Connecting Partial Words and Regular Languages

Jürgen Dassow[1], Florin Manea[2⋆] and Robert Mercaş[1]

[1] Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik,
PSF 4120, D-39016 Magdeburg, Germany,
dassow@iws.cs.uni-magdeburg.de, robertmercas@gmail.com
[2] Christian-Albrechts-Universität zu Kiel, Institut für Informatik,
D-24098 Kiel, Germany,
flm@informatik.uni-kiel.de

Abstract. We initiate a study of languages of partial words related to regular languages of full words. First, we study the possibility of expressing a regular language of full words as the image of a partial-words-language through a substitution that only replaces the hole symbols of the partial words with a finite set of letters. Results regarding the structure, uniqueness and succinctness of such a representation, as well as a series of related decidability and computational-hardness results, are presented. Finally, we define a hierarchy of classes of languages of partial words, by grouping together languages that can be connected in strong ways to regular languages, and derive their closure properties.
Keywords: partial word, regular language, finite automaton, language of partial words.

1 Introduction

Two DNA strands attach one to the other, normally, in a complementary way according to their nucleotides. That is, each purine, A or G, creates a hydrogen bond with one complementary pyrimidine, T or C, respectively. But, sometimes, it is the case that this process goes wrong, allowing G-T bonds. Starting from this situation, and motivated by the need of having a way to recover (as much as possible) and work with a correct DNA sequence, Berstel and Boasson [1] suggested the usage of partial words as a suitable mathematical model. Partial words are words that beside regular letters contain an extra "joker" symbol, also called "hole" or "do-not-know" symbol, that matches all symbols of the original alphabet, which were investigated already in the 1970s [3]. Going back to the initial example, one could recover the information regarding a DNA sequence from the badly bonded pair of DNA strands by associating actual letters to the positions where the bonds were correct and holes to the positions where the bonds were not correctly formed. Besides the above motivation, partial words may find applications in other fields, as well; for instance, they can be seen

⋆ The work of Florin Manea is supported by the DFG grant 582014.

as data sequences that are corrupted either by white noise or other external factor, or even incomplete or insufficient information, that one needs in some process. In the last decade a lot of combinatorial and algorithmic properties of partial words have been investigated (see the survey [2], and the references therein). Surprisingly, so far, no study of classes of languages of partial words (or sets of partial words that have common features) was performed. The only work on a connected topic, that we are aware of, is [4]. There, the concept of restoration of punctured languages and several similarity measures between full-words-languages, related to this concept, were investigated. More precisely, puncturing a word means replacing some of its letters with holes; from a language of punctured words, its restoration was obtained by taking all the languages that can be punctured to obtain the respective language. The results of [4] regarded classes of full-words-languages defined by applying successively puncturing and restoration operations to classes of languages from the Chomsky hierarchy.

The study of the class of regular languages, the most restrictive class of the Chomsky-hierarchy, has been one of the central topics in theoretical computer science. This class of languages, defined usually either as the class of languages accepted by finite automata or as the class of languages described by regular expressions, was extensively studied throughout the last seventy years (starting from the early 1940s) and, besides its impact in theory (mostly in language theory, but also in complexity theory, for instance), it was shown to have a wide range of applications. Regular languages, and the various mechanisms used to specify them, were used, for instance, in compilers theory, circuit design, text editing, pattern matching, formal verification, DNA computing, or natural language processing (see [7]).

In this work, we aim to establish a stronger connection between the attractive notions mentioned above: partial words, on one side, and regular languages, on the other side. First, we show how we can (non-trivially) represent every regular language as the image of a regular language of partial words through a substitution that defines the letters that may replace the hole (called $\diamond$-substitution, in the following). Moreover, we show that such a representation can be useful: for some regular languages, there exist deterministic finite automata accepting languages of partial words that represent the full-word-language and are exponentially more succinct than the minimal deterministic finite automaton accepting that language. Unfortunately, it may also be the case when the minimal non-deterministic finite automaton accepting a language is exponentially more succinct than any deterministic automaton accepting a language of partial words representing the same language. Generally, automata accepting languages of partial words representing a given full-words language can be seen as intermediate between the deterministic finite automata and the non-deterministic automata accepting that language. We also present a series of algorithmic and complexity results regarding the possibility of representing a regular language as the image of a language of partial words through a $\diamond$-substitution.

Motivated by the above results, that connect in a meaningful way languages of partial words to regular languages of full words, and by the theoretical in-

terest of studying systematically such languages, we define a series of classes of languages of partial words. Each of these classes contains languages that can be placed in a particular strong relation with the regular languages. Further, we investigate these classes from a language theoretic point of view, show that they form a hierarchy, and establish their closure properties.[3]

We begin the paper with a series of basic definitions. Let V be a non-empty finite set of symbols called an *alphabet*. Each element $a \in V$ is called a *letter*. A *full word* (or, simply, word) over V is a finite sequence of letters from V while a *partial word* over V is a finite sequence of letters from $V \cup \{\diamond\}$, the alphabet V extended with the distinguished hole symbol $\diamond$. The *length* of a (partial) word u is denoted by $|u|$ and represents the total number of symbols in u; the number of occurrences of a symbol a in a (partial) word w is denoted $|w|_a$. The *empty (partial) word* is the sequence of length zero and is denoted by λ. We denote by V^* (respectively, $(V \cup \{\diamond\})^*$) the set of words (respectively, partial words) over the alphabet V and by V^+ (respectively, $(V \cup \{\diamond\})^+$) the set of non-empty words (respectively, non-empty partial words) over V. The catenation of two (partial) words u and v is defined as the (partial) word uv. Recall that V^* (where the alphabet V may include the $\diamond$ symbol) is the free monoid generated by V, under the operation of catenation of words; the unit element in this monoid is represented by the empty word λ. A language L of full words over an alphabet V is a subset of V^*; a language of partial words L over an alphabet V (that does not contain the $\diamond$ symbol) is a subset of $(V \cup \{\diamond\})^*$. Given a language L we denote by $\mathbf{alph}(L)$ (the alphabet of L) the set of all the letters that occur in the words of L; for the precision of the exposure, we say that a language L of full (respectively, partial) words is over V, with $\diamond \notin V$, if and only if $\mathbf{alph}(L) = V$ (respectively, $\mathbf{alph}(L) = V \cup \{\diamond\}$). For instance, $L = \{abb, ab\diamond\}$ has $\mathbf{alph}(L) = \{a, b, \diamond\}$, thus, is a language of partial words over $\{a, b\}$. Note that the catenation operation can be extended to languages; more precisely, if L_1 and L_2 are languages over V, we define their catenation $L_1L_2 = \{w_1w_2 \mid w_1 \in L_1, w_2 \in L_2\}$.

Let u and v be two partial words of equal length. We say that u is *contained* in v, denoted by $u \sqsubset v$, if $u[i] = v[i]$ for all $u[i] \in V$; moreover, u and v are *compatible*, denoted by $u \uparrow v$, if there exists a word w such that $u \sqsubset w$ and $v \sqsubset w$. These notions can be extended to languages. Let L and L' be two languages of partial words with $\mathbf{alph}(L) \cup \mathbf{alph}(L') = V \cup \{\diamond\}$ and $\diamond \notin V$. We say that L is *contained* in L', denoted $L \sqsubset L'$, if, for every word $w \in L$, there exists a word $w' \in L'$ such that $w \sqsubset w'$. We say that L is *compatible* to L', denoted $L \uparrow L'$, if, for each $w \in L$, there exists $w' \in L'$ such that $w \uparrow w'$ and, for each $v' \in L'$, there exists $v \in L$ such that $v' \uparrow v$.

A substitution is a mapping $h : V^* \to 2^{U^*}$ with $h(xy) = h(x)h(y)$, for $x, y \in V^*$, and $h(\lambda) = \{\lambda\}$; h is completely defined by the values $h(a)$ for all $a \in V$. A morphism is a particular type of a substitution for which $h(a)$ contains exactly one element for all $a \in V$; i.e., a morphism is a function $h : V^* \to U^*$ with $h(xy) = h(x)h(y)$ for $x, y \in V^*$. A $\diamond$-substitution over V is a substitution

[3] A technical appendix containing full proofs of our results can be found at the webpage: `https://www.informatik.uni-kiel.de/zs/pwords`.

with $h(a) = \{a\}$, for $a \in V$, and $h(\diamond) \subseteq V$. Here we assume that $\diamond$ can replace any symbol of V.

In this paper, DFA stands for deterministic finite automaton and NFA for non-deterministic finite automaton; the language accepted by a finite automaton M is denoted $L(M)$. Also, the set of all the regular languages is denoted by **REG**; by $\mathbf{REG_{full}}$, we denote the set of all the regular languages of full words. Further definitions regarding finite automata and regular languages can be found in [7], while partial words are surveyed in [2].

2 Definability by Substitutions

Let us begin our investigation by presenting several results regarding the way regular languages can be expressed as the image of a language of partial words through a substitution.

Lemma 1. *Let $L \subseteq (V \cup \{\diamond\})^*\{\diamond\}(V \cup \{\diamond\})^*$ be a language of partial words and let σ be a $\diamond$-substitution over V. There exists a language L' such that $\sigma(L) = \sigma(L')$ and $|w|_\diamond = 1$ for all $w \in L'$.*

Lemma 2. *Let L be a regular language over V and let σ be a $\diamond$-substitution over V. Then there exists a maximal (with respect to set inclusion) language $L' \subseteq L$ which can be written as $\sigma(L'')$, where L'' is a language of partial words such that any word in L'' has exactly one hole. Moreover, L' and L'' are regular languages and, provided that L is given by a finite automaton accepting it, one can algorithmically construct a finite automaton accepting L' and L''.*

Proof. Let $\sigma(\diamond) = V'$.

We start by noting that a word w belongs to $\sigma(L'')$ for a language of partial words L'', whose elements contain at least one hole each, if and only if there exist the words x and y such that $w = xay$, for some $a \in V'$ and $\{x\}V'\{y\} \subseteq L$.

Now let $M = (Q, V, q_0, F, \delta)$ be a DFA accepting L.

Let $q \in Q$. We define the language R_q as follows. A word w of L is in R_q if and only if there exists the partial word $x \diamond y$, compatible with w, with $\delta(q_0, x) = q$, $S = \delta(q, V') \subseteq Q$, $\delta(S, y) \subseteq F$. Basically, R_q is the set of the words for which there exists a compatible partial word $x \diamond y$ with exactly one hole, such that x labels a path from q_0 to q in A and any word from $\{x\}V'\{y\}$ is in L.

Clearly, $R_q = \{x \mid x \in V^*, \delta(q_0, x) = q\}V'\{y \mid y \in V^*, \delta(q', y) \in F$ for all $q' \in \delta(q, V)\}$. It follows that R_q is regular and an automaton accepting this language can be constructed starting from M. Moreover, $R_q = \sigma(H_q)$, for $H_q = \{x \mid x \in V^*, \delta(q_0, x) = q\}\{\diamond\}\{y \mid y \in V^*, \delta(q', y) \in F$ for all $q' \in \delta(q, V)\}$.

Take now $L' = \cup_{q \in Q} R_q$ and $L'' = \cup_{q \in Q} H_q$. Then L' is regular, as all the languages R_q are regular, and $L' = \sigma(L'')$. We only have to show that L' is maximal. If there exists $L_1 \subset L$ and a language of partial words L_2, whose elements contain exactly one hole each, such that $\sigma(L_2) = L_1$, then for every word w of L_1 there exist the words x and y such that $x \diamond y \in L_2$ and $w \in \sigma(x \diamond y) = \{x\}V'\{y\} \subseteq \sigma(L_2) = L_1$. Thus, $w \in R_q$ for $q = \delta(q_0, x)$. Therefore, $L_1 \subseteq L'$. □

Note that the sets R_q with $q \in Q$ are not a partition of L, as they are not necessarily mutually disjoint.

Next we introduce two relations connecting partial-words-languages and full-words-languages.

Definition 1. *Let $L \subseteq V^*$ be a language and σ be a $\diamond$-substitution over V. We say that L is σ-defined by the language L', where $L' \subseteq (V \cup \{\diamond\})^*$ is a partial-words-language, if $L = \sigma(L')$. Moreover, we say that L is* essentially σ-defined *by L', where $L' \subseteq (V' \cup \{\diamond\})^*$, if $L = \sigma(L')$ and every word in L' contains at least a $\diamond$-symbol.*

Obviously, for any regular language L over V, there is a regular language L' of partial words and a $\diamond$-substitution σ over V such that $\sigma(L') = L$, i.e., L is σ-defined by L'. For instance, take the set L' of the words obtained by replacing in the words of L some occurrences of a symbol $a \in V$ by $\diamond$, and the $\diamond$-substitution σ over V that maps $\diamond$ to $\{a\}$. More relevant ways of defining a regular language, in the sense of Definition 1, are presented in this section. We begin by characterizing the essentially definable languages.

Assume that the regular language $L \subseteq V^*$ is essentially σ-definable for some $\diamond$-substitution σ over V. Then $\sigma(L') = L$ for some appropriate language L' such that any word of L' contains at least a hole. By Lemma 1, we get that there is a regular language L'' of partial words such that $\sigma(L'') = \sigma(L')$ and each word of L'' contains exactly one hole. Now by Lemma 2 and its proof we get immediately the following characterisation of σ-definable languages.

Theorem 1. *Let L be a regular language of full words over V and σ a $\diamond$-substitution over V. Then L is essentially σ-definable if and only if $L = \bigcup_{q \in Q} R_q$ (where R_q is given in the proof of Lemma 2).* □

We also easily get the following decidability results.

Theorem 2. *i) Given a regular language L over V and a $\diamond$-substitution σ over V, it is decidable whether L is essentially σ-definable.*
ii) Given a regular language L over V, one can algorithmically identify all $\diamond$-substitutions σ for which L is essentially σ-definable.

Proof. By the previous results, testing whether L is essentially σ-definable is equivalent to testing whether L and $L' = \cup_{q \in Q} R_q$ are equal. Because the equality of two regular languages is decidable, the first statement follows. The second statement follows by an exhaustive search in the (finite) set of all $\diamond$-substitutions σ over V for those that essentially define L. □

The following consequence of Lemma 2 is worth noting, as it provides a canonical non-trivial representation of regular languages.

Theorem 3. *Given a regular language $L \subseteq V^*$ and a $\diamond$-substitution σ over V, there exists a unique regular language $L_\diamond$ of partial words that fulfils the following three conditions: (i) $L = \sigma(L_\diamond)$, (ii) for any language L_1 with $\sigma(L_1) = L$ we have $\{w \mid w \in L_1, |w|_\diamond \geq 1\} \sqsubset \{w \mid w \in L_\diamond, |w|_\diamond \geq 1\}$, and (iii) $(L_\diamond \cap V^*) \cap \sigma(\{w \mid w \in L_\diamond, |w|_\diamond \geq 1\}) = \emptyset$.*

Proof. Using the sets defined in Lemma 2, take $L_\diamond = L'' \cup (L \setminus L')$. The conclusion follows easily. □

Motivated by this last result, we now turn to the descriptional complexity of representing a regular language of full words by regular languages of partial words. We are interested in the question whether there is a regular language $L \subseteq V^*$, a regular language $L' \subseteq (V \cup \{\diamond\})^*$, and a $\diamond$-substitution σ over V with $\sigma(L') = L$ such that the minimal DFA accepting L' has a (strictly) lower number of states than the minimal DFA accepting L? In other words, are there cases when one can describe in a more succinct way a regular language via a language of partial words and a substitution that define it? Moreover, can we decide algorithmically whether for a given regular language L there exist a language of partial words and a substitution providing a more succinct description of L?

Let L be a regular language of full words over V. We denote by $\min_{DFA}(L)$ the number of states of the (complete) minimal DFA accepting L. Furthermore, let $\min_{NFA}(L)$ denote the number of states of a minimal NFA accepting L. Moreover, for a regular language L let $\min^{\diamond}_{DFA}(L)$ denote the minimum number of states of a (complete) DFA accepting a regular language $L' \subseteq (V \cup \{\diamond\})^*$ (where $\diamond$ is considered as an input symbol) for which there exists a $\diamond$-substitution σ over V such that $\sigma(L') = L$.

We have the following relation between the defined measures.

Theorem 4. *i) For every regular language L we have*

$$\min_{DFA}(L) \geq \min^{\diamond}_{DFA}(L) \geq \min_{NFA}(L).$$

ii) There exist regular languages L such that

$$\min_{DFA}(L) > \min^{\diamond}_{DFA}(L) > \min_{NFA}(L).$$

By the previous result one can see that, for certain substitutions σ, minimal DFAs accepting languages of partial words that σ-define a given full-words-regular language can be seen as intermediate between the minimal DFA and the minimal NFA accepting that language: they provide a succinct representation of that language, while having a limited non-determinism.

In fact, one can show that the differences $\min_{DFA}(L) - \min^{\diamond}_{DFA}(L)$ and $\min^{\diamond}_{DFA}(L) - \min_{NFA}(L)$ can be arbitrarily large; more precisely, we may have an exponential blow-up with respect to both relations.

Theorem 5. *Let n be a natural number, $n \geq 3$. There exist regular languages L and L' such that $\min^{\diamond}_{DFA}(L) \leq n + 1$ and $\min_{DFA}(L) = 2^n - 2^{n-2}$ and $\min_{NFA}(L') \leq 2n + 1$ and $\min^{\diamond}_{DFA}(L') \geq 2^n - 2^{n-2}$.*

The following remark provides an algorithmic side of the results stated above.

Remark 1. Given a DFA accepting a regular language L we can construct algorithmically a DFA with $\min^{\diamond}_{DFA}(L)$ states, accepting a regular language of partial words L', and a $\diamond$-substitution σ over $\mathbf{alph}(L)$, such that L is σ-defined by L'. By exhaustive search, we take a DFA M with at most $\min_{DFA}(L)$

states, whose transitions are labelled with letters from an alphabet included in $\mathbf{alph}(L) \cup \{\diamond\}$, and a $\diamond$-substitution σ over $\mathbf{alph}(L)$. We transform M into an NFA accepting $\sigma(L(M))$ by replacing the transitions labelled with $\diamond$ by $|\sigma(\diamond)|$ transitions labelled with the letters of $\sigma(\diamond)$, respectively. Next, we construct the DFA equivalent to this NFA, and check whether it accepts L or not (that is, $\sigma(L(M)) = L$). From all the initial DFAs we keep those with minimal number of states, since they provide the answer to our question. It is an open problem whether such a DFA can be obtained by a polynomial time deterministic algorithm; however, we conjecture that the problem is computationally hard.

We conclude by showing the hardness of a problem related to definability.

Theorem 6. *Consider the problem P: "Given a DFA accepting a language L of full words, a DFA accepting a language L' of partial words, and a $\diamond$-substitution σ over $\mathbf{alph}(L)$, decide whether $\sigma(L') \neq L$." This problem is NP-hard.*

Proof. In [6], the following problem was shown to be NP-complete:
P': "Given a list of partial words $S = \{w_1, w_2, \ldots, w_k\}$ over the alphabet V with $|V| \geq 2$, each partial word having the same length L, decide whether there exists a word $v \in V^L$ such that v is not compatible with any of the partial words in S."

We show here how problem P' can be reduced in polynomial time by a many-one reduction to problem P. Indeed, take an instance of P': a list of partial words $S = \{w_1, w_2, \ldots, w_k\}$ over the alphabet V with $|V| \geq 2$, each having the same length L. We can construct in polynomial time a DFA M accepting exactly the language of partial words $\{w_1, w_2, \ldots, w_k\}$. Also, we can construct in linear time a DFA M' accepting the language of full words V^L. It is clear that for $L(M)$ and the substitution σ, mapping the letters of V to themselves and $\diamond$ to V, we have $\sigma(L(M)) \neq V^L$ (that is, the answer to the input M, M' and σ of problem P is positive) if and only if the answer to the given instance of P' is also positive. Since solving P' is not easier than solving P, we conclude our proof. □

Theorem 6 provides a simple way to show the following well known result.

Corollary 1. *The problem of deciding whether a DFA M and an NFA M' accept different languages is NP-hard.*

3 Languages of partial words

While the results of the last section study the possibility and efficiency of defining a regular language as the image of a (regular) language of partial words, it seems interesting to us to take an opposite point of view, and investigate the languages of partial words whose images through a substitution (or all possible substitutions) are regular. Also, languages of partial words compatible with at least one regular language (or only with regular languages) seem worth investigating.

The definitions of the first three classes considered in this section follow the main lines of the previous section. We basically look at languages of partial words that can be transformed, via substitutions, into regular languages.

Definition 2. *Let L be a language of partial words over V.*
1. *We say that L is $(\forall\sigma)$-regular if $\sigma(L)$ is regular for all the $\diamond$-substitutions σ over alphabets that contain V and do not contain $\diamond$.*
2. *We say that L is* **max***-regular if $\sigma(L)$ is regular, where σ is a $\diamond$-substitution over V' with $\sigma(\diamond) = V'$, and $V' = V$ if $V \neq \emptyset$, and V' is a singleton with $\diamond \notin V'$, otherwise.*
3. *We say that L is $(\exists\sigma)$-regular if there exists a $\diamond$-substitution σ over a non-empty alphabet V', that contains V and does not contain $\diamond$, such that $\sigma(L)$ is regular.*
The classes of all $(\forall\sigma)$-regular, **max***-regular, and $(\exists\sigma)$-regular languages are denoted by* $\mathbf{REG}_{(\forall\sigma)}$, $\mathbf{REG}_{\mathbf{max}}$, *and, respectively,* $\mathbf{REG}_{(\exists\sigma)}$.

We consider, in the following, two classes of languages of partial words that are defined starting from the concept of compatibility.

Definition 3. *Let L be a language of partial words over V.*
4. *We say that L is $(\exists)$-regular if exists a regular language L' of full words such that $L \uparrow L'$.*
5. *We say that L is $(\forall)$-regular if every language L' of full words such that $L \uparrow L'$ is regular.*
The class of all the $(\exists)$-regular languages is denoted $\mathbf{REG}_{(\exists)}$, *while that of $(\forall)$-regular languages by* $\mathbf{REG}_{(\forall)}$.

According to the definitions from [4], the $(\exists)$-regular languages are those whose restoration contains at least a regular language, while $(\forall)$-regular languages are those whose restoration contains only regular languages.

We start with the following result.

Theorem 7. *For every non-empty alphabet V with $\diamond \notin V$ there exist an undecidable language L of partial words over V, such that:*
i) $\sigma(L) \in \mathbf{REG}$ for all substitutions σ over V, and $\sigma'(L) \notin \mathbf{REG}$ for the $\diamond$-substitution σ' with $\sigma'(\diamond) = V \cup \{c\}$, where $c \notin V$.
ii) every language $L' \subseteq V^*$ of full words, which is compatible with L, is regular and there is an undecidable language $L'' \subseteq (V')^*$, where V' strictly extends V, which is compatible with L.

Proof. Let $L_1 \subseteq V^*$ be an undecidable language (for instance, L_1 can be constructed by the classical diagonalization technique $L_1 = \{a^n \mid$ the n^{th} Turing machine in an enumeration of the Turing machines with binary input does not accept the binary representation of $n\}$, where $a \in V$) and $L = V^* \cup \{\diamond w \mid w \in L_1\}$. Clearly, for any $\diamond$-substitution σ over V, we have $\sigma(L) = V^*$. However, if we take a letter $c \notin V$ and the $\diamond$-substitution σ' which replaces $\diamond$ by $V \cup \{c\}$ we obtain an undecidable language $\sigma'(L)$. This concludes the proof of (i). To show (ii) we just have to note that the only language contained in V^* compatible with L is V^*, and, if we take a letter $c \notin V$ and replace $\diamond$ by c (or, in other words, if we see $\diamond$ as the conventional symbol c), we obtain an undecidable language compatible with L. □

We can now show a first result regarding the classes previously defined.

Theorem 8. $\mathbf{REG} = \mathbf{REG}_{(\forall\sigma)} \subset \mathbf{REG_{max}}$.

Proof. It is rather clear that $\mathbf{REG}_{(\forall\sigma)} \subseteq \mathbf{REG_{max}}$.

Since **REG** is closed to substitutions it follows that $\mathbf{REG} \subseteq \mathbf{REG}_{(\forall\sigma)}$.

It is also not hard to see that $\mathbf{REG}_{(\forall\sigma)} \subseteq \mathbf{REG}$ (given a language L in $\mathbf{REG}_{(\forall\sigma)}$, one can take the special substitution that replaces $\diamond$ with a symbol that does not occur in $\mathbf{alph}(L)$ and obtain a regular language; therefore L is a regular language if $\diamond$ is seen as a normal symbol).

By Theorem 7, $\mathbf{REG_{max}}$ contains an undecidable language; indeed, given an non-empty alphabet V, the language L defined in its proof for V is in $\mathbf{REG_{max}}$ according to (i). The strictness of the inclusion $\mathbf{REG} \subsetneq \mathbf{REG_{max}}$ follows. □

The next result gives some insight on the structure of the class $\mathbf{REG_{max}}$.

Theorem 9. *Let $L \in \mathbf{REG_{max}}$ be a language of partial words over $V \neq \emptyset$ and σ the $\diamond$-substitution used in the definition of $\mathbf{REG_{max}}$. Then there exists a maximal language (with respect to set inclusion) $L_0 \in \mathbf{REG_{max}}$ of partial words over V such that $\sigma(L_0) = \sigma(L)$. Moreover, given an automaton accepting L, an automaton accepting L_0 can be constructed.*

It is also not hard to see that any language from $\mathbf{REG_{max}}$ whose words contain only holes is regular.

The following relation also holds:

Theorem 10. $\mathbf{REG_{max}} \subset \mathbf{REG}_{(\exists\sigma)} \subset \mathbf{REG}_{(\exists)}$.

Proof. The non-strict inclusions $\mathbf{REG_{max}} \subseteq \mathbf{REG}_{(\exists\sigma)} \subseteq \mathbf{REG}_{(\exists)}$ are immediate. We show now that each of the previous inclusions is strict.

Take $L = \{(ab)^n \diamond b(ab)^n \mid n \in \mathbf{N}\}$. Considering σ a $\diamond$-substitution as in the definition of $\mathbf{REG_{max}}$, we have $\sigma(L) \cap \{w \mid w \in \{a,b\}^*, w \text{ contains } bb\}$ is the language $\{(ab)^n bb(ab)^n\}$ which is not regular. Thus, $\sigma(L)$ is not regular, and L is not in $\mathbf{REG_{max}}$. However, it is clearly in $\mathbf{REG}_{(\exists\sigma)}$, as when we take the substitution $\sigma(a) = \{a\}$, $\sigma(b) = \{b\}$, and $\sigma(\diamond) = \{a\}$, we have $\sigma(L) = \{(ab)^{2n+1} \mid n \in \mathbf{N}\}$, which is a regular language. This shows that $\mathbf{REG_{max}} \subset \mathbf{REG}_{(\exists\sigma)}$.

Now, take $L = \{(ab)^n \diamond b(ab)^n \mid n \in \mathbf{N}\} \cup \{(ab)^n a \diamond (ab)^n \mid n \in \mathbf{N}\}$. This language is not in $\mathbf{REG}_{(\exists\sigma)}$ by arguments similar to above, but it is in $\mathbf{REG}_{(\exists)}$ as it is compatible with $\{(ab)^{2n+1} \mid n \in \mathbf{N}\}$. □

As already shown, all the languages in $\mathbf{REG}_{(\forall)}$ are in $\mathbf{REG} = \mathbf{REG}_{(\forall\sigma)}$; however, not all the languages in **REG** are in $\mathbf{REG}_{(\forall)}$. The following statement characterizes exactly the regular languages that are in $\mathbf{REG}_{(\forall)}$.

Theorem 11. *Let L be a regular partial-words-language over V. Then $L \in \mathbf{REG}_{(\forall)}$ if and only if the set $\{w \mid |w|_\diamond \geq 1, w \in L\}$ is finite.*

The previous result provides a simple procedure for deciding whether a regular partial-words-language is in $\mathbf{REG}_{(\forall)}$ or not. We simply check (taking as input a DFA for that language) whether there are finitely many words that contain $\diamond$ or not. If yes, we accept the input and confirm that the given language is in $\mathbf{REG}_{(\forall)}$; otherwise, we reject the input.

Theorem 11 has also the following consequence.

Theorem 12. $\mathbf{REG}_{(\forall)} \subset \mathbf{REG}$.

In many previous works (surveyed in [2]), partial words were defined by replacing specific symbols of full words by $\diamond$, in a procedure that resembles the puncturing of [4]. Similarly, in [5], partial words were defined by applying the finite transduction defined by a deterministic generalised sequential machine (DGSM) to full words, such that $\diamond$ appears in the output word. Accordingly, we can define a new class of partial-words-languages, $\mathbf{REG_{gsm}}$, using this automata-theoretic approach. Let L be a language of partial words over V, with $\diamond \in \mathbf{alph}(L)$; L is **gsm**-regular, and is in $\mathbf{REG_{gsm}}$, if there exists a DGSM M and a regular language L' such that L is obtained by applying the finite transduction defined by M to L'. It is not hard to show that $\mathbf{REG_{gsm}} = \mathbf{REG} \setminus \mathbf{REG_{full}}$.

By the Theorems 8,10,11, and 12 we get the following hierarchies:

$$\mathbf{REG_{full}} \subset \mathbf{REG}_{(\forall)} \subset \mathbf{REG} = \mathbf{REG}_{(\forall\sigma)} \subset \mathbf{REG_{max}} \subset \mathbf{REG}_{(\exists\sigma)} \subset \mathbf{REG}_{(\exists)}$$

$$\mathbf{REG} \setminus \mathbf{REG_{full}} = \mathbf{REG_{gsm}} \subset \mathbf{REG} = \mathbf{REG}_{(\forall\sigma)}$$

Finally, the closure properties of the defined classes are summarized in the following table. Note that y (respectively, n) at the intersection of the row associated with the class $\mathcal{C}$ and the column associated with the operation $\circ$ means that $\mathcal{C}$ is closed (respectively, not closed) under operation $\circ$. A special case is the closure of $\mathbf{REG_{max}}$ under union and concatenation: in general this class is not closed under these operations, but when we apply them to languages of $\mathbf{REG_{max}}$ over the same alphabet we get a language from the same class.

Class	$\cup$	$\cap$	$\cap\mathbf{REG}$	$\mathbf{alph}(L)^* \setminus L$	$*$	$\cdot$	ϕ	ϕ^{-1}	σ
$\mathbf{REG}_{(\forall)}$	y	y	y	n	n	n	n	n	n
$\mathbf{REG} = \mathbf{REG}_{(\forall\sigma)}$	y	y	y	y	y	y	y	y	y
$\mathbf{REG_{max}}$	n/y	n	n	n	y	n/y	n	n	n
$\mathbf{REG}_{(\exists\sigma)}$	n	n	n	n	y	n	n	n	n
$\mathbf{REG}_{(\exists)}$	y	n	n	y	y	y	n	n	n

References

1. J. Berstel and L. Boasson. Partial words and a theorem of Fine and Wilf. *Theoretical Computer Science*, 218:135–141, 1999.
2. F. Blanchet-Sadri. *Algorithmic Combinatorics on Partial Words.* Chapman & Hall/CRC Press, 2008.
3. M. J. Fischer and M. S. Paterson. String matching and other products. In *Complexity of Computation, SIAM-AMS Proceedings*, volume 7, pages 113–125, 1974.
4. G. Lischke. Restoration of punctured languages and similarity of languages. *Mathematical Logic Quarterly*, 52(1):20–28, 2006.
5. F. Manea and R. Mercaş. Freeness of partial words. *Theoretical Computer Science*, 389(1-2):265–277, 2007.
6. F. Manea and C. Tiseanu. Hard counting problems for partial words. In *LATA*, volume 6031 of *Lect. Notes Comput. Sci.*, pages 426–438. Springer-Verlag, 2010.
7. G. Rozenberg and A. Salomaa. *Handbook of Formal Languages.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

Technical Appendix

In the following we give complete proofs and a series of other remarks regarding the results of the paper. They are grouped according to the results to which they are connected. Any time we discuss the computational complexity of solving a problem, we use the classical unit-cost RAM model.

4 Definability by Substitutions

Lemma 1

Proof. We take $L' = \{w \mid w \in (V \cup \{\diamond\})^*, |w|_\diamond = 1$ and there exists $x \in L$ such that $x \sqsubset w\}$. That is, we obtain the words of L' in the following manner: for each word $x \in L$, for each hole of x, we produce words which contain x by replacing the other holes of x than the selected one in all the possible ways. For example, if $V = \{a, b\}$, from $x = a\diamond b\diamond$ we obtain the words $aab\diamond$, $abb\diamond$, $a\diamond ba$, and $a\diamond bb$.

It is clear that $\sigma(L') = \sigma(L)$.

It is worth noting that the language L' defined in Lemma 1 has an infinite number of words that contain $\diamond$-symbols if and only if L has an infinite number of words containing $\diamond$-symbols. □

Lemma 2.
The following remark completes the proof of this lemma.

Remark 2. We briefly analyse the complexity of constructing an NFA accepting L', given an automaton (NFA or DFA) accepting L. Let us denote $|Q|$ by n. DFAs accepting the languages R_q and H_q are constructed in time $\mathcal{O}(n^{|V'|}|V|)$. Indeed, constructing these automata assumes the construction of a DFA for the language $\{w \mid w \in V^*, \delta(q_0, w) = q\}$ (that can be done in $\mathcal{O}(n|V|)$ time by simply constructing a finite automaton that has the same structure as M but has the final state q instead of the set of final states of M) and of an automaton for $\{w' \mid w' \in V^*, \delta(q', w') \in F$ for all $q' \in \delta(q, V)\}$ (that can be done in time $\mathcal{O}(|V|n^{|V'|})$ by intersecting the $|V'|$ languages $R_{q,a} = \{w' \mid w' \in V^*, \delta(q', w') \in F$ for $q' = \delta(q, a)\}$ for $a \in V'$, accepted respectively by finite automata with n states each). Now we just have to construct an automaton accepting the union of the sets R_q (respectively, H_q) and we get an automaton accepting L' (respectively, L''). The overall time complexity of constructing an NFA accepting L' (respectively, L'') is $\mathcal{O}(|V|n^{|V'|+1})$ (and its number of states is $\mathcal{O}(n^{|V'|+1})$).

Theorem 2.
Note, as a completion of Theorem 2, that one can decide similarly the following properties:

- Given a regular language L and the substitution σ as in Theorem 2, is the maximal subset of L that is σ-essentially definable finite or not?

– Given a regular language L and the substitution σ as in Theorem 2, is the difference between L and the maximal subset of L that is σ-essentially definable finite or not?

Also, the test in the proof of Theorem 2 can be done algorithmically in time $\mathcal{O}(|V|n^{|V'|+2})$, as the time needed to determine the emptiness of the intersection of the complementary of L with L' is proportional with the product of $|V|$, n (the number of states of a DFA accepting L) and $\mathcal{O}(n^{|V'|+1})$ (the number of states of an NFA accepting L').

Theorem 4. i) For every regular language L we have

$$\min_{DFA}(L) \geq \min^{\diamond}_{DFA}(L) \geq \min_{NFA}(L).$$

Proof. We first show that $\min^{\diamond}_{DFA}(L) \geq \min_{NFA}(L)$. Let σ be a $\diamond$-substitution over V and L' be a language of partial words such that $\sigma(L') = L$. The number of states in a complete DFA accepting L' is always greater or equal to $\min_{NFA}(L)$. Indeed, the DFA accepting L' can be transformed into a NFA accepting L by replacing each transition labelled with $\diamond$ with transitions labelled with all letters of $\sigma(\diamond)$ and deleting the error state of the DFA, when existing, and all the transition towards it. Thus $\min^{\diamond}_{DFA}(L) \geq \min_{NFA}(L)$. Moreover, $\min_{DFA}(L) \geq \min^{\diamond}_{DFA}(L)$, since one can choose the substitution σ to be the $\diamond$-substitution over $\mathbf{alph}(L)$ that maps $\diamond$ to a, a symbol of V that appears in the words of L, and L' to be equal to L in which we replace all the occurrences of a with $\diamond$. Clearly, each DFA accepting L' can be transformed in a DFA accepting L and vice versa. Thus, $\min^{\diamond}_{DFA}(L)$ is smaller than the number of states of a minimal DFA accepting L', which, in fact, equals the $\min_{DFA}(L)$. □

ii) There exist regular languages L such that

$$\min_{DFA}(L) > \min^{\diamond}_{DFA}(L) > \min_{NFA}(L).$$

Proof. Consider first the language $L_1 = \{abbc, adbc, abbe, abde\}$. This language is defined by $L' = \{a\diamond bc, ab\diamond e\}$ and the $\diamond$-substitution σ over $\{a, b, c, d, e\}$ that maps $\diamond$ to $\{b, d\}$. The language L' is accepted by a minimal complete DFA with 8 states, while the minimal complete DFA of L has 9 states. In fact, L' is exactly the language $L_\diamond$ defined in Theorem 3, for the $\diamond$-substitution σ defined above. So, for this language we have $\min_{DFA}(L_1) = 9$ and $\min^{\diamond}_{DFA}(L_1) = 8$; also, it is not hard to see that $\min_{NFA}(L_1) = 7$. This language already exhibits the example we were looking for in our statement; however, it is somehow trivial, as the difference between the DFA accepting the language of partial words and the NFA accepting the language of full words is given only by the fact that the DFA is complete, thus, by the existence of an error state in the DFA. More relevant examples can be constructed. To this end, take the language $L_2 = \{a'b'b'c', a'd'b'c', a'b'b'e', a'b'd'e'\}$, where $V' = \{a', b', c', d', e'\}$ and $V \cap V' = \emptyset$. This language is basically a copy of L_1 so all the properties regarding the number of states of automata accepting L_1 are preserved for L_2. Take now the language $L = L_1L_2$. One can easily check that the minimal DFA for L has 16 states (the final state of the DFA accepting L_1 is put together with the initial state of

the DFA accepting L_2, and the new automata has only one error state). Also, $\min^{\diamond}_{DFA}(L) = 15$; intuitively, as V' and V are disjoint, we can only map $\diamond$ to either a subset of V or a subset of V', or we will accept words where letters from V' occur before letters from V. When $\diamond$ is mapped to either $\{b, d\}$ or $\{b', d'\}$ we obtain DFAs with 15 states, while in all the other cases we obtain larger ones. Finally, $\min^{\diamond}_{NFA}(L) = 13$ (by similar reasons as in the case of the DFAs). This concludes our proof. □

Theorem 5.
To show this result, we use the following lemmas:

Lemma 3. *Let n be a number greater than 3 and $M = (Q, \{a, b\}, \delta, 1, \{n\})$ the non-deterministic finite automaton defined by*

$$Q = \{1, 2, \ldots, n\},$$
$$\delta(i, a) = \{i + 1\} \text{ for } 1 \leq i < n,$$
$$\delta(i, b) = \{i + 1\} \text{ for } 1 < i < n - 1,$$
$$\delta(n, a) = \{1, 2\},\ \delta(1, b) = \{1\},\ \delta(n - 1, b) = \emptyset, \text{ and } \delta(n, b) = \{1\}.$$

The minimal deterministic finite automaton accepting $L(M)$ has $2^n - 2^{n-2}$ states.

Proof. The basic ideas of the proof are similar to those used in the proof of Theorem 1 from [F. R. Moore. *On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata.* IEEE Trans. Comput., 20:1211–1214, 1971.]
However, the overall discussion and most of the details are more complicated.

Let M' be the DFA obtained from M by applying the classical conversion technique:

$$M' = (2^Q, \{a, b\}, \delta', \{1\}, \{S \mid S \in 2^Q, S \cap F \neq \emptyset\})$$

with $\delta'(R, s) = \cup_{q \in R} \delta(q, s)$ for $s \in \{a, b\}$ and $R \in 2^Q$.

Let us see which states are reachable from the initial state in M'. Note first that $\delta(1, b) = 1$ and $\delta(t, a^\ell) = \{t + \ell\}$ for $1 \leq t \leq n - 1$ and $1 \leq \ell \leq n - t$. Moreover, for $t \geq 2$ we have $\delta(t, x) = \{t + |x|\}$ when $|x| \leq n - t$ and x ends with a.

We first show that all the states $R = \{q_1, \ldots, q_k\} \in 2^{Q \setminus \{n\}}$ with $q_i < q_{i+1}$ for $1 \leq i \leq k - 1$ are reachable from the initial state $\{1\}$. This can be shown by induction on k. We have already seen that the property holds for $k = 1$. If $k > 1$, we analyse two cases.

In the first case, assume that $q_2 - q_1 = 1$. Now, for $R' = \{q'_3, q'_4, \ldots, q'_k, n-1\}$, we have $\delta'(R', a^{q_2}) = R$ provided that $q'_\ell = q_\ell - q_2$ for all $3 \leq \ell \leq k$. Indeed,

$$\delta'(n - 1, a^{q_2}) = \delta'(1, a^{q_1 - 1}) \cup \delta'(2, a^{q_1 - 1}) = \{q_1\} \cup \{q_1 + 1\} = \{q_1, q_2\}$$

and $\delta'(q'_\ell, a^{q_2}) = \{q'_\ell + q_2\} = \{q_\ell\}$.

In the second case, $q_2 - q_1 > 1$. Let $q_2 - q_1 = t$; it is easy to see that $t \leq n - 2$. Take $R' = \{q'_3, \ldots, q'_k, n - 1\}$, where $q'_\ell = q_\ell = q_2$, for $3 \leq \ell \leq k$, and

$x = a^2b^{t-1}a^{q_1-1}$. Clearly, $|x| = q_2$. We will show that $\delta(R', x) = R$. Indeed, we first have

$$\begin{aligned}\delta'(n-1, x) &= \delta'(1, b^{t-1}a^{q_1-1}) \cup \delta'(2, b^{t-1}a^{q_1-1}) \\ &= \delta'(1, a^{q_1-1}) \cup \delta'(t+1, a^{q_1-1}) \\ &= \{q_1\} \cup \{t+1+q_1-1\} = \{q_1, q_2\}\end{aligned}$$

and, further, $\delta'(q'_\ell, x) = \{q'_\ell + |x|\} = \{q_\ell\}$ for $3 \leq \ell \leq k$. These two cases show that out property holds: each state $R \in 2^{Q\setminus\{n\}}$ is reachable from a state $R' \in 2^{Q\setminus\{n\}}$ such that $|R'| = |R| - 1$, so R is reachable from $\{1\}$, as well.

Further, we obtain that each state $R \in 2^{Q\setminus\{1\}}$ is reachable from 1. Clearly, a state $R = \{q_1, \ldots, q_k\}$, with $q_1 > 1$ and $q_k \leq n$, can be obtained as $\delta'(\{q_1 - 1, \ldots, q_{k-1}\}, a)$ and we already know that $\{q_1 - 1, \ldots, q_{k-1}\}$ is reachable.

Also, all the states $R \in 2^Q$ that contain $\{1, 2, n\}$ are reachable. Indeed, a state $R = \{1, 2, q_1, \ldots, q_k, n\}$, with $q_1 > 2$ and $q_k < n$, can be obtained as $\delta'(\{q_1 - 1, \ldots, q_k - 1, n\}, a)$ and we already know that $\{q_1 - 1, \ldots, q_k - 1, n\}$ is reachable.

Finally, we show that the states $R \in 2^Q$ that contain $\{1, n\}$ but do not contain 2 are not reachable. Assume, for the sake of a contradiction that there exists such a state R that is reachable. Clearly, in order to have that R contains n there must exist a state R' such that $\delta'(R', a) = R$. But, from $1 \in R$ we get that $n \in R'$; from this it follows that $2 \in R$, a contradiction.

Therefore, the states of M' that are reachable are either elements of $2^{Q\setminus\{n\}} \cup 2^{Q\setminus\{1\}}$ or subsets of Q that contain $\{1, 2, n\}$; their number is $2^{n-1} + 2^{n-2} + 2^{n-3}$. Denote by M'' the DFA obtained from M' by deleting all the unreachable states. Further we minimise the automaton M''.

Take P and R two different states of M''. Take $\ell \in (P \setminus R) \cup (R \setminus P)$. If $\ell \neq 1$, assume, without loss of generality, that $\ell \in P$. We have that $\delta'(P, a^{n-\ell})$ is final, while $\delta(R, a^{n-\ell})$ is not final. So, in this case, P and R are not equivalent with respect to the congruence defined by the language L. If $(P \setminus R) \cup (R \setminus P) = \{1\}$ assume, without loss of generality, that $1 \in P$; we obtain that $P = R \cup \{1\}$. If $n \notin R$, we have that $\delta'(P, b^{n-1}a^{n-1})$ is final, while $\delta'(R, b^{n-1}a^{n-1})$ is not final. If $n \in R$, we have that $\delta(P, x) = \delta(R, x)$; indeed, if $x = ay$ we get that $\delta(P, ay) = \delta(R, ay) \cup \delta(1, ay) = \delta(R, ay)$, as $\delta(1, ay) = \delta(2, y) \subseteq \delta(n, ay) \subseteq \delta(R, ay)$. It follows that two states P and R of M'' are equivalent if and only if $P = R \cup \{1\}$ and $n \in R$.

Therefore, after we apply the minimization algorithm on M'' we obtain a minimal DFA accepting $L(M)$ that has exactly $2^{n-1} + 2^{n-2}$ states. □

In the following we give the proof of Theorem 5. First, recall its statement:
Let n be a natural number, $n \geq 3$.
1. There exists a regular language L such that $\min^{\diamond}_{DFA}(L) \leq n+1$ and $\min_{DFA} = 2^n - 2^{n-2}$.
2. Also, there exists a regular language L' such that $\min_{NFA}(L') \leq 2n + 1$ and $\min^{\diamond}_{DFA}(L') \geq 2^n - 2^{n-2}$.

Proof. **1.** Take the language $L_1 = L(M)$ defined in Lemma 3. Take the DFA $M_\diamond = (Q, \{\diamond, a, b\}, \delta', 1, n)$ with

$$
\begin{aligned}
&Q' = \{1, \ldots, n, n+1\},\\
&\delta'(1,a) = 2,\ \delta'(1,b) = 1,\ \delta'(1,\diamond) = n+1,\\
&\delta'(i,\diamond) = i+1 \text{ for } 2 \le i \le n-2,\\
&\delta(i,s) = n+1 \text{ for } 2 \le i \le n-2 \text{ and } s \in \{a,b\},\\
&\delta(n-1,\diamond) = \delta(n-1,b) = n+1,\ \delta(n-1,a) = n,\\
&\delta(n,\diamond) = 1,\ \delta(n,b) = n+1,\ \delta(n,a) = 2, \text{ and}\\
&\delta(n+1,s) = n+1 \text{ for all } s \in \{a,b,\diamond\}.
\end{aligned}
$$

Note that M can be obtained from $M_\diamond$ be replacing all the transitions labelled with $\diamond$ with transitions labelled with a and b, by deletion of the error-state $n+1$ and of all the transitions related to it. It is not hard to see that L_1 is defined by the language $L(M_\diamond)$ of partial words and the substitution σ that maps a to $\{a\}$, b to $\{b\}$, and $\diamond$ to $\{a,b\}$.

Thus, we have $\min^{\diamond}_{DFA}(L_1) \le n+1$. However, from Lemma 3 we have $\min_{DFA}(L_1) = 2^n - 2^{n-2}$, and the conclusion follows.

2. Let L_2 be the language $f(L_1)$, where f is a morphism mapping a to a' and b to b'. The language L_2 is accepted by an NFA with n states that can be easily obtained from M (by replacing the transitions labelled with s with transitions labelled with s' for $s \in \{a,b\}$). Let L' be the language defined as the catenation of $L_1 \cup \{c\}$ and L_2, that is $L' = (L_1 \cup \{c\})L_2$; note that if L' contains a symbol a or b (respectively, a' or b') then it must contain at least n symbols from $\{a,b\}$ (respectively, $\{a',b'\}$). Clearly, $\min_{NFA}(L') \le 2n+1$. We show that $\min^{\diamond}_{DFA}(L') \ge 2^n - 2^{n-2}$.

For simplicity, denote $V = \{a,b,c,a',b'\}$ and $V_\diamond = V \cup \{\diamond\}$. Let $L_\diamond$ be a regular language included in $(V \cup \{\diamond\})^*$ and σ be a substitution that maps the elements of V to themselves and $\diamond$ to a subset of V, such that $\sigma(L_\diamond) = L'$. Let $M_\diamond$ be the minimal DFA accepting $L_\diamond$, and assume it has the transition function $\delta_\diamond$ and the initial state q_0.

Let us first analyse the case when $c \in \sigma(\diamond)$. We first assume that we have a transition $\delta_\diamond(q_1,\diamond) = q_2$, $q_1 \ne q_0$, and q_2 is co-accessible (that is, it is a state from which we can reach a final state of the automaton). As $M_\diamond$ is minimal, it has no inaccessible states, so q_1 is also accessible. It follows that there exist a string $z\diamond x \in L_\diamond$, with $z \ne \lambda$; thus, in $L' = \sigma(L_\diamond)$ there is the string $z'cx'$ with $z' \ne \lambda$, a contradiction. So, whenever we have a transition labelled with $\diamond$ it either starts from the initial state of $M_\diamond$ or it leads to the only state that is not co-accessible (once again, there is only one such state, as the automaton is minimal). Further, assume that we have a transition $\delta_\diamond(q_0,\diamond) = q_1$ and q_1 is co-accessible. It follows that $\sigma(\diamond) \subseteq \{a,b,c\}$, as otherwise we would have in L' words that start with a' or b'. Also, as c is always followed in L' by a word from L_2, the only transitions that leave q_1 and reach another state are labelled with letters from $\{a',b',c'\}$; but this means that $\sigma(\diamond) = \{c\}$, as, otherwise, we may have in L' words that

begin with a letter from $\{a, b\}$ which is immediately followed by a letter from $\{a', b'\}$. Assume now that $\delta_\diamond(q_0, c) = r$ where r is not co-accessible. In this case, the automaton obtained from $M_\diamond$ by changing the initial state from q_0 to q_1 is a DFA that accepts exactly L_2. This means that $M_\diamond$ has at least $2^n - 2^{n-2}$ states. If $\delta_\diamond(q_0, c) = q_2$ and q_2 is co-accessible, we delete from $M_\diamond$ all the transitions labelled with letters from $\{a', b', c, \diamond\}$ and we set as final states of the obtained DFA the states of $M_\diamond$ from which transitions labelled with a' or b' started. The new automaton is a DFA accepting L_1; therefore, it has at least $2^n - 2^{n-2}$ states. So, in this case as well, $M_\diamond$ has at least $2^n - 2^{n-2}$ states.

We continue with the case when $c \notin \sigma(\diamond)$. Assume first that $\sigma(\diamond) \cap \{a, b\} \neq \emptyset$ and $\sigma(\diamond) \cap \{a', b'\} \neq \emptyset$. Let $q_1 = \delta_\diamond(q_0, c)$; it is not hard to notice that q_1 is co-accessible. Note that there is no state q_2, accessible from q_1, such that there is a transition labelled with a, b, or $\diamond$ leaving from q_2; indeed, if such a transition would exist then we would have a word in L' that has a or b after c, a contradiction. It follows that the automaton obtained from $M_\diamond$ by changing the initial state from q_0 to q_1 is a DFA that accepts exactly L_2. So $M_\diamond$ has at least $2^n - 2^{n-2}$ states. The case when $\sigma(\diamond) \subseteq \{a, b\}$ follows in exactly the same manner. Finally, when $\sigma(\diamond) \subseteq \{a', b'\}$, we delete once more from $M_\diamond$ all the transitions labelled with letters from $\{a', b', c, \diamond\}$ and we set as final states of the obtained DFA the states of $M_\diamond$ from which transitions labelled with a' or b' started. We obtain a DFA accepting L_1 included in $M_\diamond$, so this DFA has at least $2^n - 2^{n-2}$ states.

This concludes the case analysis, and the proof of the second statement. □

This theorem takes, in fact, the first steps towards solving the problem of analysing the sets

$$D_n = \{m \mid \text{ there exists a regular language } L \text{ such that } \min^\diamond_{DFA}(L) = n \text{ and } \min_{DFA}(L) = m\}$$

and

$$H_n = \{m \mid \text{ there exists a regular language } L \text{ such that } \min^\diamond_{DFA}(L) = n \text{ and } \min_{NFA}(L) = m\}.$$

Corollary 1

Proof. The problem from Proposition 6 can be reduced to this problem in polynomial time, as each pair consisting in a DFA M accepting a language a partial words and a $\diamond$-substitution σ mapping $\diamond$ to an alphabet can be canonically transformed into an NFA accepting the language of full words $\sigma(L(M))$. Therefore, this problem is also NP-hard. □

Languages of Partial Words

Theorem 9.
Let $L \in \mathbf{REG_{max}}$ be a language of partial words over $V \neq \emptyset$ and σ the $\diamond$-substitution used in the definition of $\mathbf{REG_{max}}$. Then there exists a maximal

language (with respect to set inclusion) $L_0 \in \mathbf{REG_{max}}$ of partial words over V such that $\sigma(L_0) = \sigma(L)$. Moreover, given an automaton accepting L, an automaton accepting L_0 can be constructed.

Proof. First, it is not hard to see that the language L_0, we are looking for, is the union of all the languages L' that verify $\sigma(L') = \sigma(L)$. It only remains to show that it is regular and that we can construct an automaton accepting it.

Let L be a language in $\mathbf{REG_{max}}$, over an alphabet $V \neq \emptyset$. Furthermore, let $\sigma : V \cup \{\diamond\} \to 2^V$ be a substitution such that $\sigma(a) = \{a\}$ for all $a \in V$ and $\sigma(\diamond) = V$. We have that $\sigma(L)$ is a regular language. Therefore, there exists a finite automaton $A = (Q, V, q_0, F, \delta)$ that accepts $\sigma(L)$. For $S \subseteq Q$ and $U \subseteq V$, we set

$$\delta(S, U) = \cup_{q \in S}(\cup_{a \in U}\{\delta(q, a)\}).$$

We define the new finite automaton $A' = (2^Q, V \cup \{\diamond\}, \{q_0\}, 2^F, \delta')$ where δ' works as follows:

- $\delta'(S, a) = \cup_{q \in S}\delta(q, a)$, for $S \subseteq Q$ and $a \in V$.
- $\delta'(S, \diamond) = \cup_{q \in S}(\cup_{a \in V}\delta(q, a))$, for $S \subseteq Q$.

We determine the language accepted by A'. More precisely, we show that, if $w \in V \cup \{\diamond\}$, then $\delta'(S, w) = \delta(S, \sigma(w))$ for any $S \subseteq Q$. This proof is done by induction. If $w = \lambda$ or if $|w| = 1$, the conclusion follows immediately. Let $w' = ws$, with $s \in V \cup \{\diamond\}$. We have

$$\begin{aligned}\delta'(S, ws) &= \delta'(\delta'(S, w), s) = \delta'(\delta(S, \sigma(w)), s) \\ &= \delta(\delta(S, \sigma(w)), \sigma(s)) = \delta(S, \sigma(ws)) = \delta(S, \sigma(w')).\end{aligned}$$

Consequently, $w \in L(A')$ if and only if $\delta'(\{q_0\}, w) \subseteq F$ if and only if $\delta(q_0, \sigma(w)) \subseteq F$ if and only if $\sigma(w) \subseteq \sigma(L)$. Conversely, it is quite easy to see that, if $w \in L$, then $w \in L(A')$; moreover, it easily follows that any language L' such that $\sigma(L') = \sigma(L)$ is included in $L(A')$. So, $\sigma(L(A')) = \sigma(L)$ and we take $L_0 = L(A')$.

As L_0 contains $\sigma(L)$ it is clear that L_0 is in $\mathbf{REG_{max}}$. □

Remark 3. If L from the above theorem is contained in $\{\diamond\}^*$ we easily obtain that L is regular. Moreover, a language similar to L_0 could be constructed as follows. If a is a letter, then any language L' with $\mathbf{alph}(L) = \{\diamond, a\}$, for which it is true that $\{n \mid \text{there exists } w \in L, |w| = n\} = \{n \mid \text{there exists } w \in L', |w| = n\}$, is in $\mathbf{REG_{max}}$ and $\sigma(L') = \sigma(L)$ where σ is the $\diamond$-substitution that maps $\diamond$ to $\{a\}$; also, no other languages verify $\sigma(L') = \sigma(L)$ for this choice of σ. The union of all these languages is $L_0^a = \{w' \mid w' \in \{\diamond, a\}^*, \text{there exists } w \in L, |w| = |w'|\}$, which is clearly a regular language, and plays the role of L_0 from the above theorem. However, in the case when another letter b is chosen instead of a, we get a different language L_0^b. The difference from the case of the previous theorem is that we now have the freedom of choosing the alphabet over which the maximal language is.

Example 1. Although $\mathbf{REG_{max}}$ contains very general languages, there are simpler languages that are not part of this class. For instance, the context-free language $L = \{\diamond a^n b^n \mid n \in \mathbf{N}\}$ is trivially not in $\mathbf{REG_{max}}$. Also, as every language of partial words from the classes we defined is compatible with at least one regular language, it follows easily that all these languages have the constant growth property[4]. It follows that there are context-sensitive languages that are not contained in any of the classes $\mathbf{REG_{max}}$, $\mathbf{REG}_{(\exists\sigma)}$, or $\mathbf{REG}_{(\exists)}$. However, these classes also contain context-sensitive non-context-free languages (e.g., $\{a^n \diamond a^n \diamond a^n \mid n \in \mathbf{N}\}$).

Theorem 11. Let L be a regular partial-words-language over V. We have the following equivalence: $L \in \mathbf{REG}_{(\forall)}$ if and only if the set $\{w \mid |w|_\diamond \geq 1, w \in L\}$ is finite.

Proof. From the finiteness of the set $\{w \mid |w|_\diamond \geq 1,\ w \in L\}$ we obtain immediately that $L \in \mathbf{REG}_{(\forall)}$.

We show the other implication. Assume that L contains an infinite number of partial words that have at least a $\diamond$ symbol. We obtain that either there exists a word $w\diamond$ such that w contains no hole and the set $L \cap w\diamond(V \cup \{\diamond\})^*$ is infinite or there exists a word $\diamond w$ such that w contains no hole and $L \cap (V \cup \{\diamond\})^*\diamond w$ is infinite. We present here only the first case, as the other one follows in the same fashion.

Let $L' = \{x \mid w\diamond x \in L\}$. It is clear that L' is regular; also, L' is infinite according to the case that we analyse. Moreover, there is a symbol $a \in V \cup \{\diamond\}$ whose number of appearances in the words of L' is not bounded by a constant.

We assume first that $a \neq \diamond$. It is known that

$$\{n \mid n = |w|_a,\ w \in L'\} = \{\alpha_0 + n_1\alpha_1 + \ldots + n_k\alpha_k \mid n_1, \ldots, n_k \in \mathbf{N}\}$$

for some natural number $k \geq 1$ and some constants $\alpha_0, \alpha_1, \ldots, \alpha_k$ such that at least one of the constants α_i with $i \geq 1$, say α_j, is not equal to 0 (see [7]). Thus, there exists the sequence of words $w_1, w_2, \ldots, w_i, \ldots$, such that $|w_i|_a = \alpha_0 + i\alpha_j$ (so the numbers of occurrences of a in these words form an arithmetic progression). If we denote $d = \gcd(\alpha_0, \alpha_j)$, then we obtain by Dirichlet's Theorem on arithmetic progressions the fact that there are infinitely many words from this sequence whose number of occurrences of symbols a is d multiplied by a prime number; it is quite easy to show that there also are infinitely many words from this sequence in which the symbol a occurs for d multiplied by a composite number times.

Now, we construct a language compatible with L as follows. In every word $w\diamond x$ from L, we replace the first $\diamond$ by a new symbol $e \notin V$ if and only if the number $N_{x,a}$ of occurrences of a in x divided by d (that is, $\left\lfloor \frac{N_{x,a}}{d} \right\rfloor$) is a prime number; otherwise, we replace this hole with another new symbol $d \notin V$. All the other holes in the words of L are replaced with d. Denote this language by L''.

[4] A language L has the constant growth property if when arranging the strings of the language in increasing order of length, two consecutive lengths do not differ by arbitrarily large amounts.

It is easy to show by the pumping lemma (pumping the words that contain an e leads to words containing e but its number of occurrences of a differs from d multiplied by a prime number) that L'' is not regular. Therefore, $L \notin \mathbf{REG}_{(\forall)}$. This proves the statement.

The case when $a = \diamond$ can be treated analogously, and we reach once more a contradiction.

This concludes our proof. □

Note that, this theorem provides a very simple procedure for deciding whether a given regular language (that contains partial words) is in $\mathbf{REG}_{(\forall)}$ or not. We simply check (by taking as input a DFA accepting that language) whether there are finitely many words that contain $\diamond$ or not. If the set of such words is finite, we accept the input and confirm that the given language is in $\mathbf{REG}_{(\forall)}$; otherwise, we reject the input. The time complexity of such an algorithm is $\mathcal{O}(nkt)$ provided that n is the number of states of the input automaton, k is the number of letters in its alphabet, and t is the number of transitions labelled with $\diamond$ from this automaton. Indeed, for each transition labelled with $\diamond$ ending in a state from which we can access a final state, we look whether there exists a cycle reachable after that transition is made and which is on a path towards a final state, or whether there exists a cycle from which that transition is accessible. Clearly, these checks can be done by simply traversing the graph of the automaton, thus, in time $\mathcal{O}(nk)$.

On $\mathbf{REG_{gsm}}$.
Recall that a deterministic generalised sequential machine (dgsm) is a 6-tuple $M = (Q, V, U, q_0, F, f)$ where Q is a set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, V and U are finite sets of symbols, namely, the set of input symbols and, respectively, the set of output symbols, and the transition-output function $f : Q \times V \to Q \times U^*$; this function is extended canonically to $Q \times V^*$. The finite transduction defined by M is the function $T_M : V^* \to U^*$, defined by: $T_M(v) = u$ if and only if $f(q_0, v) = (q, u)$ and $q \in F$.

To show $\mathbf{REG} \setminus \mathbf{REG_{full}} = \mathbf{REG_{gsm}}$ we proceed as follows. First, every language that is the image of a regular language through a finite transduction is regular. As the languages from $\mathbf{REG_{gsm}}$ contain at least one word that has a $\diamond$, we get that $\mathbf{REG_{gsm}} \subseteq \mathbf{REG} \setminus \mathbf{REG_{full}}$. On the other hand, let L be a regular language that has words that contain $\diamond$. Let L' be the language obtained by replacing, in the words of L, the *hole*-symbols with a symbol $a \notin \mathbf{alph}(L)$. Clearly, L is the image of L' through the finite transduction defined by a DGSM that maps every letter from $\mathbf{alph}(L) \setminus \{\diamond\}$ to itself, and a to $\diamond$. Thus, $\mathbf{REG} \setminus \mathbf{REG_{full}} \subseteq \mathbf{REG_{gsm}}$, and this concludes our proof.

Note that $\mathbf{REG_{gsm}}$ is incomparable with $\mathbf{REG}_{(\forall)}$, as the latter class contains $\mathbf{REG_{full}}$, while the former contains all the regular languages that have at least one word in which the $\diamond$ symbol appears.

Closure properties

We present here, in full details, a series of closure properties that highlight partly the differences between the defined classes of languages of partial words.

As a direct consequence of the fact that $\mathbf{REG}_{(\forall\sigma)}$ is equal to the class of regular languages **REG**, it follows that this class is closed under exactly the same operations that **REG** is closed under.

Proposition 1. $\mathbf{REG}_{(\forall\sigma)}$ *is closed under union, intersection, complementation, concatenation, Kleene star, morphisms, substitutions and inverse morphisms.*

The other classes of languages are closed under some operations and not closed under some other operation. More precisely, we have the following results.

Proposition 2. $\mathbf{REG}_{(\forall)}$ *is closed under union, intersection, intersection with arbitrary regular languages (regular languages that may contain partial words, as well).* $\mathbf{REG}_{(\forall)}$ *is not closed under complementation, concatenation, Kleene star, morphisms, substitutions and inverse morphisms.*

Proof. Union, intersection, intersection with regular sets: We obtain immediately that $\mathbf{REG}_{(\forall)}$ is closed under set-union and intersection, given the fact that the languages in this class can be all expressed as just a union between a regular language with no holes and a finite language formed of words containing the $\diamond$-symbol (see Theorem 11).

Concatenation and Kleene-star: Let R_1 and R_2 be two infinite regular languages of full words over V, and let F_1 and F_2 be two finite sets of full words over V. By Theorem 11, $R_1 \cup \{\diamond\}F_1$ and $R_2 \cup \{\diamond\}F_2$ are in $\mathbf{REG}_{(\forall)}$, whereas $(R_1 \cup \{\diamond\}F_1)(R_2 \cup \{\diamond\}F_2)$ and $(R_1 \cup \{\diamond\}F_1)^*$ are not in $\mathbf{REG}_{(\forall)}$.

Morphisms, substitutions, and inverse morphisms: We take the language $L = \{a^{2^n} \mid n \in \mathbf{N}\}$; clearly, this language is in $\mathbf{REG}_{(\forall)}$. Using the morphism that replaces a by $\diamond$, we get the language $L' = \{\diamond^{2^n} \mid n \in \mathbf{N}\}$ that is not in $\mathbf{REG}_{(\forall)}$ (it is compatible with the non-regular language $\{a^n b^n \mid n \in \mathbf{N}\}$). Thus we have the non-closure under morphisms and substitutions. Analogously, the inverse image of L under the morphism which maps $\diamond$ to a gives $L' \notin \mathbf{REG}_{(\forall)}$, again.

Complementation: We consider the language $V^* \cup \{\diamond\}$ seen as a subset of $(V \cup \{\diamond\})^*$. Obviously, this language is in $\mathbf{REG}_{(\forall)}$. But the complement of this language consists of all words w with $|w| \geq 2$ and $|w|_\diamond \geq 1$ and is not in $\mathbf{REG}_{(\forall)}$ by Theorem 11. Thus, $\mathbf{REG}_{(\forall)}$ is not closed under complementation. □

The following proposition is immediate.

Proposition 3. $\mathbf{REG}_{\mathbf{gsm}}$ *is closed under union, intersection, intersection with regular languages, concatenation, Kleene star.* $\mathbf{REG}_{(\forall)}$ *is not closed under complementation, morphisms, substitutions and inverse morphisms.*

Proof. Note that the empty language $L = \emptyset$ is contained in $\mathbf{REG} \setminus \mathbf{REG_{full}}$. The closure under union, concatenation, and Kleene star follows from the fact that all these operations, when applied to languages that contain words with $\diamond$ symbols, produce the languages with words that contain $\diamond$. The closure under intersection and intersection with regular languages follows from the fact that the intersection of two languages from $\mathbf{REG} \setminus \mathbf{REG_{full}}$, as well as the intersection of a language from this class with an arbitrary regular language, is either a language from the same class, or the empty language. The class is not closed under complementation, as the complement of $L = (V \cup \{hole\})^* \{\diamond\} (V \cup \{hole\})^*$ is V^*, and this language is not in $\mathbf{REG} \setminus \mathbf{REG_{full}}$. To see that $\mathbf{REG} \setminus \mathbf{REG_{full}}$ is not closed under morphisms or substitutions, take L as above and a morphism that maps $\diamond$ to a symbol of V; the image of L through this morphism is V^*. Similar arguments hold for the non-closure under inverse morphisms. □

We now move one to the classes from the upper part of our hierarchy.

Proposition 4. $\mathbf{REG}_{(\exists)}$ *is closed under union, concatenation, complementation and Kleene star.* $\mathbf{REG}_{(\exists)}$ *is not closed under intersection, intersection with regular languages, morphisms, substitutions and inverse morphisms.*

Proof. Morphisms, substitutions and inverse morphisms: We consider the language $\{ba^n \diamond^n \mid n \in \mathbf{N}\}$ which is in $\mathbf{REG}_{(\exists)}$, and using the morphism mapping $\diamond$ to b and leaving a and b in place, we obtain the language $\{ba^n b^n \mid n \in \mathbf{N}\}$, which is context-free and non-regular. Therefore, $\mathbf{REG}_{(\exists)}$ is not closed under morphism and substitution.

Similarly, we get the non-closure under inverse morphism (just take the morphism that maps $\diamond$ to b, a to a, and b to $\diamond$.

Union, concatenation and Kleene star: Let L_1 and L_2 be two languages from $\mathbf{REG}_{(\exists)}$. There exist the regular languages L_1' and L_2' of full words such that $L_1 \uparrow L_1'$ and $L_2 \uparrow L_2'$, respectively. Clearly, $L_1 \cup L_2 \uparrow L_1' \cup L_2' \in \mathbf{REG}$; thus, $L_1 \cup L_2$ is in $\mathbf{REG}_{(\exists)}$.

For concatenation and Kleene-star, we can give a similar proof.

Intersection, intersection with regular sets, and complementation: We consider the languages $L_1 = (\{a, b\}^2)^*$ and $L_2 = \{a^n b^n \mid n \in \mathbf{N}\} \cup \{\diamond^{2n} \mid n \in \mathbf{N}\}$ which are in both $\mathbf{REG}_{(\exists)}$ (both languages are compatible with $(\{a, b\}^2)^*$). Their intersection gives us the language $\{a^n b^n \mid n \in \mathbf{N}\}$ which is a context-free non-regular language of full words. Thus, the intersection is not in $\mathbf{REG}_{(\exists)}$. Since $(\{a, b\}^2)^*$ is, in fact, a (full-words) regular language we also have the non-closure of $\mathbf{REG}_{(\exists)}$ under intersection with regular languages of full words.

It is a well-known fact from set-theory ($X \cap Y = V^* \setminus ((V^* \setminus X) \cup V^* \setminus Y)$), when X and Y are languages over an alphabet V) that closure under complementation and union implies the closure under intersection. Now non-closure under complementation follows from the closure under union and the non-closure under intersection. □

Proposition 5. $\mathbf{REG}_{(\exists\sigma)}$ *is closed under Kleene star.* $\mathbf{REG}_{(\exists\sigma)}$ *is not closed under union, intersection, complementation, intersection with regular languages, concatenation, morphisms, substitutions and inverse morphisms.*

Proof. For *morphisms, substitutions, inverse morphisms, intersection, and intersection with regular sets* we repeat the proofs given for $\mathbf{REG}_{(\exists)}$ above using the same languages.

Union and concatenation: We take $L_1 = \{ba^n\diamond^n \mid n \in \mathbf{N},\ n \geq 1\}$ and $L_2 = \{ab^n\diamond^n \mid n \in \mathbf{N}, n \geq 1\}$ which are in $\mathbf{REG}_{(\exists\sigma)}$ (take the morphisms mapping a to a, b to b and $\diamond$ to a and b, respectively). Now let σ be a substitution with $\sigma(a) = a$ and $sigma(b) = b$. If $a \in sigma(\diamond)$, then

$$\sigma(L_1 \cup L_2) \cap \{a\}\{b\}^*\{a^*\} = \{ab^n a^n \mid n \in \mathbf{N}\}$$

which is not regular. By the closure properties of $\mathbf{REG}$, $\sigma(L_1 \cup L_2) \notin \mathbf{REG}$. Analogously, we can prove that $b \in \sigma(\diamond)$ implies $\sigma(L_1 \cup L_2) \notin \mathbf{REG}$. Moreover, when $\sigma(\diamond)$ does not contain a or b, a similar argument shows that $\sigma(L_1 \cup L_2) \notin \mathbf{REG}$. Thus $L_1 \cup L_2 \notin \mathbf{REG}_{(\exists\sigma)}$.

By analogous arguments, it follows that $L_1 \cdot L_2$ is not in $\mathbf{REG}_{(\exists\sigma)}$.

It is worth noting that L_1 and L_2 from the above proof are over the same alphabet.

Kleene-star: Since $\sigma(L)^* = \sigma(L^*)$ for any substitution, the class $\mathbf{REG}_{(\exists\sigma)}$ is closed under Kleene-star.

Complementation: Let $L' = \{(ab)^n\diamond b(ab)^n \mid n \in \mathbf{N}\} \cup \{(ab)^n a\diamond(ab)^n \mid n \in \mathbf{N}\}$. We remember that L' is not in $\mathbf{REG}_{(\exists\sigma)}$ (see the proof of Theorem 10). We set $L = \{a, b, \diamond\}^* \setminus L'$. We note that L' is in $\mathbf{REG}_{(\exists\sigma)}$. Indeed, $\{\diamond^n \mid n \in \mathbf{N}\} \subset L'$, so the substitution σ that leaves a and b unchanged and maps $\diamond$ to $\{a, b\}$ maps L to $\{a, b\}^*$. However, the complement of L (with respect to the set $\{a, b, \diamond\}^*$, as these are the letters that appear in L) is obviously L', which is not in $\mathbf{REG}_{(\exists\sigma)}$. So $\mathbf{REG}_{(\exists\sigma)}$ is not closed under complementation. □

Finally, let us investigate the closure properties of the $\mathbf{REG_{max}}$ class of languages.

Proposition 6. *The class* $\mathbf{REG_{max}}$ *is closed under union between languages over the same alphabet, concatenation between languages over the same alphabets, and Kleene-star.* $\mathbf{REG_{max}}$ *is not closed under general union, intersection, intersection with regular languages, complementation, general concatenation, morphisms, substitutions, and inverse morphisms.*

Proof. The closure under *Kleene-star* follows as in the preceding proof.

Morphisms, substitutions, and inverse morphisms: We consider the language $L = \cup_{n\in\mathbf{N}}\{a, b\}^n\{\diamond\}^n$. WE have that L is in $\mathbf{REG_{max}}$ (since the substitution mapping a and b to itself and $\diamond$ to $\{a, b\}$ gives the regular language $(\{a, b\}^2)^*$. The morphism σ mapping $\diamond$ to b and leaving a and b in place leads to $\sigma(L) = \cup_{n\in\mathbf{N}}\{a, b\}^n\{b\}^n$, which is a context-free non-regular language of full words, thus, not contained in $\mathbf{REG_{max}}$.

For the case of inverse morphism, we consider the morphism which maps a to a, b to b and c to $\diamond$.

Union and Concatenation: We take the languages $L = \{a^n \diamond^n \mid n \in \mathbf{N}\}$ and $L = \{ab\}$. Both these languages are in $\mathbf{REG_{max}}$, but neither their union nor their catenation is in this class.

However, note that the fact that $\mathbf{REG_{max}}$ is not closed under union and concatenation followed from the fact that the two languages that were used were over different alphabets (differently from the case of $\mathbf{REG}_{(\exists\sigma)}$, above). Let us assume that we have L_1 and L_2 languages from $\mathbf{REG_{max}}$ over the same alphabet; it is rather simple to prove that both $L_1 \cup L_2$ and $L_1 L_2$ are also in $\mathbf{REG_{max}}$.

Complementation: Let $L' = \{(ab)^n \diamond b(ab)^n \mid n \in \mathbf{N}\}$. From the proof of Theorem 10, we know that L' is not in $\mathbf{REG_{max}}$. Let $L = \{a, b, \diamond\}^* \setminus L'$. Note that L is in $\mathbf{REG_{max}}$ (since $\{\diamond^n \mid n \in \mathbf{N}\} \subset L$, so that substitution σ from the definition of $\mathbf{REG_{max}}$ maps L to $\{a, b\}^*$. However, its complement (with respect to the set $\{a, b, \diamond\}^*$) is obviously L', which is not in $\mathbf{REG_{max}}$.

Intersection and intersection with regular languages: Let $L_1 = \{a, b\}^* \cup \{w \mid w \in \{a, b, \diamond\}^*, |w|_a \geq |w|_\diamond\}$ and $L_2 = \{b\}\{a, \diamond\} * *\{\diamond\}\{a, \diamond\}^*$ (that is, L_2 contains words that start with b but follow with a word formed only from a and $\diamond$ symbols, and has at least a $\diamond$ symbol). It is rather clear that both of them are in $\mathbf{REG_{max}}$. In fact, the image of the first through the substitution σ from the definition of $\mathbf{REG_{max}}$ is $\{a, b\}^*$, while the second language is already regular. Their intersection is $L = \{b\}\{w \mid w \in \{a, \diamond\}^*, |w|_a \geq |w|_\diamond \geq 1\}$. The image of this language through σ is the set $L = \{b\}\{w \mid w \in \{a, b\}^*, |w|_a \geq |w|_b\}$, which is a context-free non-regular language. □

Finally, the following remark seems interesting to us.

Proposition 7. *There exist three languages L, L' and L'' with $L \sqsubset L' \sqsubset L''$ such that both L and L'' are from $\mathbf{REG_{max}}$ but $L' \notin \mathbf{REG_{max}}$.*

Proof. To see this consider the languages $L = \{\diamond^{2n+1} \mid n \in \mathbf{N}\}$, $L' = \{ba^n \diamond^n \mid n \in \mathbf{N}\}$, and $L'' = \cup_{n \in \mathbf{N}} \{a, b\}^{2n+1}$. It is straightforward that $L \sqsubset L' \sqsubset L''$ holds in this case, but, following the previous discussions, $L' \notin \mathbf{REG_{max}}$ while $L, L'' \in \mathbf{REG_{max}}$. □