Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 387 (2007) 167–176

Theoretical Computer Science

www.elsevier.com/locate/tcs

The size of Higman–Haines sets[☆]

Hermann Gruber[a], Markus Holzer[b,*], Martin Kutrib[c]

[a] *Institut für Informatik, Ludwig-Maximilians-Universität München, Oettingenstraße 67, D-80538 München, Germany*
[b] *Institut für Informatik, Technische Universität München, Boltzmannstraße 3, D-85748 Garching bei München, Germany*
[c] *Institut für Informatik, Universität Giessen, Arndtstraße 2, D-35392 Giessen, Germany*

Abstract

We show that for the family of Church–Rosser languages the Higman–Haines sets, which are the sets of all scattered subwords of a given language and the sets of all words that contain some word of a given language as a scattered subword, cannot be effectively constructed, although both these sets are regular for *any* language. This nicely contrasts the result on the effectiveness of the Higman–Haines sets for the family of context-free languages. The non-effectiveness is based on a non-recursive trade-off result between the language description mechanism of Church–Rosser languages and the corresponding Higman–Haines sets, which in turn is also valid for all supersets of the language family under consideration, and in particular for the family of recursively enumerable languages. Finally for the family of regular languages we prove an upper and a matching lower bound on the size of the Higman–Haines sets in terms of nondeterministic finite automata.
© 2007 Published by Elsevier B.V.

Keywords: Finite automata; Higman's theorem; Well-partial order; Descriptional complexity; Non-recursive trade-offs

1. Introduction

A not so well-known theorem in formal language theory is that of Higman [6, Theorem 4.4], which reads as follows:

> If X is any set of words formed from a finite alphabet, it is possible to find a *finite* subset X_0 of X such that, given a word w in X, it is possible to find w_0 in X_0 such that the letters of w_0 occur in w in their right order, though not necessarily consecutively.

In fact, this statement is a corollary to a more general theorem on well-partially-ordered sets. Here a partially-ordered set is called well-partially-ordered, if every non-empty subset has at least one, but not more than a finite number of minimal elements (finite basis property). For instance, the set A^*, where A is a finite alphabet, under the scattered subword relation $\leq$, i.e., $v \leq w$ if and only if $v = v_1 \ldots v_k$ and $w = w_1 v_1 \ldots w_k v_k w_{k+1}$, for some integer k, where v_i and w_j are in A^*, for $1 \leq i \leq k$ and $1 \leq j \leq k+1$, is a well-partially-ordered set. Interestingly, the concept of

[☆] This is a completely revised and expanded version of a paper presented at the 8th Workshop on Descriptional Complexity of Formal Systems (DCFS) held in Las Cruces, New Mexico, USA, June 21–23, 2006.

[*] Corresponding author.
E-mail addresses: gruberh@tcs.ifi.lmu.de (H. Gruber), holzer@in.tum.de (M. Holzer), kutrib@informatik.uni-giessen.de (M. Kutrib).

0304-3975/$ - see front matter © 2007 Published by Elsevier B.V.
doi:10.1016/j.tcs.2007.07.036

well-partially-orders has been frequently rediscovered, for example, see [5,6,9,15,16]. Moreover, although Higman's result appears to be only of theoretical interest, it has some nice applications in formal language theory. It seems that one of the first applications has been given by Haines in [5, Theorem 3], where it is shown that the set of all scattered subwords, i.e., the *Higman–Haines set* $\text{DOWN}(L) = \{v \in A^* \mid \text{ there exists } w \in L \text{ such that } v \leq w\}$, and the set of all words that contain some word of a given language, i.e., the *Higman–Haines set* $\text{UP}(L) = \{v \in A^* \mid \text{there exists } w \in L \text{ such that } w \leq v\}$, are both regular for *any* language $L \subseteq A^*$. As pointed out in [5] this is an exceptional property, which is quite unexpected. It is worth mentioning that the regular languages $\text{DOWN}(L)$ and $\text{UP}(L)$ cannot be obtained constructively in general (in terms of finite automata). This is definite, because L is empty if and only if $\text{DOWN}(L)$ and $\text{UP}(L)$ are empty, but the question of whether or not a language is empty is undecidable for recursively enumerable languages and decidable for regular ones. A direct application to formal language theory, to be more precise, to parallel rewriting, has been given in [15], where based on the regularity of the aforementioned set a large class of ET0L-grammars are identified, which can generate regular languages only. Yet another application of Higman's theorem was used to obtain a shrinking lemma for indexed languages [4] and recently a characterization of the recursively enumerable languages in terms of context-free programmed grammars with unconditional transfer working under leftmost derivations of a certain type [3] was given. For some generalizations on Higman's result in formal language theory we refer to [8].

Although the basic results for Higman–Haines sets date back to the 1950s and 1960s, surprisingly less is known with respect to the (descriptional) size of these sets. To our knowledge the only paper dealing with effective constructibility issues is [16], where an open problem raised in [5] has been solved, i.e., $\text{DOWN}(L)$ can effectively be constructed for a given context-free grammar G with $L = L(G)$. In fact, the presented result is slightly more general than stated here. Moreover, it was also shown that $\text{UP}(L)$ can be obtained effectively, if L is a context-free language. This is the starting point of our investigations, because the effectiveness of the Higman–Haines sets for context-free languages immediately raises the question whether a similar result holds for the family of Church–Rosser languages. This language family lies in between the regular languages and the growing context-sensitive languages, but is incomparable to the family of context-free languages [1]. In fact, we show that for Church–Rosser languages the size of the Higman–Haines sets cannot be bounded by any recursive function; hence we obtain a non-recursive trade-off result between the language description mechanism and the corresponding Higman–Haines set. This non-recursive trade-off result implies that the Higman–Haines sets cannot effectively be constructed for Church–Rosser languages and all of its supersets. It turns out that the problem to decide whether a given regular language is the Higman–Haines set $\text{DOWN}(L)$ of a given recursively enumerable language is closely related to the infiniteness problem for Turing machine languages. In particular, we determine the exact level of unsolvability in the arithmetic hierarchy for the problem in question, and show that it is Π_2-complete. In case of the Higman–Haines set $\text{UP}(L)$ a similar result is obtained, namely Δ_2-completeness. For the size of the Higman–Haines set generated by regular languages matching upper and lower bounds are presented. That is, we prove that a linear blow-up is sufficient and necessary in the worst case for a nondeterministic finite automaton to accept the Higman–Haines set $\text{DOWN}(L)$ or $\text{UP}(L)$ generated by some language that is represented by another nondeterministic finite automaton.

The paper is organized as follows. The next section contains preliminaries and basics on Higman–Haines sets. Then Section 3 deals with decidability questions connected to the trade-offs in size between recursively enumerable, context-sensitive, growing context-sensitive, and Church–Rosser languages and Higman–Haines sets. After that, the upper and matching lower bounds on the size of the Higman–Haines set for regular languages in terms of nondeterministic finite automata size are shown. Finally we summarize our results and conclude with some open problems.

2. Preliminaries

We denote the set of non-negative integers by $\mathbb{N}$. The powerset of a set S is denoted by 2^S. For an alphabet A, let A^+ be the set of non-empty words w over A. If the empty word λ is included, then we use the notation A^*. For the length of w we write $|w|$. Set inclusion and strict set inclusion are denoted by $\subseteq$ and $\subset$, respectively.

Let $v, w \in A^*$ be words over alphabet A. We define $v \leq w$ if and only if there are words $v_1, v_2, \ldots, v_k$ and $w_1, w_2, \ldots, w_{k+1}$, for some $k \geq 1$, $v_i \in A^*$, $w_i \in A^*$, such that $v = v_1 v_2 \ldots v_k$ and $w = w_1 v_1 w_2, v_2 \ldots w_k v_k w_{k+1}$. In case $v \leq w$ we say that v is a scattered subword of w. Let L be a language over alphabet A. Then

$$\text{DOWN}(L) = \{v \in A^* \mid \text{ there exists } w \in L \text{ such that } v \leq w\}$$

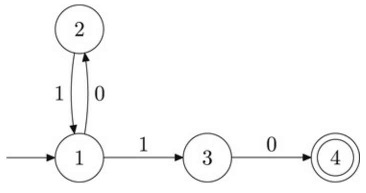

Fig. 1. A minimal NFA of size eight accepting the language L'.

and

$$\textsc{Up}(L) = \{v \in A^* \mid \text{ there exists } w \in L \text{ such that } w \le v\}$$

are the *Higman–Haines sets* generated by L.

Example 1. Let $A = \{0, 1\}$. If $v = 10011$, then

$$\lambda, 0, 1, 00, 01, 10, 11, 001, 011, 100, 101, 111, 0011, 1001, 1011, 10011$$

are all of its scattered subwords. Moreover, the words

$$10011, 010011, 100011, 100101, 100110, 100111, 101011, 110011, \ldots$$

contain v as a scattered subword. Next consider the linear context-free language $L = \{\, 10^n 1^n \mid n \ge 1\,\}$ over the alphabet A. Then the Higman–Haines sets generated by L are

$$\textsc{Down}(L) = (\lambda + 1)0^*1^* \quad \text{and} \quad \textsc{Up}(L) = (0+1)^*1(0+1)^*0(0+1)^*1(0+1)^*.$$

Finally, let L' be the regular language $(01)^*10$ over the alphabet A. A minimal nondeterministic finite automaton accepting L' is depicted in Fig. 1. Then $\textsc{Down}(L') = (0 + 1)^*$, because for every word w we have $w \le (01)^{|w|}10$ and the latter word belongs to L'. Finally, $\textsc{Up}(L') = (0 + 1)^*1(0 + 1)^*0(0 + 1)^*$, which equals $0^*1^+0(0 + 1)^*$, since $10 \le w$ for every $w \in L'$ and the word 10 is the minimal element of the scattered subword relation w.r.t. the language L'.

The next theorem is the surprising result of Haines. It had been shown about half a century ago. Actually, it is a corollary of Higman's work, but let us state it as a theorem.

Theorem 2 ([5,6]). *Let L be an arbitrary language. Then both* $\textsc{Down}(L)$ *and* $\textsc{Up}(L)$ *are regular.*

In fact, Higman's result as stated in the introduction rephrased in the notation introduced above reads as follows:

Lemma 3 ([6]). *Let L be an arbitrary language. Then there exists a natural number n, which depends only on L, such that* $\textsc{Up}(L) = \bigcup_{1 \le i \le n} \textsc{Up}(\{w_i\})$ *for some words $w_i \in L$ with $1 \le i \le n$.*

The statement of this lemma is sometimes called the *finite basis property*. The *basis of a language* w.r.t. the scattered subword relation is defined as follows: A word $w \in L$ is called *minimal* in L if and only if there is no $v \in L$ with $v \le w$ and $v \ne w$. The set of minimal elements in L is called a *basis* of the language L. Observe that any shortest word in L is minimal in L, and any such word must therefore be part of the basis.

It is worth mentioning that the Higman–Haines sets are closely related to the Straubing–Thérien hierarchy—see, e.g., [13]. In order to simplify our presentation we introduce the notation of the shuffle of two languages. Let x and y be two words over A. The *shuffle* of x and y, denoted by $x \sqcup\!\sqcup\, y$, is the set

$$\{x_1y_1x_2y_2 \ldots x_ny_n \mid \text{there is a } n \ge 1 \text{ such that } x = x_1x_2 \ldots x_n,\\ y = y_1y_2 \ldots y_n \text{ with } x_i, y_i \in A^*, \text{ for } 1 \le i \le n\}.$$

The shuffle of two languages $L_1, L_2 \subseteq A^*$ is

$$L_1 \sqcup\!\sqcup\, L_2 = \{w \in A^* \mid w \in x \sqcup\!\sqcup\, y \text{ for some } x \in L_1 \text{ and } y \in L_2\}.$$

Then every set L over alphabet A is a *shuffle ideal*, i.e., $L = L \sqcup\!\sqcup\, A^*$ and *vice versa*. Moreover, $L \subseteq A^*$ is a *shuffle co-ideal* if L is the complement of a shuffle ideal w.r.t. A^*. Note that Haines [5, Theorem 2b] has shown that every down-set is the complement of an up-set and *vice versa*. Since a language is of level $1/2$ of the Straubing–Thérien hierarchy if and only if it is a shuffle ideal, the level $1/2$ coincides with the family of languages of all up-sets $\{\,\textsc{Up}(L) \mid L \subseteq A^*\,\}$. On the other hand, the family of all down-sets $\{\textsc{Down}(L) \mid L \subseteq A^*\,\}$ equals the complements of the languages at level $1/2$. Hence from the algebraic point of view, all these languages are quite simple in nature.

The following lemma summarizes an easy observation about the relation of L and its Higman–Haines sets.

Lemma 4. *Let $L \subseteq A^*$ be an arbitrary language.* (1) *Then L is empty if and only if both* $\textsc{Down}(L)$ *and* $\textsc{Up}(L)$ *are empty.* (2) *Moreover, language L is finite if and only if the set* $\textsc{Down}(L)$ *is finite.* (3) *Finally, $\lambda \in L$ if and only if* $\textsc{Up}(L)$ *is universal, i.e.,* $\textsc{Up}(L) = A^*$.

Let D be a family of automata or grammars, satisfying that it is decidable whether a given string is a descriptor belonging to D or not. The *formal language represented* by $M \in D$ is denoted by $L(M)$ and the *family of languages represented by* D is $\mathscr{L}(D) = \{L(M) \mid M \in D\}$.

In order to talk about the economy of descriptions we first have to define what is meant by the *size of automata and grammars*. In general, we are interested in measuring the *length of the string that defines an automaton or grammar*. In particular, we sometimes use more convenient and common size measures, if there is a recursive upper bound for the length of the defining string dependent on the chosen size measure. For example, for *context-sensitive and context-free grammars M*, the size $|M|$ equals the total number of occurrences of terminal and non-terminal symbols in the productions. For *deterministic and nondeterministic finite automata M*, the size $|M|$ equals the product of the number of states and the number of input symbols. Clearly, the considered size measures imply total, recursive functions $c : D \to \mathbb{N}$, such that D is recursively enumerable in order of increasing size, and does not contain infinitely many members of the same size.

3. Undecidability of sufficient sizes for Higman–Haines sets

In this section we consider the size of finite automata (FA) sufficient to accept the Higman–Haines set generated by some language given by an automaton or grammar. The first family in question is recursively enumerable languages given by Turing machines or, equivalently, by phrase-structure grammars. The question for the sufficient size of an FA is closely related to decidability problems. For example, if it would be decidable whether a given FA accepts the Higman–Haines set of a given language L, then the sufficient size would be computable. The computation could simply enumerate all FAs in increasing order, and decide for every automaton whether it accepts $\textsc{Down}(L)$. Clearly, the size of the first matching automaton is a sufficient one. The assertion remains true even if the problem is semi-decidable only. Moreover, it is evident that one cannot effectively construct the Higman–Haines set generated by some Turing machine language L. If it were effectively constructible, one could decide the emptiness of Turing machine languages, since the Higman–Haines set $\textsc{Down}(L)$ is empty if and only if L is empty. A similar statement is valid for the Higman–Haines set $\textsc{Up}(L)$.

In order to deal with these questions, we recall some notations on computability theory [14]. A problem (or language) is called *decidable*, if there is a Turing machine that will halt on all inputs and, given an encoding of any instance of the question, will return the answer “yes” or “no” for the instance. The problem is *semi-decidable* or *recursively enumerable*, if the Turing machine halts on all instances for which the answer is “yes”. For example, the equivalence of two context-free languages given by context-free grammars is undecidable. But it is easy to see that the non-equivalence problem is semi-decidable.

As aforementioned, it is undecidable whether a given FA accepts the Higman–Haines set of a given language. So, we are interested in exploring how hard the problem is. Is it semi-decidable or is it on a higher level of unsolvability? To this end, we consider the *arithmetic hierarchy*, which is defined as follows:

$$\Sigma_1 = \{L \mid L \text{ is recursively enumerable}\},$$
$$\Sigma_{n+1} = \{L \mid L \text{ is recursive enumerable in some } A \in \Sigma_n\},$$

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. Let Π_n be the complement of Σ_n, i.e., $\Pi_n = \{\,L \mid \overline{L} \text{ is in } \Sigma_n\}$. Moreover, let $\Delta_n = \Sigma_n \cap \Pi_n$, for $n \geq 1$. Observe that $\Delta_1 = \Sigma_1 \cap \Pi_1$ is the class of all recursive sets. Completeness and hardness are always meant with respect to many-one reducibilities, if not otherwise stated.

A more revealing characterization of the arithmetic hierarchy can be given in terms of alternation of quantifiers. More precisely, a language L is in Σ_n, for $n \geq 1$, if and only if there exists a *decidable* $(n+1)$-ary predicate R such that

$$L = \{w \mid \exists y_1 \, \forall y_2 \, \exists y_3 \, \ldots \, Q \, y_n : R(w, y_1, y_2, \ldots, y_n)\},$$

where Q equals $\exists$ if n is odd, and Q equals $\forall$ if n is even. The characterization for languages in Π_n, for $n \geq 1$ is similar, by starting with a universal quantification and ending with an $\forall$ quantifier, if n is odd, and an $\exists$ quantifier, if n is even.

A well-known Σ_2-complete (Π_2-complete) problem, which we will refer to, is the finiteness (infiniteness) problem for Turing machines [14]. In fact, both problems are related to the down-set of a language, while it will turn out, that the up-set problem is slightly easier, namely related to Δ_2. The next lemma gives an upper bound of unsolvability for the Higman–Haines sets.

Lemma 5. *Given a Turing machine M and an FA M', the problem whether automaton M' accepts* $\textsc{Down}(L(M))$ $(\textsc{Up}(L(M))$*, respectively) is contained in Π_2 (Δ_2, respectively).*

Proof. First we consider the down-set problem: The Π_2 containment follows by the characterization of the arithmetic hierarchy in terms of alternation of quantifiers. The problem is equivalent to the statement: For all words v, there exists w and t, where w is a word and $t \geq 0$ a time step, such that

$$v \in L(M') \iff v \leq w \text{ and } M \text{ accepts } w \text{ in less than } t \text{ time steps.}$$

Observe that t is needed in order to obtain a decidable predicate.

For the up-set problem we argue as follows: Recall $\Delta_2 = \Sigma_2 \cap \Pi_2$. For the containment in Π_2 we similarly argue as in the proof above. The details are left to the reader. Thus, it remains to show Σ_2 containment. By Lemma 3 the problem under consideration is equivalent to the statement: There exists a natural number n, words $w_1, w_2, \ldots, w_n$, and a time step $t \geq 0$, such that for all words w and time steps $t' \geq 0$ we have

(1) that each word w_i is accepted by M in less than t steps,
(2) $L(M') = \bigcup_{1 \leq i \leq n} \textsc{Up}(\{w_i\})$, and
(3) word w is *not* accepted by M in less than t' steps or w belongs to $L(M')$.

Conditions (1) and (2) imply that $L(M') \subseteq \textsc{Up}(L(M))$. Observe that condition (3) ensures that $\textsc{Up}(L(M)) \subseteq L(M')$. Namely, for every w we have $w \in \overline{L(M)} \cup L(M')$ if and only if $w \notin L(M) \cap \overline{L(M')}$. Hence if condition (3) is satisfied, then $L(M) \cap \overline{L(M')} = \emptyset$, which is equivalent to $L(M) \subseteq L(M')$, and in turn implies $\textsc{Up}(L(M)) \subseteq L(M')$, because the up-operation is idempotent. Therefore, $L(M') = \textsc{Up}(L(M))$. All the constructions where finite automata are involved are effectively computable. Hence the above predicate is decidable, and thus, the assertion follows. □

Since Π_2 contains the complements of the Σ_2 languages, and hence Δ_2 is closed under complement, we obtain immediately:

Corollary 6. *Given a Turing machine M and an FA M', the problem whether automaton M' does not accept* $\textsc{Down}(L(M))$ $(\textsc{Up}(L(M))$*, respectively) is contained in Σ_2 (Δ_2, respectively).*

The next goal is to prove lower bounds on the level of unsolvability, i.e., to prove completeness of the problems. First we turn our attention to the down-set problem. We show a more general result that reduces infiniteness problems of language families to the problem in question. The next theorem follows by part (2) of Lemma 4 and the fact that finiteness is decidable for regular languages. Nevertheless we give an alternative proof, whose construction is interesting in its own.

Theorem 7. *Let D be a family of automata or grammars. Given $M \in D$ and an FA M', the problem whether M' accepts* $\textsc{Down}(L(M))$ *is at least as hard as the problem whether M accepts an infinite language.*

Proof. In order to solve the infiniteness problem, we start to construct a Turing machine U on unary inputs as follows. The machine expects inputs which are unary encodings of inputs for M. If the input is not a valid encoding, machine U loops forever. Otherwise it decodes the input and starts to simulate M. So, U accepts infinitely many inputs if and only if M does. Next we construct a unary FA M' that accepts all inputs. In order to complete the proof we conclude as follows: If M' accepts $\textsc{Down}(L(U))$, then $\textsc{Down}(L(U))$ and, therefore, languages $L(U)$ and $L(M)$ are infinite. If conversely $L(M)$ is infinite, then $L(U)$ is infinite. Since $L(U)$ is a unary language, we conclude in this case that $\textsc{Down}(L(U))$ contains all words over the unary alphabet. This implies that M' accepts $\textsc{Down}(L(U))$. $\square$

Now we immediately obtain the following completeness result for the down-set problem.

Theorem 8. *Given a Turing machine or a context-sensitive grammar M and a nondeterministic finite automaton M', the problem whether M' accepts* (*does not accept, respectively*) $\textsc{Down}(L(M))$ *is Π_2-complete* (*Σ_2-complete, respectively*).

Proof. It is well-known that the finiteness (infiniteness, respectively) problem for recursively enumerable as well as for context-sensitive languages is Σ_2-complete (Π_2-complete, respectively). $\square$

Now let us come to the up-set problem. In [16] it was shown that the effectiveness to determine the up-set of a language from a family of automata or languages D is closely related to the emptiness problem of $\mathscr{L}(D)$. To be more precise: Assume D to be effectively closed under intersection with regular languages. Then one can effectively determine $\textsc{Up}(L(M))$, for every $M \in D$, if and only if the emptiness problem for $\mathscr{L}(D)$ is decidable. The theorem to come shows that the lower bound of unsolvability is given by the emptiness problem for the language family under consideration. Since the proof is quite similar to that of Theorem 7, the details are left to the reader.

Theorem 9. *Let D be a family of automata or grammars. Given $M \in D$ and an FA M', the problem whether M' accepts* $\textsc{Up}(L(M))$ *is at least as hard as the problem whether M accepts the empty set.* $\square$

Since the emptiness problem for Turing machines is Π_1-complete, the previous theorem does not suffice to prove Δ_2-completeness of the up-set problem. Nevertheless, one can show the following hardness results.

Theorem 10. *Given a Turing machine M and an FA M', the problem whether automaton M' accepts* $\textsc{Up}(L(M))$ *is Σ_1- and Π_1-hard.*

Proof. We reduce the halting problem for Turing machines on some input w, i.e., $K_0 = \{\langle M, w\rangle \mid \text{M accepts w}\}$ to the problem under consideration. On input $\langle M, w\rangle$ we construct the Turing machine M_w such that $L(M_w) = L(M) \cap \{w\}$ and the FA M' with $L(M') = \textsc{Up}(\{w\})$. Then it is easy to see that $\langle M, w\rangle \in K_0$ if and only if M' accepts $\textsc{Up}(L(M_w))$. Thus, the problem is Σ_1-hard. For the Π_1-hardness we adapt the construction, changing automaton M' to accept the empty set. Then the Π_1-hardness easily follows. $\square$

If we use a more powerful reduction, namely Turing reducibility, then with a similar proof as above, one can reduce the Δ_2-complete problem $K = \{\langle M\rangle \mid M \text{ accepts the word } \langle M\rangle\}$ to the up-set problem. The completeness result reads as follows:

Corollary 11. *Given a Turing machine M and an FA M', the problem whether automaton M' accepts* $\textsc{Up}(L(M))$ *is Δ_2-complete under Turing reductions.* $\square$

For the family of context-sensitive languages we obtain the following completeness result:

Theorem 12. *Given a context-sensitive grammar M and an FA M', the problem whether automaton M' accepts* $\textsc{Up}(L(M))$ *is Π_1-complete.*

Proof. The Π_1 lower bound follows from Theorem 9. The containment within Π_1 is immediate, because for a given context-sensitive grammar M one can effectively construct a context-sensitive grammar accepting $\textsc{Up}(L(M))$, and then check equivalence of this grammar and the FA M'. Since the latter problem belongs to Π_1 for languages families with a decidable membership problem, the claim follows. $\square$

So far, we presented results concerning the completeness of decidability problems related to the sizes of Higman–Haines sets. One of the problems first investigated in connection with Higman–Haines sets was the question whether a regular representation of Higman–Haines sets can effectively be constructed from a given language, i.e., given an automaton or grammar M, can the FA that accepts $\textsc{Down}(L(M))$ or $\textsc{Up}(L(M))$ effectively be constructed. Clearly, the answer depends on the underlying family of automata or grammars from which we may choose M. As mentioned before, for recursively enumerable languages the Higman–Haines sets are not effectively constructible. On the other hand, in [16] the surprising result that the sets are constructible for context-free languages has been shown. The result is interesting, in particular since, for example, regularity is not decidable for context-free languages. This raises immediately the question for Church–Rosser languages, since they are incomparable (with respect to set inclusion) with context-free languages [1], but a proper superset of regular and a proper subset of growing context-sensitive languages. In the following we derive answers from relations between constructibility and decidability. In particular, given an automaton or grammar M, we are interested in the function $f : \mathbb{N} \to \mathbb{N}$, such that size $f(|M|)$ is sufficient for an FA to accept $\textsc{Down}(L(M))$ or $\textsc{Up}(L(M))$, respectively. It turns out that for certain devices f exceeds any recursive function. We start with a simple but fruitful lemma.

Lemma 13. *Let D be a family of automata or grammars. If for all $M \in D$ an FA accepting* $\textsc{Down}(L(M))$ *can effectively be constructed, then there exists a recursive function $f : \mathbb{N} \to \mathbb{N}$ such that size $f(|M|)$ is sufficient for an FA to accept* $\textsc{Down}(L(M))$. *A similar statement is valid for* $\textsc{Up}(L(M)$.

Proof. We only prove the statement for the down-set problem. Similar arguments apply for the up-set problem. Function f is computed as follows. Given some $n \geq 1$, all finitely many $M' \in D$ are enumerated whose size is at most n. Next a finite list is computed which contains for each M' an FA accepting $\textsc{Down}(L(M'))$. We define $f(n)$ to be the maximal size of the FAs appearing in the list. Clearly, function f is recursive and, since any n-size automaton (grammar) $M \in D$ appears in the list, size $f(|M|)$ is sufficient for an FA to accept $\textsc{Down}(L(M))$. □

The next results reveal once more the relations between infiniteness or emptiness and decidability of the Higman–Haines sets.

Theorem 14. *Let D be a family of automata or grammars.*

(1) *If there exists a recursive function $f : \mathbb{N} \to \mathbb{N}$ such that, for all $M \in D$, size $f(|M|)$ is sufficient for an FA to accept* $\textsc{Down}(L(M))$, *then infiniteness is semi-decidable for D.*
(2) *Let D have a decidable membership problem. If there exists a recursive function $f : \mathbb{N} \to \mathbb{N}$ such that, for all $M \in D$, size $f(|M|)$ is sufficient for an FA to accept* $\textsc{Up}(L(M))$, *then emptiness is decidable for D.*

Proof. Given some $M \in D$, first we compute the value $n = f(|M|)$ in order to obtain an upper bound for the size of an FA accepting $\textsc{Down}(L(M))$ or $\textsc{Up}(L(M))$. Next a list of all finitely many FAs whose sizes are at most n is created. Then we proceed as follows, according to whether we are interested in the down-set or up-set problem:

(1) For the down-set problem we argue in the following way. Since finiteness is decidable for regular languages, the list can be modified such that all FAs accepting infinite languages are deleted. From the finite number of remaining FAs that accept finite languages, respectively, the longest accepted word, say w, is determined. The language $L(M)$ is infinite if and only if $\textsc{Down}(L(M))$ is infinite. So, it suffices to semi-decide infiniteness of the language $\textsc{Down}(L(M))$. Due to the property of f, language $\textsc{Down}(L(M))$ is finite if and only if it is accepted by some FA M' in the list. Therefore, it is infinite if and only if it contains a word which is longer than w. Finally, M is simulated on all inputs longer than w by dove-tailing. If $L(M)$ is infinite, one of the simulations will accept which semi-decides infiniteness.
(2) In case of the up-set problem the list of FAs is modified such that all FAs accepting the empty language are deleted. From the finite number of remaining FAs that accept non-empty languages, respectively, for each automaton the shortest accepted word, say w, is determined. The language $L(M)$ is empty if and only if $\textsc{Up}(L(M))$ is empty. Recall that D has a decidable membership problem. Then verify whether at least one of these shortest words is accepted by M—this task is decidable by our assumption. If so, the language $L(M)$ is non-empty, otherwise $L(M)$ is empty. Hence emptiness is decidable. □

The next theorem is the main result of this section.

Theorem 15. *Let D be a family of automata or grammars which represent the recursively enumerable, recursive, context-sensitive, growing context-sensitive, or Church–Rosser languages. Then there does not exist a recursive function $f : \mathbb{N} \to \mathbb{N}$ such that, for all $M \in D$, size $f(|M|)$ is sufficient for an FA to accept* $\textsc{Down}(L(M))$ *or* $\textsc{Up}(L(M))$.

Proof. In [12] it has been shown that the Church–Rosser languages are characterized by so-called deterministic shrinking two-pushdown automata, where the nondeterministic variants are known to characterize the growing context-sensitive languages, which in turn are a proper subfamily of the deterministic context-sensitive languages [1]. Therefore, by Theorem 14 it suffices to show that infiniteness is not semi-decidable for Church–Rosser languages. We proceed similarly as in [7], where the proof utilizes results from [10,11].

In contrast to the assertion, assume that the infiniteness is semi-decidable. Then let U be an arbitrary Turing machine and w some input. In [11] it has been shown that for U and w one can effectively construct a Church–Rosser system T (or equivalently a deterministic shrinking two-pushdown automaton) and a word v, which is irreducible modulo T, such that U halts on input w if and only if $[v]_T$ is finite, where $[v]_T$ denotes the congruence class of v modulo T. Therefore, due to the assumption, it is semi-decidable whether a Turing machine U does *not* halt on a particular input, by verifying the infiniteness of a certain congruence class. This is a contradiction, since this would imply the decidability of the halting problem for Turing machines.

Second, for the language families under consideration with a decidable membership problem assume that a recursive trade-off between $M \in D$ and the FAs accepting $\text{UP}(L(M))$ exists. Then by Theorem 14 it follows that the emptiness problem for all these language families is decidable, a contradiction. If for the remaining family of recursively enumerable languages a recursive trade-off exists, then this bound also applies to one of the language families with a decidable word problem, a contradiction. □

Unfortunately, the converse of Lemma 13 is an open question. For example, if there is a recursive upper bound $f(|M|)$ for the size of an FA accepting the Higman–Haines set of some automaton M, then one can enumerate all finitely many FAs whose sizes are at most $f(|M|)$. Moreover, it is quite clear that one of these FAs is the one we are looking for. But the problem is to identify it. A rough idea could be to exclude all wrong automata from the list until the correct one remains. It is easy to exclude an FA that does not accept some word of the Higman–Haines set. But how to exclude FAs that accepts a superset of the Higman–Haines set?

Nevertheless, Lemma 13 gives answers to our problem, since together with Theorem 14 effective constructibility implies the semi-decidability of infiniteness. We obtain the next theorem.

Theorem 16. *Let D be a family of automata or grammars which represent the recursively enumerable, recursive, context-sensitive, growing context-sensitive, or Church–Rosser languages. Then given $M \in D$ there is no effective procedure to construct an FA that accepts* $\text{DOWN}(L(M))$ *or* $\text{UP}(L(M))$. □

4. Effective Higman–Haines set sizes

In this section we turn to the family of regular languages whose Higman–Haines sets can effectively be constructed. We are interested in the constructions themselves, as well as in the sizes of the Higman–Haines sets. Recall that we use the product of the number of states and the number of input symbols as size measure for nondeterministic finite automata (NFA). Let $M = (S, A, \delta, s_0, F)$ be an NFA, where S is the finite set of *internal states*, A is the finite set of *input symbols*, $s_0 \in S$ is the *initial state*, $F \subseteq S$ is the set of *accepting states*, and $\delta : S \times (A \cup \{\lambda\}) \to 2^S$ is the *partial transition function*. An NFA is deterministic (DFA) if and only if $|\delta(s, a)| \leq 1$ and $|\delta(s, a)| = 1 \iff |\delta(s, \lambda)| = 0$, for all $s \in S$ and $a \in A$.

Without loss of generality, we assume that the NFAs are always *reduced*. This means that there are no unreachable states and that from any state an accepting state can be reached.

In order to construct an NFA accepting $\text{DOWN}(L(M))$, at first the transition function δ of M is replaced by δ_1, where δ_1 provides all transitions of δ and, in addition, λ-transitions whenever δ provides a non-λ-transition, i.e.,

$$\forall s \in S, a \in A : \delta_1(s, a) = \delta(s, a)$$

and

$$\forall s \in S : \delta_1(s, \lambda) = \delta(s, \lambda) \cup \bigcup_{a \in A} \delta(s, a).$$

So, given an input v from $\text{DOWN}(L(M))$ such that $v \leq w$ and $w \in L(M)$, the new NFA simulates M on w in such a way that it guesses the missing input symbols and performs the corresponding transitions of M as λ-transitions. Moreover, since the NFA is still reduced, there is an accepting λ-path from every state. Therefore, we can define any

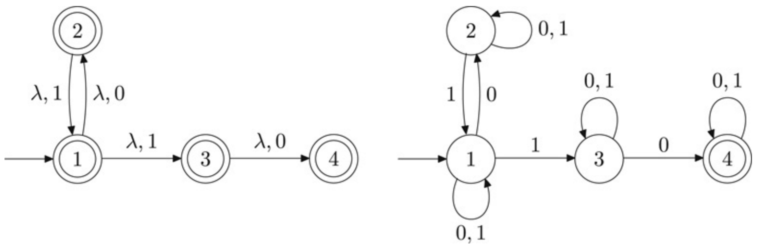

Fig. 2. An NFA (left) of size eight accepting the Higman–Haines set $\text{Down}(L')$ and an NFA (right) accepting the language $\text{Up}(L')$ also of size eight.

state to be an accepting state. Moreover, we safely may delete all λ-loops, i.e., all λ-transitions from a state to itself. It is easy to see that the new NFA accepts $\text{Down}(L(M))$.

Now we construct an NFA accepting $\text{Up}(L(M))$. The transition function δ of M is replaced by δ_1, where δ_1 provides all transitions of δ and, in addition, loops of all input symbols on all states, i.e.,

$$\forall s \in S, a \in A : \delta_1(s, a) = \delta(s, a) \cup \{s\}$$

and

$$\forall s \in S : \delta_1(s, \lambda) = \delta(s, \lambda).$$

On input v from $\text{Up}(L(M))$ such that $w \leq v$ and $w \in L(M)$, the new NFA simulates M on input w in such a way that it over-reads additional symbols and performs the introduced loop-transitions. Taking the original states as accepting, it is easy to see that the new NFA accepts $\text{Up}(L(M))$.

Example 17. Reconsider the regular language $L' = (01)^*10$. An NFA accepting the Higman–Haines set $\text{Down}(L')$ is depicted in Fig. 2 (left). By our previous investigations we know that $\text{Down}(L') = (0 + 1)^*$; hence the minimal NFA accepting $\text{Down}(L')$ has size two. The NFA accepting the Higman–Haines set $\text{Up}(L')$ is depicted in Fig. 2 (right). We know that $\text{Up}(L') = 0^*1^+0(0 + 1)^*$, and therefore the *minimal* NFA accepting $\text{Up}(L')$ has size at most 6. It is not hard to see that in fact any minimal NFA has that size.

Corollary 18. *For any NFA M of size n, one can effectively construct an NFA accepting* $\text{Down}(L(M))$ *or* $\text{Up}(L(M))$, *whose size is at most n.* □

It is not hard to see that the upper bound is tight in both cases. That is, it is also the lower bound in the worst case. The next lemma summarizes the results.

Lemma 19. *Let M be an NFA of size n. Then size n is necessary and sufficient in the worst case for an NFA M' to accept* $\text{Down}(L(M))$ *or* $\text{Up}(L(M))$. *The NFA M' can effectively be constructed.*

Proof. We use the finite languages $L_n = \{a^{n-1}\}$ with $n \geq 1$ as witnesses. Clearly, L_n is accepted by some n-state NFA M whose size is $|\{a\}| \cdot n$. Since the longest word in $\text{Down}(L_n) = \bigcup_{i=0}^{n-1}\{a^i\}$ has length $n - 1$, any NFA M' which accepts $\text{Down}(L_n)$ needs at least n states and, thus, has at least size n. A similar argument applies for the set $\text{Up}(L_n)$. □

5. Conclusions

We have investigated the size of Higman–Haines sets for the language families of recursively enumerable, recursive, context-sensitive, growing context-sensitive, and Church–Rosser languages. It turned out that in all these cases the size cannot be bounded by any recursive function. The key to this non-recursive trade-off result is the relation between infiniteness or emptiness and decidability of the Higman–Haines sets. For the family of regular languages we give a precise bound on the size of the NFA accepting the Higman–Haines sets.

The results of Higman and Haines raise a host of questions which relate to the special case of the scattered subword relation studied here. Since the aforementioned result only needs a well-partial-order, one may ask similar questions for other well-partially-ordered sets as, e.g., for the Parikh subword quasi-order or for monotone well-quasi-orders—see [2,8] for further results on these well-quasi-orders. Finally, let us mention that another direction of future research could be to consider linear context-free and context-free languages, because their Higman–Haines sets can be effectively constructed [16].

References

[1] Gerhard Buntrock, Friedrich Otto, Growing context-sensitive languages and Church–Rosser languages, Inform. Comput. 141 (1) (1998) 1–36.

[2] A. Ehrenfeucht, D. Haussler, G. Rozenberg, On regularity of context-free languages, Theoret. Comput. Sci. 27 (1983) 311–332.

[3] H. Fernau, F. Stephan, Characterizations of recursively enumerable sets by programmed grammars with unconditional transfer, J. Autom., Lang. Comb. 4 (2) (1999) 117–152.

[4] Robert H. Gilman, A shrinking lemma for indexed languages, Theoret. Comput. Sci. 163 (1966) 277–281.

[5] Leonard H. Haines, On free monoids partially ordered by embedding, J. Comb. Theory 6 (1969) 94–98.

[6] Graham Higman, Ordering by divisibility in abstract algebras, Proc. London Math. Soc. Series (2) 2 (1952) 326–336.

[7] Markus Holzer, Martin Kutrib, Jens Reimann, Descriptional complexity of deterministic restarting automata, in: C. Mereghetti, B. Palano, G. Pighizzini, D. Wotschke (Eds.), Descriptional Complexity of Formal Systems (DCFS 2005), Rapporto Tecnico 06-05, Università degli Studi di Milano, 2005, pp. 158–169.

[8] Lucian Ilie, Decision Problems on Orders of Words, Ph.D. Thesis, Department of Mathematics, University of Turku, Finland, 1998.

[9] Joseph B. Kruskal, The theory of well-quasi-ordering: A frequently discovered concept, J. Comb. Theory 13 (1972) 297–305.

[10] Robert McNaughton, Paliath Narendran, Friedrich Otto, Church-Rosser Thue systems and formal languages, J. ACM 35 (2) (1988) 324–344.

[11] P. Narendran, C. Ó'Dúnlaing, H. Rolletschek, Complexity of certain decision problems about congruential languages, J. Comput. System Sci. 30 (3) (1985) 343–358.

[12] G. Niemann, F. Otto, The Church–Rosser languages are the deterministic variants of the growing context-sensitive languages, Inform. Comput. 197 (2005) 1–21.

[13] Jean-Eric Pin, Syntactic semigroups, in: G. Rozenberg, A. Salomaa (Eds.), in: Handbook of Formal Languages, vol. 1, Springer, Berlin, 1997, pp. 679–746 (Chapter 10).

[14] H. Rogers, Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York, 1967.

[15] Jan van Leeuwen, A regularity condition for parallel rewriting systems, SIGACT News 8 (4) (1976) 24–27.

[16] Jan van Leeuwen, Effective constructions in well-partially-ordered free monoids, Discrete Math. 21 (1978) 237–252.