Regular Languages of Partial Words[☆]

Jürgen Dassow[a], Florin Manea[b,c,*], Robert Mercaş[a,b]

[a] *Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, PSF 4120, D-39016 Magdeburg, Germany*
[b] *Christian-Albrechts-Universität zu Kiel, Institut für Informatik, Christian-Albrechts-Platz 4, D-24098 Kiel, Germany*
[c] *Faculty of Mathematics and Computer Science, University of Bucharest, Str. Academiei 14, RO-010014 Bucharest, Romania*

Abstract

We initiate a study of languages of partial words related to regular languages of full words. First, we investigate 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 by 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 introduce a hierarchy of classes of languages of partial words, by grouping together languages that can be connected in various strong ways to regular languages, and derive their closure properties with respect to several regular operations.

Keywords: Automata Theory, 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

[☆] This manuscript is an extended version of a paper that was presented at the conference *Computability in Europe 2012 - How the World Computes*, and published as [5].

[*] Corresponding author.

Email addresses: dassow@iws.cs.uni-magdeburg.de (Jürgen Dassow), flm@informatik.uni-kiel.de (Florin Manea), rgm@informatik.uni-kiel.de (Robert Mercaş)

a hydrogen bond with one complementary pyrimidine, T or C, respectively. But, sometimes, this process may go wrong, allowing G-T bonds. Starting from this situation that occurs in nature, 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 [2] suggested the usage of partial words as a suitable mathematical model for studies in DNA-computing and bioinformatics. 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. These were investigated already since the 1970s [6]. Going back to the initial example, such partial words offer a possibility to represent and work with DNA sequences, defined by a 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, partial words may be used to represent data which were corrupted either by white noise or other external factors, as well as data on which we only have incomplete or insufficient information, but still needs to be somehow processed.

In the last decade a lot of combinatorial and algorithmic properties of partial words have been investigated (see the survey [3], and the references therein). Surprisingly, so far, not much effort was directed towards the study of classes of languages of partial words (or sets of partial words that have common features); to this end, we recall [4, 7, 8]. From the previous attempts of defining languages of partial words, [8] is of high relevance to our work. 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 by 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 [8] regarded classes of full-words-languages defined by applying successively the 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 [12]).

In this work, we aim to establish a stronger connection between the two notions mentioned above: partial words, on one side, and regular languages, on the other side.

The foremost connection that we identified was to (non-trivially) represent every regular language of full words 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). This approach leads to a result that seems interesting to us: for some regular languages of full words, 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. However, it may also be the case that 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. In all cases, 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. Alongside these descriptional complexity results, a series of decidability and computational complexity results regarding the possibility of representing a regular language of full words as the image of a language of partial words through a $\diamond$-substitution are derived.

Motivated by the above initial results, that connect in a meaningful way languages of partial words to regular languages of full words, and by the theoretical interest of studying systematically such languages, it seemed a natural step to define a series of other 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. We analyse these classes from a language theoretic point of view, show that they form a hierarchy, and establish their closure properties with respect to regular operations on languages.

Although these are just first steps in investigating languages of partial words, the fact that we obtain results in very diverse areas of theoretical computer science (descriptional complexity, language theory, or algorithmics and computational complexity) seems to show the fact that a rich theory can

be developed with respect to this concept.

2. Preliminaries

This section consists of a series of basic definitions and notation. Further definitions regarding finite automata and regular languages can be found in [12, Chapter 2, volume 1], while partial words are surveyed in [3].

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 $\diamond \notin V$. 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 u is denoted by $|u|_a$. The *empty (partial) word* is the sequence of length zero and is denoted by λ. Furthermore, by $u[i]$ denote the symbol at position i in u, where $0 < i \leq |u|$. The catenation of two (partial) words u and v is defined as the (partial) word uv.

For an alphabet V, that does not contain $\diamond$, we denote by V^* (respectively, $(V \cup \{\diamond\})^*$) the set of words (respectively, partial words) over V and by V^+ (respectively, $(V \cup \{\diamond\})^+$) the set of non-empty words (respectively, non-empty partial words) over V. Recall that V^* (respectively, $(V \cup \{\diamond\})^*$) is the free monoid generated by V (respectively, $V \cup \{\diamond\}$), under the operation of catenation of words; the empty word λ is the unit element of these monoids.

A language L of full words over an alphabet V, that does not contain $\diamond$, is a subset of V^*; a language of partial words L over V 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 exposition, 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 $\{a, b\}$. 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 $v \sqsubset w$. For example the words $\diamond a\diamond$ and $b\diamond\diamond$ are compatible, and they are both contained in the partial word $ba\diamond$. 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 by $L \sqsubset L'$, 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 by $L \uparrow L'$, 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 $v' \uparrow v$. For the languages $L = \{aaaa, b\diamond, b\diamond b\diamond\}$ and $L' = \{\diamond a, \diamond a\diamond a\}$, we have that $L \uparrow L'$, but only $L' \sqsubset \{a^*\}$.

A substitution is a mapping $h : W^* \to 2^{U^*}$ with $h(xy) = h(x)h(y)$, for $x, y \in W^*$, and $h(\lambda) = \{\lambda\}$; h is completely defined by the values $h(a)$ for all $a \in W$. A $\diamond$-substitution over an alphabet V, such that $\diamond \notin V$, is a substitution $h : (V \cup \{\diamond\})^* \to 2^{V^*}$ with $h(a) = \{a\}$, for $a \in V$, and $h(\diamond) \subseteq V$. That is, the hole can be replaced by any symbol of a subset of the alphabet V. A morphism is a particular type of a substitution for which $h(a)$ contains exactly one element for all $a \in W$. For simplicity, by identifying singletons with their elements, a morphism is seen as a function $h : W^* \to U^*$ with $h(xy) = h(x)h(y)$ for $x, y \in W^*$.

In this paper, DFA stands for deterministic finite automaton and NFA for non-deterministic finite automaton. A finite automaton M is defined by a quintuple (Q, W, δ, q_0, F), where Q is the set of states out of which q_0 is the initial one, W the input alphabet (which may contain or not the $\diamond$ symbol), $\delta : Q \times W \to 2^Q$ the transition function and $F \subseteq Q$ the set of final states; if $|\delta(q, a)| = 1$ for all $q \in Q$ and $a \in W$ then the automaton is said to be deterministic. The language accepted by the finite automaton M is denoted by $L(M)$. Note that this is a mechanism used in the recognition of regular languages. The set of all the regular languages is denoted by $\mathbf{REG}$; this class includes both the class of regular languages of full words $\mathbf{REG_{full}}$ as well as the class of regular languages of partial words, i.e., regular languages containing at least one word that has a $\diamond$ symbol.

Note that whenever we address computational complexity issues, we use the classical RAM with logarithmic word size model [1].

3. Definability by Substitutions

Let us begin our investigation by presenting several results regarding the way regular languages of full words can be expressed as the image of a language of partial words through a substitution. In this section we assume that the alphabets V and V' that appear in some of the definitions and statements of our results do not contain the $\diamond$ symbol.

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 $w \in L'$.*

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

Since $L' \uparrow L$, we conclude that $\sigma(L') = \sigma(L)$. □

It is worth noting that the language L' previously defined in the statement of 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. *Let $L \subseteq V^*$ be a regular language of full words and 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 $L' = \sigma(L'')$, where every word in the language L'' has exactly one hole. Moreover, L' and L'' are regular languages (of full words and, respectively, partial words) and, provided that L is given by a finite automaton accepting it, we can algorithmically construct finite automata accepting L' and L''.*

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

We start by noting that if a word $w \in L$ belongs to $\sigma(L_0)$ for some partial-words-language L_0 whose elements contain exactly one hole each, then there exist two full words x and y such that $x \diamond y \in L_0$, $w = xay \in \sigma(x \diamond y)$ for some $a \in V'$, and $\{x\}V'\{y\} \subseteq L$.

Now let $M = (Q, V, \delta, q_0, F)$ 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, where $\delta(q_0, x) = q$ and $\delta(\delta(q, V'), y) \subseteq F$. Basically, R_q is the set of words for which there exists some compatible partial word $x\diamond y$ with exactly one hole, such that x labels a path from q_0 to q in M and all words from $\{x\}V'\{y\}$ are in L. Therefore, $R_q = \{x \mid x \in V^*, \delta(q_0, x) = q\}V'\{y \mid y \in V^*, \delta(q', y) \in F$ for all $q' \in \delta(q, V')\}$. It follows that R_q is regular and an automaton accepting it 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 $q' \in \delta(q, V')\}$.

We 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 $L' = \sigma(L'')$. To complete the proof of the statement, we only have to show that L' is maximal.

If there exists $L_1 \subseteq L$ 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 $w \in \sigma(x \diamond y) = \{x\}V'\{y\} \subseteq \sigma(L_2) = L_1 \subseteq L$. Now, if we take $q = \delta(q_0, x)$, we get that $y \in \{y \mid y \in V^*, \delta(q', y) \in F \text{ for all } q' \in \delta(q, V')\}$. It follows that $w \in R_q = \{x \mid x \in V^*, \delta(q_0, x) = q\}V'\{y \mid y \in V^*, \delta(q', y) \in F \text{ for all } q' \in \delta(q, V')\}$. As $L' = \cup_{q \in Q} R_q$ we obtain that $w \in L'$. Therefore, $L_1 \subseteq L'$, and the maximality of L' with respect to set inclusion follows. □

We note that the sets R_q with $q \in Q$ are not a partition of L, as they are not necessarily mutually disjoint and their union might be a proper subset of L. Furthermore, we could also have the situation that $L' = L'' = \emptyset$; that is, the maximal set with respect to set inclusion is empty, as when $V' \setminus \mathbf{alph}(L) \neq \emptyset$. For example, consider $\mathbf{alph}(L) = \{a, b\}$ and $V' = \{a, c\}$, and see that it is impossible to have a nonempty language of partial words L'' with $\sigma(L'') \subseteq L$ and $L'' \neq L$, as $c \in \mathbf{alph}(\sigma(L''))$, but $c \notin \mathbf{alph}(L)$.

Also, the remark in the beginning of the previous proof can be slightly generalised. Namely, if a word w belongs to $\sigma(L_0)$ for a language L_0 whose elements contain at least one hole each, then there exist two partial words x' and y' such that $x' \diamond y' \in L_0$, $w = xay \in \sigma(x' \diamond y')$ for some $x \in \sigma(x')$, $a \in V'$, $y \in \sigma(y')$, and $\{x\}V'\{y\} \subseteq L$.

Further, the following remark is a brief analysis regarding the complexity of constructing an NFA accepting L' defined in the proof of Lemma 2, given a deterministic automaton accepting the regular language L.

Remark 1. *We denote by n the number of states $|Q|$ of the deterministic finite automaton accepting L. 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 deterministic 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 a DFA for $\{w' \mid w' \in V^*, \delta(q', w') \in F \text{ for all } q' \in \delta(q, V')\}$ (that can be done in time $\mathcal{O}(|V|n^{|V'|})$ by intersecting the $|V'|$ languages $R_{q,a} = \{w' \mid w' \in V^*, \delta(q', w') \in F \text{ for } q' = \delta(q, a)\}$ for $a \in V'$, accepted respectively by finite deterministic automata with n states each). Now we just have to construct a nondeterministic finite*

automaton accepting the union of the sets R_q (respectively, H_q) and we get a nondeterministic 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 $\mathcal{O}(n^{|V'|+1})$).

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

Definition 1. *Let $L \subseteq V^*$ be a language of full words and σ be a $\diamond$-substitution with $\sigma(\diamond) = V' \subseteq V$. We say that L is σ-definable 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-definable *by L' if $L = \sigma(L')$ and $L' \subseteq (V' \cup \{\diamond\})^*\{\diamond\}(V' \cup \{\diamond\})^*$.*

Generally, we say that a language of full words L is σ-definable if it is σ-definable by some language of partial words L'.

Obviously, for any regular language L of full words 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-definable 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 the rest of this section. We begin by characterising the essentially definable languages.

We assume now that the regular language L of full words over V is essentially σ-definable for some $\diamond$-substitution σ over V. Then, according to Definition 1, $\sigma(L') = L$ for some appropriate language L' such that any word of L' contains at least one 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 have 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. *The following hold:*

(i) Given a regular language L of full words over V and a $\diamond$-substitution σ over V, it is decidable whether L is essentially σ-definable.

(ii) Given a regular language L of full words over V, we 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. □

We note, as a completion of Theorem 2, that we can decide similarly the following properties:

- Given a regular language L of full words 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 of full words 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, if L is given by a DFA, 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').

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

Theorem 3. *Given a regular language L of full words over 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 $\{w \mid w \in L_1, |w|_\diamond \geq 1\} \sqsubset \{w \mid w \in L_\diamond, |w|_\diamond \geq 1\}$,

(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 of full words $L \subseteq V^*$, a regular language of partial words $L' \subseteq (V \cup \{\diamond\})^*$, 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 we can describe in a more succinct way a regular language via a language of partial words and a substitution that defines it? Moreover, can we decide algorithmically whether for a given regular language L of full words 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 $\min_{DFA}(L)$ 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 the 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. *The following hold:*

(i) For every regular language L of full words we have

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

(ii) There exist regular languages L of full words such that

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

Proof. Let us start by first proving statement (i).

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 an NFA accepting L by replacing each transition labelled with $\diamond$ by transitions

labelled with all letters of $\sigma(\diamond)$ and deleting the error state of the DFA, when existing, and all transitions towards it. Thus $\min^{\diamond}_{DFA}(L) \geq \min_{NFA}(L)$. Moreover, $\min_{DFA}(L) \geq \min^{\diamond}_{DFA}(L)$, since we 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 by $\diamond$. Clearly, each DFA accepting L' can be transformed in a DFA accepting L and vice versa. As L' is a regular language of partial words for which there exists the $\diamond$-substitution σ such that $\sigma(L') = L$, we get that $\min^{\diamond}_{DFA}(L) \leq \min_{DFA}(L')$. However, $\min_{DFA}(L') = \min_{DFA}(L)$, as L' is nothing but a copy of L with all the a symbols replaced by $\diamond$ symbols. Consequently, $\min^{\diamond}_{DFA}(L) \leq \min_{DFA}(L)$.

Let us now consider the second statement and take 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_1 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 an example we were looking for in our statement; however, it is somewhat 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 $L = L_1L_2$. We 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, $\min_{NFA}(L) = 13$ (by similar reasons as in the case of the DFAs). This concludes our proof. □

By the previous result we can see that, for certain substitutions σ, min-

imal 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, we 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. We start by proving the following technical lemma.

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

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

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

Proof. The proof uses a rather standard technique to show the exponential blow-up in the number of states of the DFA with respect to the number of states of the NFA, see, e.g., the proof of [11, Theorem 1]. However, compared to the cited proof, the overall discussion and most of the details are more complicated in our case.

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 $R \in 2^Q$.

To 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.

First, we 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 $t \leq n-2$. We take $R' = \{q'_3, \ldots, q'_k, n-1\}$, where $q'_\ell = q_\ell - q_2$, for $3 \leq \ell \leq k$, and $x = a^2 b^{t-1} a^{q_1-1}$. 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 our 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 include $\{1, 2, n\}$ 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 $\delta'(\{q_1 - 1, \ldots, q_k - 1, n-1, n\}, a)$ and we already know that $\{q_1 - 1, \ldots, q_k - 1, n-1, n\}$ is reachable.

Finally, we show that the states $R \in 2^Q$ that include $\{1, n\}$ 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 include $\{1, 2, n\}$; 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''.

We take P and R two different states of M''. We 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 $\delta(1, ay) = \delta(2, y) \subseteq \delta(n, ay) \subseteq \delta(R, ay)$; also, if $x = by$ we have $\delta(P, by) = \delta(R, by) \cup \delta(1, by) = \delta(R, by)$, as $\delta(1, by) = \delta(1, y) \subseteq \delta(n, by) \subseteq \delta(R, by)$. 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 minimisation algorithm on M'' we obtain a minimal DFA accepting $L(M)$ that has exactly $2^{n-1} + 2^{n-2}$ states. □

The previous lemma can be used to show that there are regular languages for which there is an exponential blow-up from $\min^{\diamond}_{DFA}$ to $\min_{DFA}$, respectively, from $\min_{NFA}$ to $\min^{\diamond}_{DFA}$.

Theorem 5. *Let n be a natural number with $n \geq 3$.*

(i) There exists a regular language L of full words such that $\min^{\diamond}_{DFA}(L) \leq n + 1$ and $\min_{DFA}(L) = 2^n - 2^{n-2}$.

(ii) There exists a regular language L' of full words such that $\min_{NFA}(L') \leq 2n + 1$ and $\min^{\diamond}_{DFA}(L') \geq 2^n - 2^{n-2}$.

Proof. Let us first investigate problem (i), the existence of a regular language L of full words such that $\min^{\diamond}_{DFA}(L) \leq n + 1$ and $\min_{DFA} = 2^n - 2^{n-2}$.

We take the language $L = L(M)$ defined in Lemma 3. Further, we take the DFA $M_{\diamond} = (Q', \{\diamond, a, b\}, \delta', 1, F')$ with

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

We note that M can be obtained from $M_{\diamond}$ replacing the transitions labelled with $\diamond$ by transitions labelled with a and b, and deleting the error-state $n+1$

and all the transitions related to it. See that L is defined by the language $L(M_\diamond)$ of partial words and the $\diamond$-substitution σ that maps $\diamond$ to $\{a, b\}$.

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

In the remainder of this proof we show (ii), namely, there exists a regular language L' of full words such that $\min_{NFA}(L') \leq 2n + 1$ and $\min^{\diamond}_{DFA}(L') \geq 2^n - 2^{n-2}$. In the following, L is the language defined in the first part of the proof.

Let L_0 be the language $f(L)$, where f is a morphism mapping a to a' and b to b'. The language L_0 is accepted by an NFA with n states that can be easily obtained from M (by replacing the transitions labelled with s by transitions labelled with s' for $s \in \{a, b\}$). Let L' be the language defined as the catenation of $L \cup \{c\}$ and L_0, that is $L' = (L \cup \{c\})L_0$; we note that if L' contains a symbol a or b (respectively, a' or b') then it must contain at least $n - 1$ symbols from $\{a, b\}$ (respectively, $\{a', b'\}$). As $L \cup \{c\}$ is accepted by an NFA with $n + 1$ states (in which, compared to M there exists a new final state reachable from q_0 by a transition labelled by c) and L_0 is accepted by an NFA with n states, we obtain that $\min_{NFA}(L') \leq 2n + 1$. We show that $\min^{\diamond}_{DFA}(L') \geq 2^n - 2^{n-2}$.

For simplicity, we 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 $\diamond$-substitution that maps $\diamond$ to a subset of V such that $\sigma(L_\diamond) = L'$. Let $M_\diamond$ be the minimal DFA accepting $L_\diamond$ with the transition function $\delta_\diamond$ and the initial state q_0.

We start our analysis with the case when $c \in \sigma(\diamond)$. First, assume that we have a transition $\delta_\diamond(q_1, \diamond) = q_2$, $q_1 \neq q_0$, 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 exists a string $z \diamond x \in L_\diamond$ with $z \neq \lambda$; thus, in $L' = \sigma(L_\diamond)$ there is the string $z'cx'$ with $z' \neq \lambda$, a contradiction. Therefore, 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, we 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_0, the only transitions that leave q_1 and reach another state are labelled with letters from $\{a', b'\}$; 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 $\{a', b'\}$. We 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_0. 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; thus, it has at least $2^n - 2^{n-2}$ states and, in this case as well, $M_\diamond$ has at least $2^n - 2^{n-2}$ states.

We continue with the case when $c \notin \sigma(\diamond)$. We 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_0. 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 included in $M_\diamond$, so this DFA has at least $2^n - 2^{n-2}$ states.

With this we conclude 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

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

and

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

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

Remark 2. *Given a DFA accepting a regular language L of full words we can construct algorithmically a DFA with $\min^{\diamond}_{DFA}(L)$ states, accepting a regular language L' of partial words, and a $\diamond$-substitution σ over $\mathbf{alph}(L)$, such that L is σ-definable 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 [10], 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 $v \in V^\ell$ 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–to-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 $\sigma(L(M)) \neq V^\ell$ (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. □

It is well known that deciding whether a DFA and an NFA accept different languages is PSPACE-complete (see, e.g., [12, Chapter 2, volume 1]). Theorem 6 provides a simple way to show the following weaker hardness result.

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

Proof. The problem from Theorem 6 can be reduced to this problem in polynomial time, as each pair consisting in a DFA M accepting a language of 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. □

4. Languages of partial words

While the results of the last section study the possibility and efficiency of defining a regular language of full words 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) of full words seem worth investigating. Just like before, in this section we assume that the alphabet V that appears in some definitions and statements does not contain the $\diamond$ symbol.

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 of full words.

Definition 2. *Let L be a language of partial words over V, with $\diamond \notin V$.*

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 $V \neq \emptyset$ and $\sigma(L) \in \mathbf{REG_{full}}$, where σ is the unique $\diamond$-substitution over V with $\sigma(\diamond) = V$, or if $V = \emptyset$ (i.e., $L \subseteq \{\diamond\}^*$) and $L \in \mathbf{REG}$.*
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,* $\mathbf{max}$*-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*, with* $\diamond \notin V$*.*

4. *We say that* L *is* $(\exists)$*-regular if there exists a regular language* L' *of full words such that* $L \uparrow L'$*.*

5. *We say that* L *is* $(\forall)$*-regular if for every language* L' *of full words with* $L \uparrow L'$*, the language* L' *is regular.*

The class of all the $(\exists)$*-regular languages is denoted by* $\mathbf{REG}_{(\exists)}$*, while that of* $(\forall)$*-regular languages by* $\mathbf{REG}_{(\forall)}$*.*

According to the definitions from [8], the $(\exists)$-regular languages are those whose restoration contains at least one regular language of full words, while the $(\forall)$-regular languages are those whose restoration contains only regular languages of full words.

We start this direction of investigation with the following observation.

Theorem 7. *For every non-empty alphabet* V *with* $\diamond \notin V$ *there exists 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 any* $\diamond$*-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 (that is, recursively enumerable but not a recursive language) 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 the above language 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 $\mathbf{REG}$ 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 $\mathbf{REG}_{(\forall\sigma)}$, we can take the special substitution that replaces $\diamond$ by a symbol that does not occur in $\mathbf{alph}(L)$ and obtain a regular language of full words; 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 a non-empty alphabet V, the language L defined in the proof of this theorem 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 over V with $\sigma(\diamond) = V$. 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 $\sigma(L)$, an automaton accepting $\sigma(L_0)$ can be constructed.*

Proof. First, we observe 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 $V \neq \emptyset$, as in the statement of the lemma. Furthermore, let σ be the single $\diamond$-substitution over V with $\sigma(\diamond) = V$. By definition, we have that $\sigma(L)$ is a regular language of full words. Therefore, there exists a deterministic finite automaton $A = (Q, V, \delta, q_0, F)$ 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 $A' = (2^Q, V \cup \{\diamond\}, \delta', \{q_0\}, 2^F)$ where δ' works as follows:

- $\delta'(S, a) = \delta(S, a) = \cup_{q \in S}\{\delta(q, a)\}$, for $S \subseteq Q$ and $a \in V$.
- $\delta'(S, \diamond) = \delta(S, V) = \cup_{q \in S}(\cup_{a \in V}\{\delta(q, a)\})$, 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 from the definition. 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 $\sigma(w) \subseteq \sigma(L)$. A first consequence is that $\sigma(L(A')) \subseteq \sigma(L)$. Moreover, any language L' such that $\sigma(L') = \sigma(L)$ is included in $L(A')$, as for each $w \in L'$ we have $\sigma(w) \subseteq \sigma(L)$, so $w \in L(A')$. In conclusion, we obtain that L_0, the union of all the languages L' with $\sigma(L') = \sigma(L)$, is in fact exactly $L(A')$.

As L_0 is accepted by a finite automaton, thus being regular, it is clear that L_0 is in $\mathbf{REG_{max}}$. □

Moreover, we note that any language from $\mathbf{REG_{max}}$ defined over the empty set, has in fact all words made out only of holes, and is, according to the definition, a regular language.

Remark 3. *If L from the previous theorem is contained in $\{\diamond\}^*$, 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 $\{n \mid n \in \mathbf{N}$, there exists $w \in L, |w| = n\} = \{n \mid n \in \mathbf{N}$, 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 $L_0^a = \{w' \mid w' \in \{\diamond, a\}^*$, there exists $w \in L, |w| = |w'|\}$, which is clearly a regular language, and plays the role of L_0 from the above theorem. In the case when another letter b is chosen instead of a, so the image of $\diamond$ becomes $\{b\}$, we get a different language L_0^b. However, each two languages L_0^x and L_0^y, obtained as above for different choices of the letters x and y, are isomorphic.Thus, 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 defined on.*

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 contained in $\mathbf{REG_{max}}$.*

Furthermore, we have that:

Remark 4. *As every language of partial words from the classes we defined is compatible with at least one regular language of full words, it follows easily that all these languages have the constant growth property*[1]*. Therefore, 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.,* $\{a^n \diamond a^n \diamond a^n \mid n \in \mathbf{N}\}$*).*

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 deduced immediately from the definition. We show now that each of the previous inclusions is strict.

We 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 $\diamond$-substitution σ with $\sigma(\diamond) = \{a\}$, 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, we 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 $\{(ab)^{2n+1} \mid n \in \mathbf{N}\}$. □

Following the definitions, all the languages in $\mathbf{REG}_{(\forall)}$ are in $\mathbf{REG}_{(\forall\sigma)} = \mathbf{REG}$; however, not all the languages in $\mathbf{REG}$ are in $\mathbf{REG}_{(\forall)}$. The following statement characterises exactly the regular languages that are in $\mathbf{REG}_{(\forall)}$.

Theorem 11. *Let* $L \in \mathbf{REG}$*. Then* $L \in \mathbf{REG}_{(\forall)}$ *if and only if the set* $\{w \mid |w|_\diamond \geq 1, w \in L\}$ *is finite.*

Proof. We assume that L is a regular partial-words-language over an alphabet V.

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

[1] 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.

We now show the other implication. Assume that L contains an infinite number of partial words that have at least one $\diamond$ 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 (due to the closure of regular languages under quotient operation). Also, L' is infinite according to the case that we analyse. Moreover, there exists 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 [12]). Thus, there exists a 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. Given the description of the set we also have that there 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 by another new symbol $f \notin V$. All the other holes are replaced by f. Denote this language by L''.

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.

Since the case when $a = \diamond$ can be treated analogously, and we reach once more a contradiction, we conclude our proof. □

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. 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 $\mathcal{O}(nk)$.

Theorem 11 has also the following consequence.

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

In many previous works (surveyed in [3]), partial words were defined by replacing specific symbols of full words by $\diamond$, in a procedure that resembles the puncturing of [8]. Similarly, in [9], 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.

Recall that a DGSM is a 6-tuple $M = (Q, V, U, f, q_0, F)$ 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 $q \in F$.

Accordingly, we can define a new class of partial-words-languages, denoted by $\mathbf{REG}_{\mathbf{gsm}}$, 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 $\mathbf{REG}_{\mathbf{gsm}}$, 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'. Moreover, recall that we denote by $\mathbf{REG}_{\mathbf{full}}$ the set of all the regular languages of full words (languages whose words contain no $\diamond$ symbol).

Theorem 13. $\mathbf{REG} \setminus \mathbf{REG}_{\mathbf{full}} = \mathbf{REG}_{\mathbf{gsm}}$.

Proof. Every language that is the image of a regular language through a finite transduction is regular. As the languages from $\mathbf{REG}_{\mathbf{gsm}}$ contain at least one word that has a $\diamond$, we get that $\mathbf{REG}_{\mathbf{gsm}} \subseteq \mathbf{REG} \setminus \mathbf{REG}_{\mathbf{full}}$.

On the other hand, let L be a regular language that has words that contain $\diamond$ and let L' be the language obtained by replacing, in the words of L, the $\diamond$-symbols by 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 we conclude. □

We 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 (thus also languages having an infinite number of words with holes).

By Theorems 8, 10, 11, 12, and 13 we get the following hierarchies:

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

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

5. Closure properties

We end this work with 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 $\mathbf{REG}$, it follows that this class is closed under exactly the same operations that $\mathbf{REG}$ 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 of the operations and not closed under other. More precisely, we have the following results.

Proposition 2. $\mathbf{REG}_{(\forall)}$ *is closed under union, intersection, and intersection with arbitrary regular languages (including 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, and 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 $L = \{a^{2n} \mid n \in \mathbf{N}\}$; clearly, this language is in $\mathbf{REG}_{(\forall)}$. Using the morphism that replaces a by $\diamond$, we get the language $L' = \{\diamond^{2n} \mid n \in \mathbf{N}\}$ 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. It follows that $\mathbf{REG}_{(\forall)}$ is not closed under complementation. □

The following proposition is immediate.

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

Proof. Union, concatenation, and Kleene star: 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 languages with words that contain $\diamond$.

Intersection and intersection with regular sets: The closure under intersection with regular languages follows from the fact that the intersection with the empty set, will give us the empty set. For the intersection of two languages from $\mathbf{REG} \setminus \mathbf{REG}_{\mathbf{full}}$, we consider the languages $L = \{a\diamond\}$ and $L' = \{b\diamond\}$. Obviously their intersection is the empty set not contained in $\mathbf{REG}_{\mathbf{gsm}}$, while both languages are from this class.

Complementation: The complement of $L = (V \cup \{\diamond\})^*\{\diamond\}(V \cup \{\diamond\})^*$ is V^*, and this language is not in $\mathbf{REG} \setminus \mathbf{REG_{full}}$.

Morphisms, substitutions, and inverse morphisms: To see that $\mathbf{REG_{gsm}}$ is not closed under morphisms or substitutions, we take $L = \{\diamond\}V^*$ and a morphism that maps $\diamond$ to a symbol of $a \in V$; the image of L through this morphism is aV^*. Similar arguments hold for the non-closure under inverse morphisms. □

We now move on 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. 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 $L_1 \cup L_2$ is compatible with $L_1' \cup L_2'$, and since $L_1' \cup L_2' \in \mathbf{REG}$, we have that $L_1 \cup L_2$ is in $\mathbf{REG}_{(\exists)}$. For concatenation and Kleene-star, the proofs follow in a similar manner.

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 $\{ba^nb^n \mid n \in \mathbf{N}\}$, 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 by just taking the morphism that maps $\diamond$ to b, a to a, and b to $\diamond$.

Intersection, intersection with regular sets, and complementation: We consider the languages $L_1 = (\{a, b\}^2)^*$ and $L_2 = \{a^nb^n \mid n \in \mathbf{N}\} \cup \{\diamond^{2n} \mid n \in \mathbf{N}\}$ which are both in $\mathbf{REG}_{(\exists)}$ (both languages are compatible with $(\{a, b\}^2)^*$). Their intersection gives us the language $\{a^nb^n \mid n \in \mathbf{N}\}$ 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. Kleene-star: Since $\sigma(L)^* = \sigma(L^*)$ for any substitution, the class $\mathbf{REG}_{(\exists\sigma)}$ is closed under Kleene-star.

Morphisms, substitutions, inverse morphisms, intersection, and intersection with regular sets: The previous proofs given for $\mathbf{REG}_{(\exists)}$ apply in this case as well.

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 or b, respectively). Now let σ be a substitution with $\sigma(a) = a$ and $\sigma(b) = b$. If $a \in \sigma(\diamond)$, then

$$\sigma(L_1 \cup L_2) \cap \{a\}\{b\}^*\{a\}^* = \{ab^n a^n \mid n \in \mathbf{N}, n \geq 1\}$$

which is not regular. 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.

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 recall that L' is not in $\mathbf{REG}_{(\exists\sigma)}$ (see the proof of Theorem 10). Take $L = \{a, b, \diamond\}^* \setminus L'$ and note that L is in $\mathbf{REG}_{(\exists\sigma)}$. Indeed, $\{\diamond^n \mid n \in \mathbf{N}\} \subset L$, so the $\diamond$-substitution σ that maps $\diamond$ to $\{a, b\}$ takes in fact 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, we 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 al-*

phabet, 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. Kleene-star: This property follows as in the preceding proof.

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, we note that the non-closure of $\mathbf{REG_{max}}$ under union and concatenation followed from the fact that the two used languages were defined over different alphabets (differently from the case of $\mathbf{REG}_{(\exists\sigma)}$, above). Assume now that we have L_1 and L_2 languages from $\mathbf{REG_{max}}$ over the same alphabet; it is rather simple to prove that in this case 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 $\mathbf{REG_{max}}$ and, since $\{\diamond^n \mid n \in \mathbf{N}\} \subset L$, the 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 $\mathbf{REG_{max}}$.

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 themselves, and $\diamond$ to $\{a, b\}$ gives the regular language $(\{a, b\}^2)^*$). The morphism γ mapping $\diamond$ to b ,and leaving a and b in place leads to $\gamma(L) = \cup_{n \in \mathbf{N}} \{a, b\}^n \{b\}^n$, 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 $\diamond$.

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 followed by only a and $\diamond$ symbols, and have at least one $\diamond$ 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 $L = \{b\}\{w \mid w \in \{a, b\}^+, |w|_a \geq |w|_b \geq 1\}$, which is a context-free non-regular language. □

The following observation, is yet another interesting fact in our opinion.

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, we 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 $L, L'' \in \mathbf{REG_{max}}$. □

We end this section by summarising in the following table all the closure properties of the previously defined classes. 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
$\mathbf{REG_{gsm}}$	y	n	n	n	y	y	n	n	n

6. Conclusions

In this paper we established a series of connections between the classical model of the class of regular languages of full words and some new classes consisting in languages of partial words. We showed that, although trivial representations of regular languages of full words as languages of words containing "don't-care" symbols can be devised, there are some more insightful possibilities to do this, leading to meaningful descriptional complexity and language theoretic results.

One possibility we considered was to describe a regular language of full words by a DFA accepting a language of partial words that can be mapped to the initial language through a substitution that only rewrites the $\diamond$ symbol. This can be seen as allowing a description of the regular language by a finite automaton with limited non-determinism. Our main results state that

such a representation can be sometimes exponentially more succinct than the usual description of a regular language by its minimal DFA. However, this approach can also lead to descriptions of a regular language exponentially more complex than a minimal NFA accepting it. An interesting question is to check the existence of a regular language of full words that can be represented by a partial-words-language whose minimal DFA is exponentially smaller than the minimal DFA accepting the initial language, but also exponentially larger than the minimal NFA accepting that same language.

It seems interesting to us to continue this line of research by investigating which possible gaps can actually occur between the size of a minimal DFA (or NFA) accepting a regular language of full words and the DFA accepting a language of partial words that can be mapped to the initial language through a $\diamond$-substitution. Also, a central topic in descriptional complexity was the study of the state complexity of languages defined by applying regular operations to regular languages (see [12, Chapter 2, volume 1]). Accordingly, it seems interesting to investigate in which conditions the succinctness of representing regular languages as regular languages of partial words is preserved when regular operations are applied. To this end, the proof of Theorem 5 shows that the succinctness cannot be always preserved in the case of catenation of regular languages, but we expect that the case of other operations is different.

We continued our investigation by defining a series of classes of partial-words-languages, each of which was placed in a particular strong relation with the class of regular languages. While some classes were defined starting from the possibility to represent regular languages of full words as the image of languages of partial words through $\diamond$-substitutions, others were defined starting from one of the central concepts regarding partial words, namely that of compatible partial words. We investigated these classes from a language theoretic point of view, showed that they form a hierarchy, and established their closure properties with respect to regular operations.

Besides defining other (more meaningful) classes of languages of partial words that can be connected to regular languages, it seems interesting, at least from a theoretical point of view, a continuation of this investigation on other classes of the Chomsky-hierarchy. For instance, it is well known that in the case of context-free languages the class of languages accepted by deterministic push-down automata is different from that of the languages accepted by non-deterministic push-down automata. Is it the case that when we allow the limited non-determinism introduced by the presence of $\diamond$-symbols we

obtain something in the middle? Further work in this direction might also consist in finding languages from a certain class that have an exponentially more succinct description whenever partial non-determinism is allowed compared to their typical deterministic representations, just as in the case of regular languages.

Finally, as both regular languages and partial words are strongly motivated by practical applications, it is our hope that some of the results reported in this paper may have a practical relevance as well.

Acknowledgements

The work of Florin Manea is supported by the *DFG* grant 596676. The work of Robert Mercaş was supported by the *Alexander von Humboldt Foundation* and *DFG* grant 582014.

The authors thank the anonymous reviewers for there valuable comments that improved in a great measure the presentation of this paper.

[1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[2] J. Berstel and L. Boasson. Partial words and a theorem of Fine and Wilf. *Theoretical Computer Science*, 218:135–141, 1999.

[3] F. Blanchet-Sadri. *Algorithmic Combinatorics on Partial Words*. Chapman & Hall/CRC Press, 2008.

[4] Francine Blanchet-Sadri. Codes, orderings, and partial words. *Theor. Comput. Sci.*, 329(1-3):177–202, 2004.

[5] J. Dassow, F. Manea, and R. Mercaş. Connecting partial words and regular languages. In S. B. Cooper, A. Dawar, and B. Löwe, editors, *How the World Computes - Turing Centenary Conference and 8th Conference on Computability in Europe, CiE 2012, Cambridge, UK, June 18-23, 2012. Proceedings*, volume 7318 of *Lecture Notes in Computer Science*, pages 151–161. Springer, 2012.

[6] 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.

[7] Peter Leupold. Languages of partial words - how to obtain them and what properties they have. *Grammars*, 7:179–192, 2004.

[8] G. Lischke. Restoration of punctured languages and similarity of languages. *Mathematical Logic Quarterly*, 52(1):20–28, 2006.

[9] F. Manea and R. Mercaş. Freeness of partial words. *Theoretical Computer Science*, 389(1-2):265–277, 2007.

[10] F. Manea and C. Tiseanu. Hard counting problems for partial words. In *LATA*, volume 6031 of *Lecture Notes in Computer Science*, pages 426–438. Springer-Verlag, 2010.

[11] 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.

[12] G. Rozenberg and A. Salomaa. *Handbook of Formal Languages*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.