for context-free languages w.r.t. the value p holds for L. Therefore the input A and p is accepted.

Since all constructions rely on basic operations on regular languages that can be done by a Turing machine, the problem in question is decidable. This proves the stated claim. $\square$

With the idea used in the previous proof one can also show the following decidability result for linear context-free pumping on regular languages.

Theorem 10. *Given a finite automaton A and a natural number p, it is decidable whether for the language $L(A)$ the statement of the pumping lemma for linear context-free languages holds for the value p.* $\square$

3.2 Undecidability of Context-Free Pumping for Context-Free Languages

In contrast to the previous section, where it was shown that the PUMPING-PROBLEM is decidable if we use the pumping lemmata under consideration on regular languages, here we show that the PUMPING-PROBLEM becomes undecidable if we consider context-free languages. In fact, the undecidability already holds for linear-context-free languages. To this end, we exploit non-semi-decidable properties of Turing machines by encoding complex Turing machine computations into small grammars [9].

Basically, we consider *valid computations* of Turing machines. Roughly speaking, these are histories of accepting Turing machine computations. It suffices to consider deterministic Turing machines with one single tape and one single read-write head. Without loss of generality and for technical reasons, we assume that the Turing machine accepts by halting and cannot print blanks. A valid computation is a string built from a sequence of configurations passed through during an accepting computation. To be more precise, let Q be the state set of some Turing machine M, where q_0 is the initial state, T is the tape alphabet containing the blank symbol satisfying $T \cap Q = \emptyset$, and $\Sigma \subseteq T$ is the input alphabet.

Then a *configuration* of M can be written as a word of the form T^*QT^* such that $a_1a_2\cdots a_iqa_{i+1}\cdots a_n$ is used to express that M is in state q, scanning tape symbol a_{i+1}, and a_1, a_2 to a_n is the support of the tape inscription.

Let $\mathsf{VAL}(M)$ be the set of all words of the form

$$w_1\$w_2^R\$w_3\$w_4^R\$\cdots\$w_{2k-1}\$ \quad\text{or}\quad w_1\$w_2^R\$w_3\$w_4^R\$\cdots\$w_{2k}^R\$$$

where \$ is a new symbol not contained in $T\cup Q$, sub-words $w_i\in T^*QT^*$ are configurations of M, word w_1 is an *initial configuration* of the form $q_0\Sigma^*$, word w_{2k-1} (w_{2k}, respectively) is a *halting configuration*, i.e., accepting configuration, and w_{i+1} is the *successor configuration* of w_i. The set of all *invalid computations* $\mathsf{INVAL}(M)$ is the complement of $\mathsf{VAL}(M)$ w.r.t. the alphabet $T\cup Q\cup\{\$\}$. From [9] it is known, that a linear context-free grammar generating the language $\mathsf{INVAL}(M)$ can be effectively constructed from a description of M.

It is worth mentioning that $\mathtt{mpcf}(\mathsf{INVAL}(M))=1$ holds. This is quite surprising, but due to the fact, that whenever the word under consideration contains the encoding of a state, then this state can be pumped without changing the membership of the pumped word. If there is no state in the considered word, then we pump any single letter that, which the pumped string within the language $\mathsf{INVAL}(M)$. Nevertheless, we will use a particular encoding of $\mathsf{INVAL}(M)$ for our purpose to show that the PUMPING-PROBLEM becomes undecidable if one considers context-free languages and their pumping lemma.

Theorem 11. *Given a context-free grammar G and a natural number p, it is undecidable whether for the language $L(G)$ the statement of Lemma 1 for the value p holds. The statement remains valid if a linear context-free grammar and the pumping lemma for linear-context-free languages is considered instead.*

Proof. We only prove the statement for the context-free pumping case (on a linear context-free language). The proof for the linear context-free case is similar and is left to the interested reader.

The emptiness problem for Turing machines, which is undecidable, is reduced to the problem in question. Let M be a Turing machine. Recall, that $L(M)=\emptyset$ if and only if $\mathsf{INVAL}(M)=\Sigma^*$, for some alphabet Σ that depends on M. Without loss of generality we assume that Σ contains the letters a and b.

Let $p\geq 2$ and $T_p\subseteq\{a,b\}^*$ be a regular language satisfying $\mathtt{mpcf}(T_p)=p$. Consider the language

$$L_M=h(\Sigma^*)\cdot\#\cdot T_p\cup h(\mathsf{INVAL}(M))\cdot\#\cdot\Sigma^+\cup\Sigma^*\#\cup\#^*,$$

where # is a new symbol not contained in Σ and $h:\Sigma\to\Sigma^*$ is the homomorphism defined by $h(a)=a^2$, for $a\in\Sigma$. It is not hard to see that one can construct a linear context-free grammar for the language L_M—the details are omitted.

Next we show how to decide the emptiness problem for Turing machines using the linear-context free grammar G_M that generates L_M. First observe that whenever we have a word from the sub-languages $\Sigma^*\#$ or $\#^*$, respectively, then such a word can be pumped by using a single letter from Σ, or by using the symbol #, respectively. Thus for those words, the pumping constant can be chosen to be 1. It remains to consider the words from the remaining two sub-languages. To this end we consider two cases:

1. If $L(M) = \emptyset$, then $\mathsf{INVAL}(M) = \Sigma^*$. This implies that
$$L_M = h(\Sigma^*) \cdot \# \cdot \Sigma^+ \cup \Sigma^* \# \cup \#^*,$$
and any word $u\#w$ from the sub-language $h(\Sigma^*) \cdot \# \cdot \Sigma^+$ can be pumped by any single letter from w, even if the word w consists of only a single letter. By the above argumentation we conclude that the pumping constant for the whole language L_M can be chosen to be $p = 1$.
2. Otherwise, let $L(M) \neq \emptyset$. Regardless of the assumption, note that any non-empty word in $h(\mathsf{INVAL}(M)) \cdot \# \cdot \Sigma^+$ can be pumped by any single letter that appears after the #-symbol. Thus, for these words we can choose the pumping constant $p = 1$.
It remains to consider the words in $h(\Sigma^*) \cdot \# \cdot T_p$ that are not covered by the set $h(\mathsf{INVAL}(M)) \cdot \# \cdot \Sigma^+$. Since by assumption we have $L(M) \neq \emptyset$, there is at least one word $u = h(u')$ such that u' does not belong to $\mathsf{INVAL}(M)$. Next we consider the word $u\#w$, for a non-empty word $w \in T_p$ such that w is a witness for $\texttt{mpcf}(T_p) = p$, i.e., any context-free pumping within w requires the length of the simultaneously pumped words (both together) are of length p. The word $u\#w$ can be properly pumped as follows:
   (a) If the pumping appears entirely in the sub-word u, then the total length of the simultaneously pumped words is even, since otherwise the length constraint being of even length is not satisfied for the prefix up to the #-symbol. Hence the minimal pumping constant is at least 2.
   (b) The pumping appears entirely in the sub-word w, which is a member of T_p and a witness for $\texttt{mpcf}(T_p) = p$. Hence, the pumping constant for $u\#w$ must be chosen to be at least p. By assumption $p \geq 2$.
   (c) Finally, the pumping appears in both u and w—note that the #-symbol can not be part of any word for pumping. Here we exclude the minimal pumping constant 1, which implies that minimal pumping constant is at least 2. If the length of the simultaneously pumped word is 1, then either the pumped sub-word in u or w is empty. In both cases, pumping of a single letter is not possible, because either the length constraint being of even length is not satisfied for the prefix up to the #-symbol or the suffix doesn't belong to T_p anymore, since w was a witness for $\texttt{mpcf}(T_p) = p$, for $p \geq 2$. Thus, in this case the minimal pumping constant is at least 2.

   Summarizing, in all sub-cases the minimal pumping constant is at least 2.

Thus, the case analysis shows we can decide whether $L(M)$ is empty or not by checking whether for the linear context-free grammar G_M that generates the language L_M the Bar-Hillel pumping lemma is satisfied w.r.t. the value $p = 1$. If this is the case, then $L(M) = \emptyset$. Otherwise, the Turing machine M accepts at least one word and thus its language is non-empty. This proves the stated claim on the undecidability of the Bar-Hillel pumping lemma applied to context-free languages. □

We can reuse the proof for Theorem 11 to prove the following statement for regular pumping.

Theorem 12. *Given a context-free grammar G and a natural number $p \geq 3$, it is undecidable whether for the language $L(G)$ the statement of Lemma 3 for the value p holds. The statement remains valid if a linear context-free grammar is considered, or the constraint $|xy| \leq p$ is no longer a prerequisite.*

Proof. First we argue about the case where $|xy| \leq p$ is not required for a decomposition regarding Lemma 3. We observe that dropping the length constraint implies that we can pump a word $w \in L$ which fulfills $|w| \geq p$ if it can be decomposed into xyz for any words $x, y, z \in \Sigma$ such that $|y| \geq 1$ and $xy^tz \in L$ for $t \geq 0$. We refer to the minimal constant p fulfilling this statement for a language L by $\texttt{mpc}(L)$.

By inspecting the cases of the proof of Theorem 11 we obtain that in the case $L(M) = \emptyset$ we have $\texttt{mpc}(L) = 1$ and in the case $L(M) \neq \emptyset$ we get $\texttt{mpc}(L) \geq 2$. Therefore the statement of this theorem follows for the variant of Lemma 3. Unfortunately we obtain for the original version of the lemma that $\texttt{mpl}(L_M) = 2$ regardless of whether $L(M) \neq \emptyset$ or not. Therefore we use a variant of the language L_M instead by reversing the concatenations but not the individual languages. Let

$$L'_M = T_p \cdot \# \cdot h(\Sigma^*) \cup \Sigma^+ \cdot \# \cdot h(\mathsf{INVAL}(M)) \cup \# \cdot \Sigma^* \cup \#^*.$$

Then we consider two cases:

1. If $L(M) = \emptyset$, then $\mathsf{INVAL}(M) = \Sigma^*$ and we get that every word w in the language
$$L'_M = \Sigma^+ \cdot \# \cdot h(\Sigma^*) \cup \# \cdot \Sigma^* \cup \#^*,$$
can be pumped by its first letter except the words in $\# \cdot \Sigma^*$. All these words allow pumping by their second letter. Therefore $\texttt{mpl}(L'_M) = 2$ in this case.
2. For $L(M) \neq \emptyset$ there exists a word $w\#u \in T_p \cdot \# \cdot h(\Sigma^*)$ such that w is a witness for $\texttt{mpl}(T_p) = p$ and $u = h(u')$ with $u' \notin \mathsf{INVAL}(M)$. Without loss of generality we assume that $|w| \geq p - 1$ and $u \neq \lambda$. Pumping w.r.t. Lemma 3 inside the prefix of length $p-1$ of w results in a word $x\#u$ such that $x \notin T_p$. This implies that $x\#u \notin L_M^R$ since $u \notin h(\mathsf{INVAL}(M))$. Hence, we obtain $\texttt{mpl}(L_M^R) \geq p$ in the case that $L(M) \neq \emptyset$.

Therefore we obtain that for a linear context-free grammar it is undecidable whether the statement of Lemma 3 holds for a given value p. □

3.3 More on Context-Free Pumping for Context-Free Languages

As mentioned above, it is undecidable whether for a given context-free grammar G and a value p the Bar-Hillel's pumping w.r.t. the value p is satisfied.

So, we are interested to explore how hard the problem is. Is it semi-decidable or is it placed at a higher degree of unsolvability? To this end, we consider the *arithmetic hierarchy*, which is defined as follows—see, e.g., [19]:

$$\begin{aligned} \Sigma_1^0 &= \{\, L \mid L \text{ is recursively enumerable} \,\}, \\ \Sigma_{n+1}^0 &= \{\, L \mid L \text{ is recursive enumerable in some } A \in \Sigma_n^0 \,\}, \end{aligned}$$

for $n \geq 1$. Here a language L is said to be recursively enumerable in some B if there is a Turing machine with oracle B that semi-decides L. Here Π_n^0 is the complement of Σ_n^0, i.e., $\Pi_n^0 = \{\, L \mid \overline{L} \text{ is in } \Sigma_n^0 \,\}$. Observe that the intersection $\Sigma_1^0 \cap \Pi_1^0$ is the class of all recursive sets. Completeness and hardness are always meant with respect to many-one reducibilities.

A well-known Π_1^0-complete problem, which we will refer to, is the emptiness problem for Turing machines [19], denoted by EMPTY. Here EMPTY $= \{\, \langle M \rangle \mid L(M) = \emptyset \,\}$, where $\langle M \rangle$ is the index, or Gödel number, of M. As usual, $\overline{\text{EMPTY}}$ denotes the complement of EMPTY, which is Σ_1^0-complete. In fact, both problems are related to the Pumping-Problem in question. Recall the proof of Theorem 11. There we reduced the problem EMPTY to the Pumping-Problem for context-free languages w.r.t. Bar Hillel's lemma by

$\langle M \rangle \in$ EMPTY if and only if $\langle G_M, 1 \rangle$ is a positive instance of the Pumping-Problem w.r.t. Bar Hillel's lemma,

where G_M generates the language L_M described in proof of Theorem 11. Thus, we can state the following result:

Lemma 13. *The* EMPTY*-problem reduces to the* Pumping-Problem *for context-free languages w.r.t. Bar Hillel's pumping lemma via a many-one reduction.* □

In order to show Π_1^0-completeness it remains to show that the Pumping-Problem for context-free languages w.r.t. Lemma 1 is contained in Π_1^0. This is shown next—compare with [15]:

Lemma 14. *Given a context-free grammar G and a natural number p, it belongs to Π_1^0 to check whether for the language $L(G)$ the statement of Lemma 1 for the value p holds, i.e., the problem is co-recursively enumerable.*

Proof. The encoding $\langle G, p \rangle$ is a *negative* instance of the Pumping-Problem w.r.t. Bar Hillel's lemma if and only if there exists a word z of length at least p, with $z \in L(G)$, such that for each 5-tuple of words (u, v, w, x, y) with

1. $z = uvwxy$,
2. $|vwx| \leq p$, and $|vx| \geq 1$

there exists an integer i such that uv^iwx^iy is not in $L(G)$.

We find it easier to describe a nondeterministic Turing machine which accepts all negative instances. To begin Turing machine nondeterministically guesses a word z of length at least p, and verifies that $z \in L(G)$ using the well-known CYK's algorithm. If this is not the case, the Turing machine immediately rejects on this computation path. Then it generates a list of all 5-tuples (u, v, w, x, y) of words which constitute a decomposition $z = uvwxy$ and satisfy both $|vwx| \leq p$ and $|vx| \geq 1$. For each of these decompositions, the machine guesses a nonnegative integer i, and verifies that $uv^iwx^iy \notin L$. If this is not the case, the machine immediately rejects on this computation path. After all such decompositions are verified, the Turing machine accepts on this computation path.

It is clear from the above description that an accepting computation witnesses the existence of a word z of length at least p with $z \in L(G)$, such that each p-admissible decomposition of z can be pumped to a word not in $L(G)$. Conversely, from a given word z of length at least p in $L(G)$ and a list of exponents $i_1, i_2, \ldots, i_j, \ldots$, such that for each admissible decomposition $w = u_jv_jw_jx_jy_j$ the pumped word $u_j{v_j}^{i_j}w_j{x_j}^{i_j}y_j$ is not in $L(G)$, we can construct an accepting computation of the nondeterministic Turing machine described above.

Since there is a nondeterministic Turing machine which halts exactly on the negative instance of the Pumping-Problem w.r.t. Bar Hillel's lemma, we can see that the set of positive instances is co-semidecidable, and thus in Π^0_1. □

In summary, we have obtained the following result:

Theorem 15. *Given a context-free grammar G and a natural number p to check whether for the language $L(G)$ the statement of Lemma 1 for the value p holds. is Π^0_1-complete.* □

4 Conclusion

In this paper, we continued our research on the Pumping-Problem, recently started in [7]. Here we focused on Bar-Hillel's pumping lemma for context-free languages and its variants. It turned out that the Pumping-Problem for context-free languages w.r.t. Bar-Hillel's lemma is undecidable and complete for the level Π^0_1 of the arithmetical hierarchy. It remains undecidable even if regular pumping is considered. On the other hand, context-free pumping becomes decidable if the underlying language family is regular. This result relies heavily on the congruence representation of regular languages. Whether this result can be extended to other language families that allow for some congruence representation, such as the family of visibly pushdown languages [1, 2], is subject of further research. Furthermore, for the decidable variants of the problem, it remains to consider their computational complexity, which would nicely complement our previous investigations mentioned above.

References

1. Alur, R., Kumar, V., Madhusudan, P., Viswanathan, M.: Congruences for visibly pushdown languages. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C.,

Yung, M. (eds.) Proceedings of the 32nd International Colloquim Automata, Languages and Programming. pp. 1102–1114. No. 3580 in LNCS, Springer, Lisbon, Portugal (2005)
2. Alur, R., Madhusudan, P.: Adding nesting structure to words. J. ACM **56**(3), Art. 16 (2009)
3. Bar-Hillel, Y., Perles, M., Shamir, E.: On formal properties of simple phrase structure grammars. Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung **14**, 143–177 (1961)
4. Brauer, W.: Automatentheorie: Eine Einführung in die Theorie endlicher Automaten. Leitfäden und Monographien der Informatik, Teubner Stuttgart (1984). https://doi.org/10.1007/978-3-322-92151-2, (in German)
5. Dassow, J., Jecker, I.: Operational complexity and pumping lemmas. Acta Inform. **59**, 337—355 (2022). https://doi.org/10.1007/s00236-022-00431-3
6. Ginsburg, S.: The Mathematical Theory of Context-Free Languages. McGraw-Hill (1966)
7. Gruber, H., Holzer, M., Rauch, C.: The pumping lemma for regular languages is hard. In: Nagy, B. (ed.) Proceedings of the 27th International Conference on Implementation and Application of Automata. pp. 128–140. No. 14151 in LNCS, Springer, Famagusta, Cyprus (2023). https://doi.org/10.1007/978-3-031-40247-0˙9
8. Harrison, M.A.: Introduction to Formal Language Theory. Addison-Wesley (1978)
9. Hartmanis, J.: Context-free languages and Turing machine computations. In: Proceedings of Symposia in Applied Mathematics. vol. 19, pp. 42–51. American Mathematical Society, Providence, Rhode Island (1967)
10. Holzer, M., Rauch, C.: On Jaffe's pumping lemma, revisited. In: Bordihn, H., Tran, N., Vaszil, G. (eds.) Proceedings of the 25th International Conference on Descriptional Complexity of Formal Systems. pp. 65–78. No. 13918 in LNCS, Springer, Potsdam, Germany (2023). https://doi.org/10.1007/978-3-031-34326-1˙5
11. Holzer, M., Rauch, C.: On minimal pumping constants for regular languages. In: Gazdag, Z., Iván, S., Kovásznai, G. (eds.) Proceedings of the 16th International Conference on Automata and Formal Languages. pp. 127–141. No. 386 in EPTCS, Eger, Hungary (2023). https://doi.org/10.4204/EPTCS.386.11
12. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
13. Horváth, S.: The family of languages satisfying Bar-Hillel's lemma. RAIRO–Informatique théorique et Applications / Theoretical Informatics and Applications **12**(3), 193–199 (1978)
14. Jaffe, J.: A necessary and sufficient pumping lemma for regular languages. SIGACT News **10**(2), 48–49 (Sommer 1978). https://doi.org/10.1145/990524.990528
15. Kalociński, D.: On computability and learnability of the pumping lemma function. In: Dediu, A.H., Martín-Vide, C., Sierra-Rodríguez, J.L., Truthe, B. (eds.) Proceedings of the 8th International Conference Language and Automata Theory and Applications. pp. 433–440. No. 8370 in LNCS, Springer, Madrid, Spain (2014)
16. Kozen, D.C.: Automata and Computability. Undergraduate Texts in Computer Science, Springer (1997). https://doi.org/10.1007/978-1-4612-1844-9
17. Lange, M., Leiß, H.: To CNF or not to CNF? An efficient yet presentable version of the CYK algorithm. Informatica Didactica **8** (2009)
18. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM J. Res. Dev. **3**, 114–125 (1959). https://doi.org/10.1147/rd.32.0114
19. Rogers, Jr., H.: Theory of Recursive Functions and Effective Computability. Higher Mathematics, McGraw-Hill (1967)

20. Sommerhalder, R.: Classes of languages proof against regular pumping. RAIRO–Informatique théorique et Applications / Theoretical Informatics and Applications **14**(2), 169–18 (1980)