On the state complexity of closures and interiors of regular languages with subwords and superwords

P. Karandikar[a,c,1], M. Niewerth[b,2], Ph. Schnoebelen[c,3]

[a] *Chennai Mathematical Institute*
[b] *University of Bayreuth*
[c] *LSV, ENS Cachan, CNRS*

Abstract

The downward and upward closures of a regular language L are obtained by collecting all the subwords and superwords of its elements, respectively. The downward and upward interiors of L are obtained dually by collecting words having all their subwords and superwords in L, respectively. We provide lower and upper bounds on the size of the smallest automata recognizing these closures and interiors. We also consider the computational complexity of decision problems for closures of regular languages.

Keywords: Finite automata and regular languages; Subwords and superwords; State complexity; Combined operations; Closures and interiors of regular languages.

1. Introduction

State complexity is a standard measure of the descriptive complexity of regular languages. The most common state complexity problems ask, given a regularity-preserving operation f on languages, to bound the size of an automaton recognizing f(L) when L is recognized by an n-state automaton. We refer to [26, 46] for a survey of the main known results in the area.

In this article, we consider language operations based on subwords. Recall that a (scattered) subword of some word x is a word obtained from x by removing any number of letters at arbitrary positions in x, see formal definitions in Section 2. Symmetrically, a superword is obtained by inserting letters at arbitrary positions. Subwords and superwords occur in many areas of computer science, from searching in texts and databases [4] to the theory of codes [29], computational linguistics [40], and DNA computing [33].

For a language $L \subseteq \Sigma^*$, we write $\downarrow L$ for the set of all its subwords and $\uparrow L$ for the set of all its superwords (in Σ^*) and call them the *downward closure* and

---

upward closure of L, respectively. Dual to closures are *interiors*. The *upward interior* and *downward interior* of L, denoted $\Updownarrow L$ and $\Downarrow L$, are the largest upward-closed and downward-closed sets included in L. It has been known since [22] that $\downarrow L$ and $\uparrow L$ are regular for any L. Then $\Downarrow L$ and $\Updownarrow L$ are regular too by duality, as expressed in the following equalities:

$$\Downarrow L = \Sigma^* \smallsetminus \uparrow(\Sigma^* \smallsetminus L)\,, \qquad\qquad \Updownarrow L = \Sigma^* \smallsetminus \downarrow(\Sigma^* \smallsetminus L)\,. \qquad (1)$$

Computing closures and interiors has several applications in computer-aided reasoning [32] and program verification. Computing closures is an essential ingredient in the verification of safety properties of channel systems – see [1, 20] – while computing interiors is required for the verification of their game-theoretical properties [5]. More generally, the regularity of upward and downward closures make them good overapproximations of more complex languages – see [3, 21, 47] – and interiors can be used as regular underapproximations.

Recently Gruber *et al.* explicitly raised the issue of the state complexity of downward and upward closures of regular languages [18, 19] (less explicit precursors exist, for example, [7]). Given an n-state automaton A that recognizes L, automata $A^{\downarrow}$ and $A^{\uparrow}$ that recognize $\downarrow L$ and, $\uparrow L$ respectively can be obtained by simply adding extra transitions to A. However, when A is a deterministic automaton (a DFA), the resulting $A^{\downarrow}$ and $A^{\uparrow}$ are in general not deterministic (are NFAs), and their determinization may entail an exponential blowup. With n denoting the number of states of A, Gruber *et al.* proved a $2^{\Omega(\sqrt{n}\log n)}$ lower bound on the number of states of any DFA recognizing $\downarrow L$ or $\uparrow L$ [19], to be compared with the $2^n - 1$ upper bound that comes from the simple closure+determinization method.

Okhotin improved on these results by showing an improved $2^{\frac{n}{2}-2}$ lower bound for $\downarrow L$. He also established the exact state complexity for $\uparrow L$ by proving a $2^{n-2} + 1$ upper bound and showing that this is tight [39].

All the above lower bounds assume an unbounded alphabet, and Okhotin showed that his $2^{n-2} + 1$ state complexity for $\uparrow L$ requires $n - 2$ distinct letters. He then considered the case of languages over a *fixed alphabet* and, in the 3-letter case, he demonstrated exponential $2^{\sqrt{2n+30}-6}$ and $\frac{1}{5}4^{\sqrt{n/2}}n^{-\frac{3}{4}}$ lower bounds for $\downarrow L$ and $\uparrow L$ respectively [39]. In the 2-letter case, Héam had previously proved an $\Omega(r^{\sqrt{n}})$ lower bound for $\uparrow L$, here with $r = (\frac{1+\sqrt{5}}{2})^{\frac{1}{\sqrt{2}}}$ [23]. Regarding $\downarrow L$, the question whether its state complexity is exponential even when $|\Sigma| = 2$ was left open (note that the one-letter case is trivial).

The state complexity of interiors has not yet been considered in the literature. When working with DFAs, complementation is essentially free so that computing interiors reduces to computing closures, thanks to duality. However, when working with NFAs, the simple complement+closure+complement method comes with a quite large 2^{2^n} upper-bound on the number of states of an NFA that recognizes $\Updownarrow L$ or $\Downarrow L$ – it actually yields DFAs – and one would like to improve on this, or to prove a matching lower bound. As we explain in Section 5.3, this is related to the state complexity of closures when working

2

with alternating automata (AFAs), a question recently raised in [25].

Our contribution. Regarding closures with DFAs, we prove in Section 3 a tight 2^{n-1} state complexity for downward closure and show that its tightness requires unbounded alphabets. In Section 4 we prove an exponential lower bound on both $\downarrow L$ and $\uparrow L$ in the case of a two-letter alphabet, answering the open question raised above. Regarding interiors on NFAs, we show in Section 5 doubly-exponential lower bounds for downward and upward interiors, assuming an unbounded alphabet. We also provide improved upper bounds, lower than the naive 2^{2^n} but still doubly exponential. Table 1 shows a summary of the results. Finally, Section 6 proves lower bounds on unambiguous automata for the witness languages used in Section 3, and Section 7 considers the computational complexity of some basic decision problems for sets of subwords or superwords described by automata.

Table 1: A summary of the results on state complexity for closures and interiors, where $\psi(n)$ ($\leqslant 2^{2^n}$) is the nth Dedekind's number[5].

Operation	Unbounded alphabet	Fixed alphabet
$\uparrow L$ (DFA to DFA)	$=2^{n-2}+1$ for $\lvert\Sigma\rvert\geqslant n-2$	$2^{\Omega(n^{1/2})}$ for $\lvert\Sigma\rvert=2$
$\downarrow L$ (DFA to DFA)	$=2^{n-1}$ for $\lvert\Sigma\rvert\geqslant n-1$	$2^{\Omega(n^{1/3})}$ for $\lvert\Sigma\rvert=2$
$\uparrow L$ (AFA to AFA)	$\geqslant 2^{\left\lfloor\frac{n-3}{2}\right\rfloor}$ and $<2^n$ for $\lvert\Sigma\rvert$ in $2^{\Omega(n)}$	$\vdots$
$\downarrow L$ (AFA to AFA)	$\geqslant 2^{\left\lfloor\frac{n-4}{3}\right\rfloor}$ and $\leqslant 2^n$ for $\lvert\Sigma\rvert$ in $2^{\Omega(n)}$	(unknown)
$\circlearrowright L$ (NFA to NFA)	$>2^{2^{\left\lfloor\frac{n-4}{3}\right\rfloor}}$ and $\leqslant\psi(n)$ for $\lvert\Sigma\rvert$ in $2^{\Omega(n)}$	$\vdots$
$\circlearrowleft L$ (NFA to NFA)	$\geqslant 2^{2^{\left\lfloor\frac{n-3}{2}\right\rfloor}}$ and $\leqslant\psi(n)$ for $\lvert\Sigma\rvert$ in $2^{\Omega(n)}$	$\vdots$

Related work. We already mentioned previous work on the closure of regular languages: it is also possible to compute closures by subwords or superwords for larger classes like context-free languages or Petri net languages, see [3, 21, 47] and the references therein for applications and some results on descriptive complexity.

Interiors are duals of closures and should not be confused with the inverse operations considered in [6], or the shuffle residuals from [29]. Duals of regularity-preserving operations have the form "*complement–operation–complement*" and thus can be seen as special cases of the *combined operations* studied in [44]

---

[5]Recall that the nth Dedekind number $\psi(n)$ is the number of antichains in the lattice of subsets of an n-element set, ordered by inclusion [34]. Kahn [30, Corollary 1.4] shows

$$\binom{n}{\lfloor n/2\rfloor}\leqslant\log_2\psi(n)\leqslant\left(1+\frac{2\log(n+1)}{n}\right)\binom{n}{\lfloor n/2\rfloor}.$$

and following papers. Dual operations occur naturally in algorithmic or logical contexts but have not yet been considered widely from a state-complexity perspective: we are only aware of [8] studying the dual of $L \mapsto \Sigma^* \cdot L$.

2. Basic notions and results

Subwords. We assume familiarity with regular languages and the automata that recognize them. We write $x, y, u, v, \ldots$ to denote words over a finite alphabet $\Sigma = \{a, b, \ldots\}$, with $|x|$ denoting the length of a word x. For $1 \leqslant i \leqslant |x|$, we let $x[i]$ denote the i-th letter of x. The empty word is denoted ε and concatenation is denoted multiplicatively.

We say that a word x is a *subword* of y, written $x \sqsubseteq y$, when y can be written in the form $y = y_0 \, x_1 \, y_1 \cdots y_{m-1} \, x_m \, y_m$ for some factors such that $x = x_1 \cdots x_m$. For example, $\varepsilon \sqsubseteq a\,b \sqsubseteq a\,c\,b\,a$. Equivalently, $x \sqsubseteq y$ when there are positions $0 < p_1 < p_2 < \cdots < p_\ell \leqslant |y|$ such that $x[i] = y[p_i]$ for all $1 \leqslant i \leqslant \ell = |x|$. When $x \sqsubseteq y$ we also say that x *embeds* in y, or that y is a *superword* of x.

Closures. For a language $L \subseteq \Sigma^*$, we define $\uparrow L \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid \exists y \in L : y \sqsubseteq x\}$ and $\downarrow L \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid \exists y \in L : x \sqsubseteq y\}$, and call them the upward and downward closures of L respectively.[6]

The Kuratowski closure axioms are satisfied:

$$\downarrow \varnothing = \varnothing \,, \quad L \subseteq \downarrow L = \downarrow\downarrow L \,, \quad \downarrow\Big(\bigcup_i L_i\Big) = \bigcup_i \downarrow L_i \,, \quad \downarrow\Big(\bigcap_i \downarrow L_i\Big) = \bigcap_i \downarrow L_i \,,$$

and similarly for upward closures. We say that a language $L \subseteq \Sigma^*$ is *downward-closed* if $L = \downarrow L$ and that a language is *upward-closed* if $L = \uparrow L$. Note that L is downward-closed if, and only if, its complement $\Sigma^* \smallsetminus L$ is upward-closed but the complement of $\downarrow L$ is *not* $\uparrow L$.

Regularity. The upward-closure $\uparrow x$ of a word $x = a_1 \cdots a_\ell$ is a regular language given by the regular expression $\Sigma^* a_1 \Sigma^* \cdots a_\ell \Sigma^*$. Since, by Higman's Lemma [24], any language L only contains finitely many elements that are minimal for the subword ordering, one deduces that $\uparrow L$ is regular for any $L \subseteq \Sigma^*$, a result also known as Haines's Theorem [22]. Then $\downarrow L$, being the complement of an upward-closed language, is regular too. In fact, upward-closed languages are simple star-free languages. They correspond exactly to the level $\frac{1}{2}$ of Straubing's hierarchy [41], and coincide with the *shuffle ideals*, that is, the languages that satisfy $L = L \shuffle \Sigma^*$ [9]. Downward-closed languages coincide with *strictly piecewise-testable* languages [43].

Effective construction of a finite-state automaton recognizing $\downarrow L$ or $\uparrow L$ is easy when L is regular (see Section 3), is possible when L is context-free [14, 35], and is not possible in general since this would allow deciding the emptiness of L.

---

[6]Formally $\uparrow L$ should more precisely be denoted $\uparrow_\Sigma L$ since it depends on the underlying alphabet but in the rest of this article Σ will always be clear from the context.

Interiors. The *upward interior* of a language L over Σ is $\circlearrowright L \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid \uparrow x \subseteq L\}$. Its *downward interior* is $\circlearrowleft L \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid \downarrow x \subseteq L\}$. Alternative characterizations are possible, for example, by noting that $\circlearrowright L$ or $\circlearrowleft L$ are the largest upward-closed or downward closed languages, respectively, included in L, or by using the duality equations from page 2. These equations show that $\circlearrowleft L$ and $\circlearrowright L$ are regular for any L. They also show how, when L is regular, one may compute automata recognizing the interiors of L by combining complementations and closures.

State complexity. When considering a finite automaton $A = (\Sigma, Q, \delta, I, F)$, we usually write n for $|Q|$, k for $|\Sigma|$, and $L(A)$ for the language recognized by A. In the context of a fixed automaton A we often write $q \xrightarrow{a} q'$ to mean $q' \in \delta(q, a)$. We also write $q \xrightarrow{w} q'$ where $w \in \Sigma^*$ to denote the existence of a w-labeled path from q to q' in the graph of A. In Section 6 we consider unambiguous automata (UFAs): recall that an NFA A is *unambiguous* if every word $w \in L(A)$ has a single accepting run [12].

For a regular language L, $n_{\mathrm{D}}(L)$, $n_{\mathrm{N}}(L)$ and $n_{\mathrm{U}}(L)$ denote the minimum number of states of a DFA, an NFA, and a UFA, respectively, that accepts L. Note that NFAs are allowed to have multiple initial states, and DFAs need not be complete. Since any DFA is unambiguous, one obviously has $n_{\mathrm{N}}(L) \leqslant n_{\mathrm{U}}(L) \leqslant n_{\mathrm{D}}(L)$ for any regular language. In cases where $n_{\mathrm{N}}(L) = n_{\mathrm{D}}(L)$ we may use $n_{\mathrm{N\&D}}(L)$ to denote the common value.

An application of the fooling set technique. The following lemma is a well-known tool for proving lower bounds on $n_{\mathrm{N}}(L)$.

Lemma 2.1 (Extended fooling set technique, [17]) *Let L be a regular language. Suppose that there exists a set of pairs of words $S = \{(x_i, y_i)\}_{1 \leqslant i \leqslant n}$, called a* fooling set, *such that $x_i\, y_i \in L$ for all $i = 1, \ldots, n$, and such that for all $j \neq i$, at least one of $x_i\, y_j$ and $x_j\, y_i$ is not in L. Then $n_N(L) \geqslant n$.*

PROOF. Let $A = (\Sigma, Q, \delta, I, F)$ be an NFA recognizing L. For each $i = 1, \ldots, n$, $x_i y_i \in L$, so A has an accepting run of the form $s_i \xrightarrow{x_i} q_i \xrightarrow{y_i} f_i$, starting at some initial state $s_i \in I$, ending at some accepting state $f_i \in F$, and visiting some intermediary state $q_i \in Q$. Observe that if $q_i = q_j$ for $i \neq j$ then A has accepting runs for both $x_i y_j$ and $x_j y_i$, which contradicts the assumption. Hence the states $q_1, q_2, \ldots, q_n$ are all distinct and $|Q| \geqslant n$. $\qquad\square$

In preparation for Section 3, let us apply the fooling set technique to the following languages, where Σ is an arbitrary finite alphabet:

$$U_\Sigma \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid \forall a \in \Sigma : \exists i : x[i] = a\}, \qquad U'_\Sigma \stackrel{\text{def}}{=} \Sigma \cdot U_\Sigma,$$

$$V_\Sigma \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid \forall i \neq j : x[i] \neq x[j]\}.$$

Note that U_Σ consists of all words where every letter in Σ appears at least once while V_Σ consists of all words where no letter appears twice. A word in U'_Σ consists of an arbitrary letter from Σ followed by a word in U_Σ. Note that U_Σ and U'_Σ are upward-closed while V_Σ is downward-closed.

Lemma 2.2 $n_{\text{N\&D}}(U_\Sigma) = n_{\text{N\&D}}(V_\Sigma) = 2^{|\Sigma|}$ *and, if* Σ *is not empty,* $n_{\text{N\&D}}(U'_\Sigma) = 2^{|\Sigma|} + 1$.

PROOF. Let us start with the lower bounds for $n_{\text{N}}(U_\Sigma)$ and $n_{\text{N}}(V_\Sigma)$: With any $\Gamma \subseteq \Sigma$, we associate two words x_Γ and $x_{\neg\Gamma}$, where x_Γ has exactly one occurrence of each letter from Γ, and where $x_{\neg\Gamma}$ has exactly one occurrence of each letter not in Γ. Then $x_\Gamma x_{\neg\Gamma}$ belongs to U_Σ and V_Σ, while for any $\Delta \neq \Gamma$ one of $x_\Gamma x_{\neg\Delta}$ and $x_\Delta x_{\neg\Gamma}$ does not belong to U_Σ and one does not belong to V_Σ. Thus for U_Σ or V_Σ we may use the same fooling set $S = \{(x_\Gamma, x_{\neg\Gamma})\}_{\Gamma \subseteq \Sigma}$. By Lemma 2.1, we conclude that $n_{\text{N}}(U_\Sigma) \geqslant 2^{|\Sigma|}$ and $n_{\text{N}}(V_\Sigma) \geqslant 2^{|\Sigma|}$.

For U'_Σ we pick an arbitrary letter $a \in \Sigma$ and let our fooling set be $S = \{(a\, x_\Gamma, x_{\neg\Gamma})\}_{\Gamma \subseteq \Sigma} \cup \{(\varepsilon, a\, x_\Sigma)\}$. As above $a\, x_\Gamma x_{\neg\Gamma}$ belongs to U'_Σ while, for any $\Delta \neq \Gamma$, one of $a\, x_\Gamma x_{\neg\Delta}$ and $a\, x_\Delta x_{\neg\Gamma}$ does not belong to U'_Σ. Furthermore $\varepsilon \cdot a\, x_\Sigma$ belongs to U'_Σ, while $\varepsilon \cdot x_{\neg\Gamma}$ does not belong to U'_Σ for any $\Gamma \subseteq \Sigma$. By Lemma 2.1, we conclude that $n_{\text{N}}(U'_\Sigma) \geqslant 2^{|\Sigma|} + 1$.

Proving the upper bounds is a well-known exercise in automata theory. One designs DFAs using the powerset $2^\Sigma = \{\Gamma, \Gamma', \ldots\}$ as the set of states, that is, automata with $2^{|\Sigma|}$ states. With rules of the form $\Gamma \xrightarrow{a} \Gamma \cup \{a\}$, these states record the set of letters read so far, starting from $\varnothing$ as initial state. In the DFA for U_Σ, one accepts when all letters have been seen. In the DFA for V_Σ, all states are accepting but it is forbidden to read a letter that has already been seen: there are no transitions $\Gamma \xrightarrow{a} \Gamma \cup \{a\}$ when $a \in \Gamma$. A DFA recognizing U'_Σ is obtained from the DFA for U_Σ by adding a new initial state from which one will read a first letter before continuing as for U_Σ. □

In the following, we use $\Sigma_k \overset{\text{def}}{=} \{a_1, \ldots, a_k\}$ to denote a k-letter alphabet, and write U_k and V_k instead of U_{Σ_k} and V_{Σ_k}.

3. State complexity of closures

Let $L \subseteq \Sigma^*$ be a regular language recognized by an NFA A. One may obtain NFAs recognizing the upward and downward closures $\uparrow L$ and $\downarrow L$ by simply adding transitions to A, without increasing its number of states. More precisely, an NFA $A^\uparrow$ recognizing $\uparrow L$ is obtained from A by adding self-loops $q \xrightarrow{a} q$ for every state q of A and every letter $a \in \Sigma$. Similarly, an NFA $A^\downarrow$ recognizing $\downarrow L$ is obtained from A by adding a silent transition $p \xrightarrow{\varepsilon} q$, also called an "$\varepsilon$-transition", for every original transition $p \xrightarrow{a} q$ in A.

If L is recognized by a DFA or an NFA A and we want a DFA recognizing $\uparrow L$ or $\downarrow L$, we can start with the NFA $A^\uparrow$ or $A^\downarrow$ defined above and transform it into a DFA using the powerset construction. This shows that if L is recognized by an n-state DFA, then both its upward and downward closures are recognized by DFAs with at most $2^n - 1$ states.

It is possible to provide tighter upper bounds by taking advantage of specific features of $A^\uparrow$ and $A^\downarrow$. The next two propositions give tight upper bounds for upward and downward closure, respectively.

Proposition 3.1 (State complexity of upward closure, after [39]) *1. If $L \subseteq \Sigma^*$ is a regular language with $n_N(L) = n$ then $n_D(\uparrow L) \leqslant 2^{n-2} + 1$.*
2. Furthermore, for any $n > 1$ there exists a regular language L_n with $n_N(L_n) = n_D(L_n) = n$ and $n_D(\uparrow L_n) = n_U(\uparrow L_n) = 2^{n-2} + 1$.

PROOF. 1. Let $A = (\Sigma, Q, \delta, I, F)$ be an n-state NFA recognizing $L = L(A)$. We can assume $I \cap F = \varnothing$ and $|I \cup F| \geqslant 2$ otherwise L contains ε or is empty, resulting in a trivial $\uparrow L$ with $n_D(\uparrow L) = 1$.

Since $A^{\uparrow}$ has loops on all its states and for any letter, applying the powerset construction yields a DFA where $P \xrightarrow{a} P'$ implies $P \subseteq P'$, hence any state P reachable from I includes I. Furthermore, if P is accepting, that is, $P \cap F \neq \varnothing$, and $P \xrightarrow{a} P'$, then P' is accepting too, hence all accepting states recognize exactly Σ^* and are equivalent. Then there can be at most $2^{|Q \smallsetminus (I \cup F)|}$ states in the powerset automaton that are both reachable and not accepting. To this we add 1 for the accepting states since they are all equivalent and will be merged in the minimal DFA. Finally $n_D(\uparrow L) \leqslant 2^{n-2} + 1$ since $|I \cup F|$ is at least 2.

2. To show that $2^{n-2} + 1$ states may be necessary, we first consider the case where $n = 2$: taking $L_2 = \{a\}$ over a 1-letter alphabet witnesses both $n_D(L_n) = n = 2$ and $n_D(\uparrow L_n) = 2^{n-2} + 1 = 2$. Further, $n_U(\uparrow L_2) = 2$ since clearly $n_N(\uparrow L_2) > 1$.

In the general case where $n > 2$ we define $L_n \stackrel{\text{def}}{=} E_{n-2}$ where

$$E_k \stackrel{\text{def}}{=} \{a\,a \mid a \in \Sigma_k\} = \{a_1\,a_1, \ldots, a_k\,a_k\}\,.$$

In other words, L_n contains all words consisting of two identical letters from $\Sigma = \Sigma_{n-2}$. The minimal DFA recognizing L_n has n states, see Figure 1. Now $\uparrow L_n = \bigcup_{a \in \Sigma} \Sigma^* \cdot a \cdot \Sigma^* \cdot a \cdot \Sigma^*$, that is, $\uparrow L_n$ contains all words in Σ^* where *some letter reappears*. Thus $\uparrow L_n$ is the complement of the language we called V_{n-2} above.
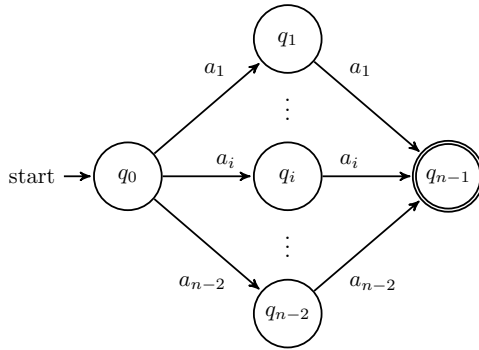

Figure 1: n-state DFA recognizing $L_n = E_{n-2} = \{a_1\,a_1, a_2\,a_2, \ldots, a_{n-2}\,a_{n-2}\}$.

The simplest way to recognize $\uparrow L_n$ is via a DFA that records, in its states, the set of letters previously read and accepts when one reappears. This will use $2^{|\Sigma|} + 1 = 2^{n-2} + 1$ states, one for each subset of previously read letters,

to which one adds a single accepting state. This DFA is minimal: given any two words x and y that reach distinct states, one finds a z such that $x\,z \in\, \uparrow L_n$ and $y\,z \notin\, \uparrow L_n$ or vice versa. We conclude that $n_D(\uparrow L_n) = 2^{n-2}+1$ and deduce $n_N(L_n) = n$ (that is, we rule out $n_N(L_n) < n$) from the first part of the lemma.

We refer to Proposition 6.5 in Section 6 for a proof that recognizing $\uparrow L_n$ requires $2^{n-2}+1$ states *even for UFAs*. $\qquad\square$

Proposition 3.2 (State complexity of downward closure) *1. If $L \subseteq \Sigma^*$ is recognized by an n-state NFA with a single initial state then $n_D(\downarrow L) \leqslant 2^{n-1}$. 2. Furthermore, for any $n > 1$ there exists a language L'_n with $n_N(L'_n) = n_D(L'_n) = n$ and $n_D(\downarrow L'_n) = n_U(\downarrow L'_n) = 2^{n-1}$.*

PROOF. 1. Assume that L is recognized by $A = (\Sigma, Q, \delta, \{q_{\text{init}}\}, F)$, an NFA where all states are reachable from q_{init}, the single initial state. From A one derives an NFA $A^{\downarrow}$ recognizing $\downarrow L$ by adding ε-transitions $q \xrightarrow{\varepsilon} q'$ for all pairs of states q, q' such that q' is reachable from q. In particular, $A^{\downarrow}$ contains transitions $q_{\text{init}} \xrightarrow{\varepsilon} q$ for all states $q \in Q$, and the language accepted from q is a subset of the language accepted from q_{init}. Hence, in the deterministic powerset automaton obtained from $A^{\downarrow}$, all states $P \subseteq Q$ that contain q_{init} are equivalent. This powerset automaton also has up to $2^{n-1} - 1$ nonempty states that do not contain q_{init}. Thus $1 + 2^{n-1} - 1$ bounds the number of non-equivalent nonempty states in the powerset automaton obtained from $A^{\downarrow}$, showing $n_D(\downarrow L) \leqslant 2^{n-1}$.

2. To show that 2^{n-1} states are sometimes necessary, we assume $n > 1$ and let $L'_n \overset{\text{def}}{=} D_{n-1}$ where

$$D_k \overset{\text{def}}{=} \{x \in \Sigma_k^+ \mid \forall i > 1 : x[i] \neq x[1]\} = \bigcup_{a \in \Sigma_k} a \cdot \big(\Sigma_k \smallsetminus a\big)^* .$$

Thus L'_n contains all words in Σ_{n-1}^+ where *the first letter does not reappear*. The minimal DFA recognizing L'_n has n states, see Figure 2. Every NFA for L'_n has at least n states, as shown by considering the following fooling set:

$$S = \left\{ \begin{array}{l} (\varepsilon, a_1 a_2 a_3 \cdots a_{n-1}),\ (a_1, a_2 a_3 a_4 \cdots a_{n-1}),\ (a_2, a_1 a_3 a_4 \cdots a_{n-1}), \\ (a_3, a_1 a_2 a_4 \cdots a_{n-1}),\ \ldots,\ (a_{n-1}, a_1 a_2 a_3 \cdots a_{n-2}) \end{array} \right\} .$$

We now turn to $\downarrow L'_n = \{x \mid \exists a \in \Sigma_{n-1} : \forall i > 1 : x[i] \neq a\}$. That is, $\downarrow L'_n$ contains all words x such that the first suffix $x[2..]$ does not use all letters. Equivalently $x \in\, \downarrow L'_n$ if, and only if, $x \in L'_n$ or x does not use all letters, that is, $\downarrow L'_n$ is the union of L'_n and the complement of the language we called U_{n-1} above.

To show that $n_D(L'_n) = 2^{n-1}$, we start with a DFA A that reads a first letter and then starts recording which letters have been encountered after the first one, in a manner similar to the construction of a DFA for U'_Σ in the proof of Lemma 2.2. All the states of A are accepting but, in states of the form $\Sigma \smallsetminus a$, the DFA has no a-labelled transitions, hence Σ is not a reachable state. Finally A has $1 + 2^{|\Sigma|} - 1 = 2^{n-1}$ states: the initial state reading the first letter, and
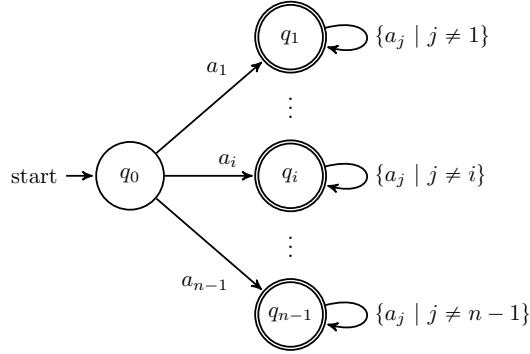
Figure 2: n-state DFA recognizing $L'_n = D_{n-1} = \bigcup_{a \in \Sigma} a \cdot (\Sigma - \{a\})^*$ with $|\Sigma| = n - 1$.

one state for each strict subset of Σ. This DFA is minimal: as in the previous proof, one checks that no two states in A are equivalent. Alternatively, one can refer to Section 6 where we prove – see Proposition 6.4 – that recognizing $\downarrow L'_n$ requires 2^{n-1} states *even for UFAs*. □

Remark 3.3 *The condition of a single initial state in Proposition 3.2 cannot be lifted. It is possible to have $n_D(\downarrow L) = n_U(\downarrow L) = 2^n - 1$ when $n_N(L) = n$. For example, the downward-closed language $L = \Sigma_n^* \smallsetminus U_n$ of all words that do not use all letters is recognized by an n-state NFA (see Figure 3) but its minimal DFA has $2^n - 1$ states. In fact, any UFA recognizing L has at least $2^n - 1$ states (see Proposition 6.3 in Section 6).*
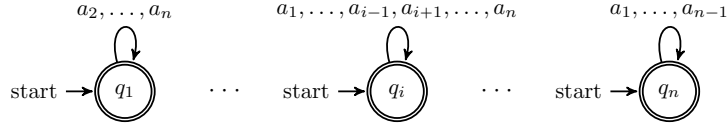

Figure 3: n-state NFA recognizing $\Sigma_n^* \smallsetminus U_n$.

The language families $(L_n)_{n \in \mathbb{N}}$ and $(L'_n)_{n \in \mathbb{N}}$ used to prove that the upper bounds given in Propositions 3.1 and 3.2 are tight use alphabets with a size linear in n.

It is known that the size of the alphabets matter for the state complexity of closure operations. The automata witnessing tightness in Figures 1 and 2 use the smallest possible alphabets. Okhotin showed that the $2^{n-2} + 1$ state complexity for $\uparrow L$ cannot be achieved with an alphabet of a size smaller than $n-2$, see [39, Lemma 4.4]. We now prove a similar result for downward closures:

Lemma 3.4 *For $n > 2$, let $L \subseteq \Sigma^*$ be a regular language accepted by an n-state NFA with a single initial state. If $|\Sigma| < n - 1$ then $n_D(\downarrow L) < 2^{n-1}$.*

9

PROOF. We assume that L is accepted by $A = (\Sigma, Q, \delta, \{q_{\text{init}}\}, F)$ with $|Q| = n$, that $n_{\text{D}}(\downarrow L) = 2^{n-1}$ and deduce that $|\Sigma| \geqslant n - 1$.

We write $Q = \{q_{\text{init}}, q_1, \ldots, q_{n-1}\}$ to denote the states of A. As we saw in the proof of the first part of Proposition 3.2, the powerset automaton built from $A^{\downarrow}$ can only have 2^{n-1} non-equivalent reachable states if all non-empty subsets of $Q \smallsetminus q_{\text{init}}$ are reachable. Since $A^{\downarrow}$ has ε-transitions doubling all transitions from A, it is possible to construct the powerset automaton with Q as its initial state. Then all edges $P \xrightarrow{a} P'$ in the powerset automaton satisfy $P \supseteq P'$. As a consequence, if $P \xrightarrow{x} P'$ for some $x \in \Sigma^*$ then in particular one can pick x with $|x| \leqslant |P \smallsetminus P'|$.

Since every non-empty subset of $Q \smallsetminus q_{\text{init}}$ is reachable from Q there is, for every $i = 1, \ldots, n-1$, some x_i of length 1 or 2 such that $Q \xrightarrow{x_i} Q \smallsetminus q_{\text{init}}, q_i$ (here $Q \smallsetminus q, q'$ is shorthand for $Q \smallsetminus \{q, q'\}$). For a given i, there are three
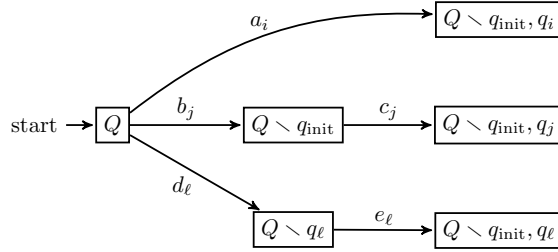

Figure 4: A part of the powerset automaton of $A^{\downarrow}$

possible cases (see Figure 4): $x_i = a_i$ is a single letter (type 1), or x_i is some $b_i c_i$ with $Q \xrightarrow{b_i} Q \smallsetminus q_{\text{init}} \xrightarrow{c_i} Q \smallsetminus q_{\text{init}}, q_i$ (type 2), or x_i is some $d_i e_i$ with $Q \xrightarrow{d_i} Q \smallsetminus q_i \xrightarrow{e_i} Q \smallsetminus q_{\text{init}}, q_i$.

We now claim that the a_i's for type-1 states, the c_i's for type-2 states and the d_i's for type-3 states are all distinct, hence $|\Sigma| \geqslant n - 1$.

Clearly the a_i's and the d_i's are pairwise distinct since they take Q to different states in the deterministic powerset automaton. Similarly, the c_i's are pairwise distinct, taking $Q \smallsetminus q_{\text{init}}$ to different states.

Assume now that $a_i = c_j$ for a type-1 q_i and a type-2 q_j. Then $Q \smallsetminus q_{\text{init}} \xrightarrow{c_j} Q \smallsetminus q_{\text{init}}, q_j$ and $Q \xrightarrow{a_i(=c_j)} Q \smallsetminus q_{\text{init}}, q_i$, implying $q_i = q_j$ by monotonicity of δ (the fact that $P_1 \subseteq P_2$ implies $\delta(P_1, a) \subseteq \delta(P_2, a)$ for any $a \in \Sigma$).

Similarly, assuming $d_\ell = c_j$ leads to $Q \smallsetminus q_{\text{init}} \xrightarrow{c_j} Q \smallsetminus q_{\text{init}}, q_j$ and $Q \xrightarrow{c_j(=d_\ell)} Q \smallsetminus q_\ell$, implying $q_\ell = q_j$ by monotonicity of δ. Thus we can associate a distinct letter with each state $q_1, \ldots, q_{n-1}$, which concludes the proof. $\square$

In view of the above results, the main question is whether, *in the case of a fixed alphabet*, exponential lower bounds still apply for the (deterministic) state complexity of upward and downward closures. The 1-letter case is degenerate since, when $|\Sigma| = 1$, both $n_{\text{D}}(\uparrow L)$ and $n_{\text{D}}(\downarrow L)$ are at most $n_{\text{D}}(L)$. In the 3-letter case, exponential lower bounds for upward and downward closures were shown by Okhotin [39].

In the critical 2-letter case, say $\Sigma = \{a, b\}$, an exponential lower bound for upward closure was shown by Héam with the following witness: For $n > 0$, let $L_n'' = \{a^i b\, a^{2j} b\, a^i \mid i + j + 1 = n\}$. Then $n_D(L_n'') = (n+1)^2$, while $n_D(\uparrow L_n'') \geqslant \frac{1}{7}(\frac{1+\sqrt{5}}{2})^n$ when $n \geqslant 4$ [23, Proposition 5.11]. Regarding downward closures for languages over a 2-letter alphabet, the question was left open and we answer it in the next section.

4. Exponential state complexity of closures in the 2-letter case

In this section we show an exponential lower bound for the state complexity of downward closure in the case of a two-letter alphabet. Interestingly, the same lower bound for upward closure can be proved using the same witnesses, but Héam already gave a stronger lower bound for upward closure [23].

Theorem 4.1 (State complexity of closures with $|\Sigma| = 2$) *The deterministic state complexity of downward closure for languages over the binary alphabet $\Sigma = \{a, b\}$ is in $2^{\Omega(n^{1/3})}$. The same result holds for upward closure.*

We now prove the theorem. Fix $\Sigma = \{a, b\}$ and $n \in \mathbb{N}$. Let

$$H = \{n, n+1, \ldots, 2n\}\,,$$

and define morphisms $c, d : H^* \to \Sigma^*$ by

$$c(i) \overset{\text{def}}{=} a^i\, b^{3n-i}\,, \qquad\qquad d(i) \overset{\text{def}}{=} c(i)\, c(i)\,, \qquad\qquad (2)$$

for $i \in H$. Note that $c(i)$ always has length $3n$, begins with at least n a's, and ends with at least n b's. Let

$$L_n \overset{\text{def}}{=} \{c(i)^n \mid i \in H\}\,.$$

The language L_n is finite and contains $n+1$ words, each of length $3n^2$ so that $n_D(L_n)$ is in $O(n^3)$. In fact, $n_D(L_n) = 3n^3 + 1$.

In the rest of this section we show that, for n even and strictly positive, both $n_D(\uparrow L_n)$ and $n_D(\downarrow L_n)$ are greater than or equal to $\binom{n+1}{n/2}$. Since $\binom{n+1}{n/2} \approx \frac{2^{n+3/2}}{\sqrt{\pi n}}$ and $n_D(L_n) = 3n^3 + 1$, the languages $(L_n)_{n=2,4,6,\ldots}$ witness the lower bound claimed in Theorem 4.1.

For each $i \in H$, let the morphisms $\eta_i, \theta_i : H^* \to (\mathbb{N}, +)$ be defined by

$$\eta_i(j) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } i \neq j\,, \\ 2 & \text{if } i = j\,, \end{cases} \qquad\qquad \theta_i(j) \overset{\text{def}}{=} \begin{cases} 2 & \text{if } i \neq j\,, \\ 1 & \text{if } i = j\,. \end{cases}$$

Thus for $\sigma = p_1\, p_2 \cdots p_s \in H^*$, $\eta_i(\sigma)$ is s plus the number of occurrences of i in σ, while $\theta_i(\sigma)$ is $2s$ minus the number of these occurrences of i.

Lemma 4.2 *Let $\sigma \in H^*$. The smallest ℓ such that $c(\sigma)$ is a subword of $c(i)^\ell$ is $\theta_i(\sigma)$.*

PROOF. We write $\sigma = p_1 \, p_2 \cdots p_s$ and prove the result by induction on s. The case of $s = 0$ is trivial. For the case of $s = 1$, note that for any p_1 and i, $c(p_1) \sqsubseteq d(i) = c(i)^2$ and that $c(p_1) \sqsubseteq c(i)$ if and only if $p_1 = i$.

Assume now that $s > 1$, write $\sigma = \sigma' p_s$ and let $\ell' = \theta_i(\sigma')$. By the induction hypothesis, $c(\sigma') \not\sqsubseteq c(i)^{\ell'-1}$ and $c(\sigma') \sqsubseteq c(i)^{\ell'} = c(i)^{\ell'-1} a^i b^{3n-i}$. Write now $c(i)^{\ell'} = w \, v$ where w is the shortest prefix of $c(i)^{\ell'}$ with $c(\sigma') \sqsubseteq w$. Since $c(\sigma')$ ends with some b that only embeds in the suffix $a^i b^{3n-i}$ of $c(i)^{\ell'}$, v is necessarily b^r for some r. So, for all $z \in \Sigma^*$, $c(p_s) \sqsubseteq z$ if and only if $c(p_s) \sqsubseteq v \, z$. We have $c(p_s) \sqsubseteq c(i)^{\theta_i(p_s)}$ and $c(p_s) \not\sqsubseteq v \, c(i)^{\theta_i(p_s)-1}$. Noting that $\sigma = \sigma' p_s$, we get $c(\sigma) \sqsubseteq c(i)^{\theta_i(\sigma)}$ and $c(\sigma) \not\sqsubseteq c(i)^{\theta_i(\sigma)-1}$. $\qquad\square$

We now derive the announced lower bound on $n_{\mathrm{D}}(\downarrow L_n)$. Recall that n is even and strictly positive. For every subset X of H of size $n/2$, let $w_X \in \Sigma^*$ be defined as follows: let the elements of X be $p_1 < p_2 < \cdots < p_{n/2}$ and let

$$ w_X \stackrel{\mathrm{def}}{=} c(p_1 p_2 \cdots p_{n/2}) \, . $$

Note that $\theta_i(p_1 p_2 \cdots p_{n/2}) = n$ if $i \notin X$ and $\theta_i(p_1 p_2 \cdots p_{n/2}) = n - 1$ if $i \in X$.

Lemma 4.3 *Let X and Y be subsets of H of size $n/2$ with $X \neq Y$. There exists a word $v \in \Sigma^*$ such that $w_X v \in \downarrow L_n$ and $w_Y v \notin \downarrow L_n$.*

PROOF. Let $i \in X \smallsetminus Y$. Let $v = c(i)$. By Lemma 4.2, $w_X \sqsubseteq c(i)^{n-1}$, and so $w_X v \sqsubseteq c(i)^n$, hence $w_X v \in \downarrow L_n$.

By Lemma 4.2, the smallest ℓ such that $w_Y v \sqsubseteq c(i)^\ell$ is $n + 1$. Similarly, for $j \neq i$, the smallest ℓ such that $w_Y v \sqsubseteq c(j)^\ell$ is at least $n - 1 + 2 = n + 1$ (at least $n - 1$ for the w_Y factor and 2 for the v factor). So $w_Y v \notin \downarrow L_n$. $\qquad\square$

This shows that for any DFA $A = (\Sigma, Q, \delta, q_1, F)$ recognizing $\downarrow L_n$, the states of the form $\delta(q_1, w_X)$ for a subset $X \subseteq H$ with $|X| = n/2$ are all distinct. Thus A has at least $\binom{n+1}{n/2}$ states as claimed.

For $n_{\mathrm{D}}(\uparrow L_n)$, the reasoning is similar:

Lemma 4.4 1. *For $i, j \in H$, the longest prefix of $c(i)^\omega$ that is a subword of $d(j) = c(j) \, c(j)$ is $c(i)$ if $i \neq j$ and $c(i) \, c(i)$ if $i = j$.*
2. *Let $\sigma \in H^*$. For all $i \in H$, the longest prefix of $c(i)^\omega$ that is a subword of $d(\sigma)$ is $c(i)^{\eta_i(\sigma)}$.*

PROOF. 1. The statement is trivial when $i = j$, so we now assume $i \neq j$. Equation (2) entails $c(i) \sqsubseteq c(j) c(j)$ since $n \leqslant i, j \leqslant 2n$. It remains to show that no longer prefix of $c(i)^\omega$ embeds in $c(j) c(j)$, that is, that $c(i) a \not\sqsubseteq c(j) c(j)$. But this is clear when one considers the leftmost embedding of $c(i) a$ in $c(j) c(j)$: this is illustrated by Figure 5 in the case of $i > j$, the case of $i < j$ being similar.
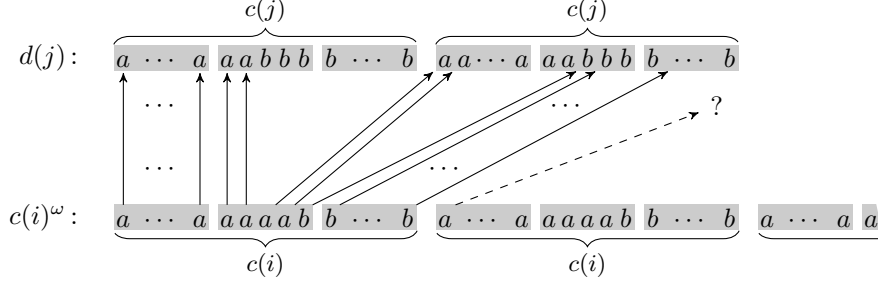2. By induction on the length of σ, as above. $\qquad\square$

$d(j)$:

$$\overbrace{a\ \cdots\ a}\ \overbrace{a\,a\,b\,b\,b\ \ b\ \cdots\ b}^{c(j)}\quad \overbrace{a\,a\cdots a}\ \overbrace{a\,a\,b\,b\,b\ \ b\ \cdots\ b}^{c(j)}$$

$c(i)^\omega$:

$$\underbrace{a\ \cdots\ a\ \underbrace{a\,a\,a\,a\,b\ \ b\ \cdots\ b}_{c(i)}}\quad \underbrace{a\ \cdots\ a\ \underbrace{a\,a\,a\,a\,b\ \ b\ \cdots\ b}_{c(i)}}\quad \underbrace{a\ \cdots\ a\ a}$$

Figure 5: Case "$i > j$" in Lemma 4.4 (here with $n = 5$, $i = n + 4$ and $j = n + 2$).

Recall that n is even and strictly positive. For every subset X of H of size $n/2$, let $w'_X \in \Sigma^*$ be defined as follows: let the elements of X be $p_1 < p_2 < \cdots < p_{n/2}$ and let

$$w'_X \stackrel{\text{def}}{=} d(p_1 p_2 \cdots p_{n/2}) = c(p_1 p_1 p_2 p_2 \cdots p_{n/2} p_{n/2})\,.$$

Lemma 4.5 *Let X and Y be subsets of H of size $n/2$ with $X \neq Y$. There exists a word $v \in \Sigma^*$ such that $w'_X v \in {\uparrow}L_n$ and $w'_Y v \notin {\uparrow}L_n$.*

PROOF. Let $i \in X \smallsetminus Y$. Let $v = c(i)^{n-(n/2+1)} = c(i)^{n/2-1}$. By Lemma 4.4, $c(i)^{n/2+1} \sqsubseteq w'_X$, thus $c(i)^n \sqsubseteq w'_X v$, hence $w'_X v \in {\uparrow}L_n$.

We now show that $w'_Y v \notin {\uparrow}L_n$. By Lemma 4.4, the longest prefix of $c(i)^n$ that embeds in $w'_Y v$ is a prefix of $c(i)^\ell$ where $\ell = n/2 + n/2 - 1 = n - 1$. Thus $w'_Y v \notin {\uparrow}c(i)^n$. For $j \neq i$, we show $c(j)^n \nsqsubseteq w'_Y v$ by contradiction. Suppose $c(j)^n \sqsubseteq w'_Y v$. The longest prefix of $c(j)^n$ that is a subword of w'_Y is a prefix of $c(j)^{n/2+1}$. Thus $c(j)^{n/2-1} \sqsubseteq v$. But $c(j)^{n/2-1}$ and v are different words of the same length, so this is not possible. Thus $c(j)^n \nsqsubseteq w'_Y v$. Finally $w'_Y v \notin {\uparrow}L_n$. $\square$

With Lemma 4.5 we reason exactly as we did for $n_{\mathrm{D}}({\downarrow}L_n)$ after Lemma 4.3 and conclude that $n_{\mathrm{D}}({\uparrow}L_n)$ is at least $\binom{n+1}{n/2}$.

5. State complexity of interiors

Recall Equation (1) expressing interiors with closures and complements. Since complementation of DFAs does not increase the number of states, except perhaps adding a single state if we start with an incomplete DFA, the state complexity of interiors, seen as DFA to DFA operations, is essentially the same as the state complexity of closures modulo swapping of up and down.

The remaining question is the *nondeterministic state complexity* of interiors, now seen as NFA to NFA operations. For this, Equation (1) provides an obvious 2^{2^n} upper bound on the nondeterministic state complexity of both upward and downward interiors, simply by combining the powerset construction for complementation and the results of Section 3. Note that this procedure yields DFAs

for the interiors while we are happy to accept NFAs if this improves the state complexity.

In the rest of this section, we prove that the nondeterministic state complexity of upward and downward interiors is in $2^{2^{\Theta(n)}}$. Sections 5.1 and 5.2 establish the upper and lower bounds, respectively. Section 5.3 mentions the consequences on representations based on alternating automata.

5.1. Upper bounds for interiors and the approximation problem

We first give an upper bound for the state complexity of interiors that slightly improves on the obvious 2^{2^n} upper bound. For this we adapt a technique from [13, 38] and rely on the fact, already used in [11, Theorem 6.1], that the state complexity of a positive Boolean combination of left-quotients of some regular language L is at most $\psi(n_{\mathrm{N}}(L))$.

Proposition 5.1 Let $L \subseteq \Sigma^*$ be a regular language with $n_N(L) = n$. Then $n_D(\circlearrowright L) < \psi(n)$ and $n_D(\circlearrowleft L) < \psi(n)$.

PROOF. We handle both interiors in a uniform way.

Let K_0 and $K_1, \ldots, K_p$ be arbitrary languages in Σ^* (these need not be regular). With the K_i's we associate an alphabet $\Gamma = \{b_1, \ldots, b_p\}$ and a substitution σ given inductively by $\sigma(\varepsilon) \stackrel{\text{def}}{=} K_0$ and $\sigma(w\, b_i) \stackrel{\text{def}}{=} \sigma(w) \cdot K_i$. With a language $L \subseteq \Sigma^*$, we associate the language $W \subseteq \Gamma^*$ defined by

$$W \stackrel{\text{def}}{=} \{x \in \Gamma^* \mid \sigma(x) \subseteq L\}. \tag{3}$$

CLAIM. If L is regular then W is regular.

To prove this first claim, assume $A_1 = (\Sigma, Q, \delta_1, I_1, F_1)$ is an n-state NFA recognizing L. Using the powerset construction, one obtains a DFA $A_2 = (\Sigma, Q_2, \delta_2, i_2, F_2)$ recognizing L. We have as usual $Q_2 = 2^Q$, with typical elements $S, S', \ldots$, δ_2 given by $\delta_2(S, a) = \bigcup_{q \in S} \delta_1(q, a)$, $i_2 = I_1$, and $F_2 = \{S \mid S \cap F_1 \neq \varnothing\}$.

From A_2 we now derive a DFA $A_3 = (\Gamma, Q_3, \delta_3, i_3, F_3)$ given by $Q_3 = 2^{Q_2}$, with typical elements $U, U', \ldots$; $\delta_3(U, b_j) = \{\delta_2(S, z) \mid S \in U, z \in K_j\}$; $i_3 = \{\delta_2(i_2, z) \mid z \in K_0\}$; and $F_3 = 2^{F_2} = \{U \mid U \subseteq F_2\}$.

The intention is that A_3 will recognize W, so let us check, using induction on $w \in \Gamma^*$, that $\delta_3(i_3, w) = \{\delta_2(i_2, z) \mid z \in \sigma(w)\}$: For the base case, one has $\delta_3(i_3, \varepsilon) = i_3 = \{\delta_2(i_2, z) \mid z \in K_0\}$ by definition, and $\sigma(\varepsilon) = K_0$. For the inductive case, one has

$$\begin{aligned}
\delta_3(i_3, w\, b_j) &= \delta_3(\delta_3(i_3, w), b_j) \\
&= \delta_3(\{\delta_2(i_2, z) \mid z \in \sigma(w)\}, b_j) && \text{(induction hypothesis)} \\
&= \{\delta_2(S, z') \mid S \in \{\delta_2(i_2, z) \mid z \in \sigma(w)\}, z' \in K_j\} && \text{(definition of } \delta_3) \\
&= \{\delta_2(\delta_2(i_2, z), z') \mid z \in \sigma(w), z' \in \sigma(b_j)\} && \text{(rearrange, use } \sigma(b_j) = K_j) \\
&= \{\delta_2(i_2, z'') \mid z'' \in \sigma(w\, b_j)\}.
\end{aligned}$$

14

Now, for all $w \in \Gamma^*$, one has

$$
\begin{aligned}
w \in W &\iff \sigma(w) \subseteq L && \text{(definition of } W) \\
&\iff \forall z \in \sigma(w) : \delta_2(i_2, z) \in F_2 && \text{(since } A_2 \text{ recognizes } L) \\
&\iff \{\delta_2(i_2, z) \mid z \in \sigma(w)\} \subseteq F_2 \\
&\iff \delta_3(i_3, w) \in F_3 \, . && \text{(as just shown)}
\end{aligned}
$$

This proves that A_3 recognizes W. In particular, W is regular as claimed.

CLAIM. $n_{\mathrm{D}}(W) < \psi(n)$.

The DFA A_3 that recognizes W has $|Q_3| = 2^{2^n}$ states. We now examine our construction more closely to detect equivalent states in A_3. Observe that the powerset construction for A_2 in terms of A_1 is "existential", that is, a state of A_2 is accepting if and only if at least one of its constituent states from A_1 is accepting. In contrast, the powerset construction for A_3 in terms of A_2 is "universal", that is, a state of A_3 is accepting if and only if all of its constituent states from A_2 are accepting. Suppose $S, S' \in Q_2$ are two states of A_2 with $S \subseteq S'$. Then if some word is accepted by A_2 starting from S, it is also accepted starting from S'. If a state of A_3 contains both S and S', then S already imposes a stronger constraint than S', and so S' can be eliminated. We make this precise below:

Define an equivalence relation $\equiv$ on Q_3 as follows:

$$
U \equiv U' \stackrel{\mathrm{def}}{\iff} (\forall S \in U : \exists S' \in U' : S' \subseteq S) \wedge (\forall S' \in U' : \exists S \in U : S \subseteq S') \, .
$$

We now claim that, in A_3, $\equiv$-equivalent states accept the same language. First $U \equiv V$ and $U \in F_3$ imply $V \in F_3$ since for any $S' \in V$, there is $S \in U$ with $S \subseteq S'$, and since $S \in F_2$, also $S' \in F_2$. Furthermore $U \equiv V$ and $b_j \in \Gamma$ imply $\delta(U, b_j) \equiv \delta(V, b_j)$: each element of $\delta_3(U, b_j)$ is some $\delta_2(S, z)$ with $S \in U$ and $z \in K_j$. There exists $S' \in V$ such that $S' \subseteq S$, and then $\delta_2(S', z)$ belongs to $\delta_3(V, b_j)$ and is a subset of $\delta_2(S, z)$ because δ_2 is monotone in its first argument. The reasoning in the reverse direction is similar.

Thus we can quotient the DFA A_3 by $\equiv$ to get an equivalent DFA recognizing W. Further, we can remove (the equivalence class of) the sink state $\{\varnothing\}$, so that $n_{\mathrm{D}}(W) < |Q_3/{\equiv}|$.

Let us now show that $|Q_3/{\equiv}|$ is exactly $\psi(|Q|)$. A state $U \in Q_3$ is called an *antichain* if it does not contain some $S, S' \in Q_2$ with $S \subsetneq S'$. Every $U \in Q_3$ is $\equiv$-equivalent to the antichain $U_{\min}$ obtained by retaining only the elements of U that are minimal by inclusion. Further, two distinct antichains cannot be $\equiv$-equivalent. Thus the number of equivalence classes in $Q_3/{\equiv}$ is exactly the number of subsets of 2^Q which are antichains, and this is the Dedekind number $\psi(n)$, see [34]. This shows $n_{\mathrm{D}}(W) < \psi(n)$, completing the proof of our second claim.

We may now instantiate the above construction for the upward and downward interiors. Choose alphabets $\Sigma = \Gamma = \{b_1, \ldots, b_k\}$ and let $K_0 = \Sigma^*$

and $K_i = \Sigma^* b_i \Sigma^*$. Then Equation (3) yields $W = \circlearrowleft(L)$ and we deduce $n_{\mathrm{D}}(\circlearrowleft(L)) < \psi(n)$. Letting now $K_0 = \{\varepsilon\}$ and $K_i = \{b_i, \varepsilon\}$ yields $W = \Omega(L)$ and again we deduce $n_{\mathrm{D}}(\Omega(L)) < \psi(n)$. This concludes the proof of Proposition 5.1.

<div align="right">□</div>

Remark 5.2 *In the usual setting – see [38, Section 6] – W is defined with $\sigma(\varepsilon) = \{\varepsilon\}$ and there is no need for K_0. The idea is that W is the best under-approximation of L by sums of products of K_i's, and Conway showed that if L is regular then W is too [13]. We allowed $\sigma(\varepsilon) = K_0$ to account directly for upward interiors.*

5.2. Lower bounds for interiors

Proposition 5.3 (Downward interior) *There exists a family of languages $(L_n)_{n=3,4,\ldots}$ with $n_N(L_n) \leqslant n$ and $n_N(\Omega L_n) = 2^{2^{\left\lfloor \frac{n-3}{2} \right\rfloor}}$.*

PROOF. Fix $n \geqslant 3$ and let $\ell = \left\lfloor \frac{n-3}{2} \right\rfloor$. We let $\Sigma = \{0, 1, 2, \ldots, 2^\ell - 1\}$, so that $|\Sigma| = 2^\ell$. Let

$$L_n \stackrel{\text{def}}{=} \Sigma^* \smallsetminus \{a\, a \mid a \in \Sigma\} = \{a\, b \mid a, b \in \Sigma, a \neq b\} \cup \{w \in \Sigma^* \mid |w| \neq 2\}.$$

That is, L_n contains all words over Σ consisting of two different letters and all words whose length is not 2.

We first prove that $n_{\mathrm{N}}(L_n) \leqslant 2\ell + 3 \leqslant n$: Two letters in Σ, viewed as ℓ-bit sequences, are distinct if and only if they differ in at least one bit. Figure 6 displays an NFA recognizing $\{a\, b \in \Sigma^2 \mid a \neq b\}$ with $2\ell + 2$ states: the idea is that the NFA reads a, guesses the position of a bit where a and b differ, records the value of a's corresponding bit and checks b's bit at that position.
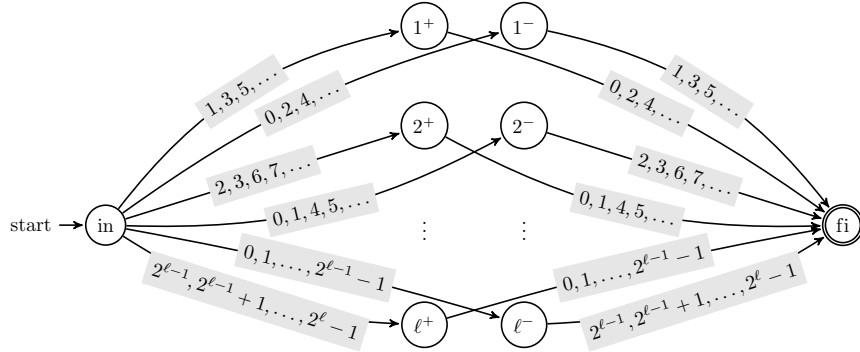

Figure 6: NFA recognizing $\{a\, b \mid a, b \in \Sigma_{2^\ell}, a \neq b\}$ with $2\ell + 2$ states.

We then modify this NFA so that it also accepts all words whose length is not 2. This can be done by adding a single new state and appropriate transitions, and making all original states accepting. The resulting NFA has $2\ell + 3$

states.

It remains to prove that $n_N(\mathcal{Q}L_n) = 2^{2^\ell}$, but $\mathcal{Q}L_n$ consists of all words in Σ^* of pairwise distinct letters, the language called V_Σ in Lemma 2.2 where we showed $n_N(V_\Sigma) = 2^{|\Sigma|} = 2^{2^\ell}$. $\qquad\square$

Proposition 5.4 (Upward interior) *There exists a family of languages* $(L_n)_{n=7,8,\ldots}$ *with* $n_N(L_n) \leqslant n$ *and* $n_N(\mathcal{U}L_n) \geqslant 2^{2^{\left\lfloor \frac{n-4}{3} \right\rfloor}} + 1$.

PROOF. Fix $n \geqslant 7$ and let $\ell = \left\lfloor \frac{n-4}{3} \right\rfloor$. We use two subalphabets: $\Gamma \stackrel{\text{def}}{=} \{0, 1, \ldots, 2^\ell - 1\}$ and $\Upsilon \stackrel{\text{def}}{=} \{1, \ldots, \ell\}$, letting $\Sigma \stackrel{\text{def}}{=} \Gamma \cup \Upsilon$. The symbols in Γ, denoted $x, y, \ldots$ are disjoint from the symbols in Υ, denoted $k, k', \ldots$ (for example, we can imagine that they have different colors) and one has $|\Sigma| = 2^\ell + \ell$.

For $x, y \in \Gamma$ and $k \in \Upsilon$, we write $x =_k y$ when x and y, viewed as ℓ-bit sequences, have the same kth bit. We consider the following languages:

$$
\begin{aligned}
L_n' &\stackrel{\text{def}}{=} \{x\,w\,y\,k\,w' \in \Gamma \cdot \Sigma^* \cdot \Gamma \cdot \Upsilon \cdot \Sigma^* \mid x =_k y\}\,, \\
L_n'' &\stackrel{\text{def}}{=} \Gamma \cdot (\Gamma \cdot \Upsilon)^*\,, \\
L_n &\stackrel{\text{def}}{=} L_n' \cup (\Sigma^* \smallsetminus L_n'')\,.
\end{aligned}
$$

In other words, L_n' contains all words such that the initial letter $x \in \Gamma$ has one common bit with a later $y \in \Gamma$ and this bit is indicated by the $k \in \Upsilon$ that immediately follows the occurrence of y.

CLAIM. $n_N(L_n') \leqslant 3\ell + 2$ and $n_N(L_n) \leqslant n$.

Figure 7 displays the schematics of an NFA for L_n'. In order to recognize inputs of the form $x\,w\,y\,k\,w' \in \Gamma \cdot \Sigma^* \cdot \Gamma \cdot \Upsilon \cdot \Sigma^*$ with $x =_k y$, the NFA reads the first letter x, nondeterministically guesses k, and switches to a state r_k^+ or r_k^- depending on what the kth bit of x is. From there it waits nondeterministically for the appearance of a factor $y\,k$ with $x =_k y$ before accepting. This uses $3\ell + 2$ states. Adding states for $\Sigma^* \smallsetminus L_n''$, one obtains $n_N(L_n) \leqslant 3\ell + 4 \leqslant n$.

We now consider the upward interior of L_n. Let $U_\Gamma, U_\Gamma' \subseteq \Gamma^*$ be as in Lemma 2.2: a word w is in U_Γ if it uses each letter from Γ at least once, and $U_\Gamma' \stackrel{\text{def}}{=} \Gamma \cdot U_\Gamma$.

CLAIM. $\Gamma^* \cap \mathcal{U}L_n = U_\Gamma'$.

We first show $\Gamma^* \cap \mathcal{U}L_n \subseteq U_\Gamma'$ by showing the contrapositive. Let $w \in \Gamma^* \smallsetminus U_\Gamma'$. If $w = \varepsilon$, then clearly $w \notin \mathcal{U}L_n$. Otherwise, $w = z\,z_1 \cdots z_p$, where $z, z_i \in \Gamma$. Since $z_1 \cdots z_p$ is not in U_Γ, there is some $x \in \Gamma$ that differs from all the z_i's. Pick $k_1, \ldots, k_p$ witnessing this, that is, such that $x \neq_{k_i} z_i$ for all i. If $x = z$ we let $w' \stackrel{\text{def}}{=} z\,z_1\,k_1 \cdots z_p\,k_p$ so that $w' \in L_n''$ and $w' \notin L_n'$, that is, $w' \notin L_n$. If $x \neq z$ we let $w' \stackrel{\text{def}}{=} x\,z\,k\,z_1\,k_1 \cdots z_p\,k_p \notin L_n$ for some k witnessing $x \neq z$, so that $w' \notin L_n$. In both cases $w \sqsubseteq w' \notin L_n$ and we deduce $w \notin \mathcal{U}L_n$.
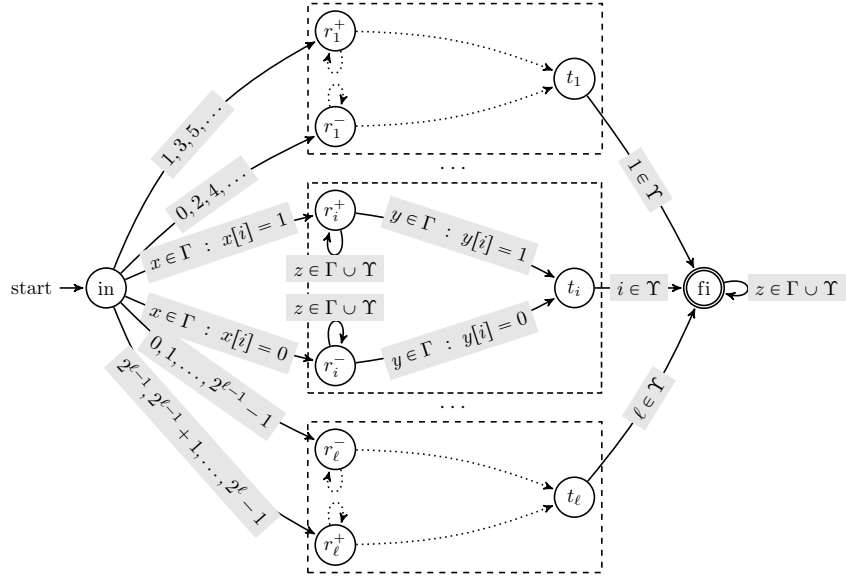
Figure 7: NFA recognizing L'_n with $3\ell + 2$ states.

We now show $U'_\Gamma \subseteq \Gamma^* \cap \circlearrowleft L_n$. Let $w = z\, z_1 \cdots z_p \in U'_\Gamma$. We show that $w \in \circlearrowleft L_n$ by showing that $w' \in L_n$ for every w' such that $w \sqsubseteq w'$. If $w' \notin L''_n$, then $w' \in L_n$. So assume $w' = x\, y_1\, k_1 \cdots y_n\, k_n \in L''_n$. There is some i such that $x = z_i$ (since $w \in U'_\Gamma$) and some j such that $z_i = y_j$ (since $w \sqsubseteq w'$). We then have $x =_{k_j} y_j$ (this does not depend on the actual value of k_j). Hence $w' \in L'_n \subseteq L_n$. Thus $w \in \circlearrowleft L_n$.

We are now ready to conclude the proof of Proposition 5.4. Recall that $n_{\mathrm{N}}(L \cap \Gamma^*) \leqslant n_{\mathrm{N}}(L)$ holds for any regular L and any alphabet Γ. In particular the above claim entails $n_{\mathrm{N}}(U'_\Gamma) \leqslant n_{\mathrm{N}}(\circlearrowleft L_n)$. Combining with $n_{\mathrm{N}}(U'_\Gamma) = 2^{2^\ell} + 1$ from Lemma 2.2 yields the required $n_{\mathrm{N}}(\circlearrowleft L_n) \geqslant 2^{2^{\left\lfloor \frac{n-4}{3} \right\rfloor}} + 1$. □

The doubly-exponential lower bounds exhibited in Propositions 5.3 and 5.4 rely on alphabets of exponential size. It is an open question whether, in the case of a fixed alphabet, the nondeterministic state complexity of downward or upward interiors is still doubly-exponential.

5.3. On alternating automata for closures

The state-complexity analysis of interiors can be used to show lower bounds on the computation of closures for regular languages represented via alternating automata (AFAs). The question was recently raised in [25] where it is suggested that the construction of a piecewise-testable separator could be done more efficiently by using AFAs for representing regular languages. It is indeed natural to ask whether an AFA recognizing $\downarrow L$ or $\uparrow L$ can be built efficiently from an

AFA recognizing L, perhaps in the same spirit as the constructions for closures on NFAs.

In the rest of this section we briefly justify the claims on AFAs made in Table 1 in the introduction of this article. We assume basic knowledge of AFAs (otherwise see [11, Section 6]) and write $n_A(L)$ to denote the minimal number of states of an AFA recognizing L.

For the upper bounds, recall that an AFA A can be transformed into an equivalent NFA A' with the powerset construction. If A has n states, A' has 2^n states. We deduce that if $n_A(L) = n$, then $n_A(\uparrow L) \leqslant n_N(\uparrow L) \leqslant n_N(L) \leqslant 2^n$ and $n_A(\downarrow L) \leqslant n_N(\downarrow L) \leqslant n_N(L) \leqslant 2^n$.

For the lower bounds, we can reuse the witness languages from Section 5.2. Recall the properties of $L_n \subseteq \Sigma^*$ from Proposition 5.3. We showed that $n_N(L_n) \leqslant n$, entailing $n_A(L_n) \leqslant n$. Hence $n_A(\Sigma^* \smallsetminus L_n) \leqslant n$ since one can complement an AFA without any increase in the number of states. Let A be an ℓ-state AFA recognizing $\uparrow(\Sigma^* \smallsetminus L_n)$. By complementing A, we get an ℓ-state AFA recognizing $\Sigma^* \smallsetminus \uparrow(\Sigma^* \smallsetminus L_n)$, that is, $\Diamond L_n$. Transforming this into an NFA, we get a 2^ℓ-state NFA that recognizes $\Diamond L_n$. Using Proposition 5.3 we deduce $\ell \geqslant 2^{\lfloor \frac{n-3}{2} \rfloor}$. Thus the languages $(\Sigma^* \smallsetminus L_n)_{n=3,4,\ldots}$ witness the lower bound for $n_A(\uparrow L)$ claimed in Table 1.

For $n_A(\downarrow L)$ we use the same reasoning, with up and down interchanged, and based on the witnesses $(L_n)_{n=7,8,\ldots}$ used in Proposition 5.4.

6. On unambiguous automata

Recall that an unambiguous automaton (a UFA) is an NFA A in which every accepted word is accepted by exactly one run. When handling regular languages it is sometimes interesting to work with UFAs since, like NFAs, they can be exponentially more succinct than DFAs and, like DFAs, they admit polynomial-time algorithms for testing inclusion or equality, see [12] and references therein. With this in mind, it was natural to state in Section 3 that upward or downward closures are in general not more succinct when given in the form of UFAs. We now prove these specific claims.

Lower bounds on the size of UFAs can be shown via the following lemma:

Lemma 6.1 (Fooling sets for unambiguous automata, after Schmidt)
Given a regular language L and a set of m pairs of words $S = \{(x_i, y_i)\}_{1 \leqslant i \leqslant m}$, let $M_{L,S}$ be the $m \times m$ matrix given by $M[i,j] = 1$ if $x_i y_j \in L$, and $M[i,j] = 0$ otherwise. Let $r = rank(M_{L,S})$. Then any UFA for L has at least r witness states, where a witness state is any state that accepts at least one of the y_i's.

The above lemma is actually a refinement of Theorem 2 from [37] where the lower bound is given for $n_U(L)$: the proof by Leung easily adapts to Lemma 6.1

19

since non-witness states contribute a null row in the matrix M' one derives from M in [37].

We shall also use the following result by Leung:

Lemma 6.2 ([36]) *Let X be an n-element set and consider M_X, the $2^n \times 2^n$ matrix with rows and columns indexed by subsets of X, given by $M[Y, Z] = 1$ if $Y \cap Z \neq \varnothing$, $M[Y, Z] = 0$ otherwise. Then $rank(M_X) = 2^n - 1$.*

As a first application, let us consider the language $\Sigma_n^* \setminus U_n$ from Remark 3.3 (see also Figure 3). Recall that $\Sigma_n^* \setminus U_n$ contains all words where at least one letter from Σ_n does not occur.

Proposition 6.3 *For any $n > 0$, $n_U(\Sigma_n^* \setminus U_n) = 2^n - 1$.*

PROOF. The upper bound is clear since already $n_D(\Sigma_n^* \setminus U_n) = 2^n - 1$.

For the lower bound, consider $S = \{(x_{\neg\Gamma}, x_{\neg\Gamma})\}_{\Gamma \subseteq \Sigma_n}$ (recall that the words x_Γ and $x_{\neg\Gamma}$ with $\Gamma \subseteq \Sigma$ were introduced in the proof of Lemma 2.2). The associated matrix has $M_{\Sigma_n^* \setminus U_n, S}[x_{\neg\Gamma_i}, x_{\neg\Gamma_j}] = 1$ if $x_{\neg\Gamma_i} x_{\neg\Gamma_j} \notin U_n$, that is, if $\Gamma_i \cap \Gamma_j \neq \varnothing$. Note that this is exactly the M_X matrix from Lemma 6.2, instantiated with $X = \Sigma_n$. So $rank(M_{\Sigma_n^* \setminus U_n, S}) = 2^n - 1$ and, by Lemma 6.1, we can conclude that $n_U(\Sigma_n^* \setminus U_n) \geqslant 2^n - 1$. $\qquad\square$

The language $\downarrow D_n$ from Section 3 is a small variation: recall that $\downarrow D_n$ contains all words $x \in \Sigma_n^*$ whose first suffix $x[2..]$ does not use all letters.

Proposition 6.4 *For any $n > 0$, $n_U(\downarrow D_n) = 2^n$.*

PROOF. The upper bound is clear since already $n_D(\downarrow D_n) = 2^n$.

For the lower bound, consider $S = \{(a_1 x_{\neg\Gamma}, x_{\neg\Gamma})\}_{\Gamma \subseteq \Sigma_n} \cup \{(\varepsilon, x_{\neg\varnothing})\}$ where a_1 is the first letter of Σ_n. The associated matrix $M_{\downarrow D_n, S}$ has

$$M_{\downarrow D_n, S}[a_1 x_{\neg\Gamma_i}, x_{\neg\Gamma_j}] = 1 \text{ if and only if } \Gamma_i \cap \Gamma_j \neq \varnothing,$$

$$M_{\downarrow D_n, S}[\varepsilon, x_{\neg\varnothing}] = 1, \quad M_{\downarrow D_n, S}[a_1 x_{\neg\Gamma_i}, x_{\neg\varnothing}] = 0,$$

that is,

$$M_{\downarrow D_n, S} = \begin{pmatrix} M_{\Sigma_n} & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ 1 \; \cdots \; 1 & 1 \end{pmatrix}.$$

We note that the column representing $x_{\neg\varnothing}$ (that is, the last column) occurs twice in $M_{\downarrow D_n, S}$, as the word $x_{\neg\varnothing}$ occurs twice as the second component in S. One has $rank(M_{\downarrow D_n, S}) = rank(M_{\Sigma_n}) + 1 = 2^n$ since the rightmost column forbids combining the last row with any of the earlier rows. We deduce $n_U(\downarrow D_n) \geqslant 2^n$ with Lemma 6.1. $\qquad\square$

We finally consider $\uparrow E_n$ from Section 3. Recall that $\uparrow E_n$ contains all words over Σ_n in which at least one letter reappears.

Proposition 6.5 *For any $n > 0$, $n_U(\uparrow E_n) = 2^n + 1$.*

PROOF. The upper bound is clear since already $n_D(\uparrow E_n) = 2^n + 1$.

For the lower bound, consider $S = \{(x_\Gamma, x_\Gamma)\}_{\Gamma \subseteq \Sigma_n} \cup \{(a_1 a_1, \varepsilon)\}$. The associated matrix $M_{\uparrow E_n, S}$ has

$$M_{\uparrow E_n, S}[x_{\Gamma_i}, x_{\Gamma_j}] = 1 \text{ if and only if } \Gamma_i \cap \Gamma_j \neq \varnothing ,$$

$$M_{\uparrow E_n, S}[a_1 a_1, x_{\Gamma_j}] = 1 , \quad M_{\uparrow E_n, S}[x_{\Gamma_i}, \varepsilon] = 0 ,$$

that is,

$$M_{\uparrow E_n, S} = \begin{pmatrix} M_{\Sigma_n} & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ 1 \ \cdots \ 1 & 1 \end{pmatrix} .$$

Again one has $rank(M_{\uparrow E_n, S}) = 2^n$ so that, by Lemma 6.1, any UFA for $\uparrow E_n$ has at least 2^n witness states. Note however that the pairs (x_i, y_i) in S are such that no y_i belongs to $\uparrow E_n$. Hence in any automaton accepting $\uparrow E_n$ the initial state is not a witness state. We conclude that $n_U(\uparrow E_n) \geqslant 2^n + 1$. $\square$

7. Complexity of decision problems on closures

In automata-based procedures for logic and verification, the state complexity of automata constructions is not always the best measure of computational complexity. In this section we gather some elementary results on the complexity of subword-related decision problems for automata: for finite automata A, B we want to know whether the accepted language $L(A)$ is downward or upward closed, respectively, and whether $L(A)$ and $L(B)$ have the same downward or upward closures. These questions are in the spirit of the work done in [10, 31, 42] for various notions of closures. Some of the results we give are already known but are scattered in the literature and sometimes even reappear as open questions (see, for example, [16]).

7.1. Deciding closedness

Deciding whether $L(A)$ is upward-closed or downward-closed is, unsurprisingly, PSPACE-complete for NFAs, and NL-complete for DFAs. For upward-closedness this is already shown in [23], and quadratic-time algorithms that decide upward-closedness of $L(A)$ for a DFA A already appear in [2, 41].

Proposition 7.1 *Deciding whether $L(A)$ is upward-closed or downward-closed is PSPACE-complete when A is an NFA, even in the 2-letter alphabet case.*

PROOF. Membership in PSPACE is clear since it is enough to decide whether A and $A^\uparrow$ or $A^\downarrow$ accept the same language.

PSPACE-hardness can be shown by adapting the proof for hardness of universality. Let R be a length-preserving semi-Thue system and x, x' two strings of same length. It is PSPACE-hard to say whether there is a derivation $x \xrightarrow{*}_R x'$, even for a fixed R over a 2-letter alphabet Σ. We reduce (the negation of) this question to our problem.

Fix x and x' of length $n > 1$: a word $x_1 \, x_2 \cdots x_m$ of length $n \times m$ encodes a derivation if $x_1 = x$, $x_m = x'$, and $x_i \to_R x_{i+1}$ for all $i = 1, \ldots, m - 1$. The language $L_{R,x,x'}$ of words that do *not* encode a derivation from x to x' is regular and recognized by an NFA with $O(n)$ states. Now, there is a derivation $x \xrightarrow{*}_R x'$ if and only if $L_{R,x,x'} \neq \Sigma^*$. We conclude by observing that $L_{R,x,x'} = \Sigma^*$ if and only if $L_{R,x,x'}$ is upward-closed or, equivalently, downward-closed; this is because $L_{R,x,x'}$ contains all words of length not divisible by $n > 1$. $\square$

Proposition 7.2 *Deciding whether $L(A)$ is upward-closed or downward-closed is NL-complete when A is a DFA, even in the 2-letter alphabet case.*

PROOF. We only prove the result for upward-closure since L is downward-closed if and only if $\Sigma^* \smallsetminus L$ is upward-closed, and since one easily builds a DFA for the complement of $L(A)$.

For membership in NL, we first observe that L is upward-closed if and only if, for all $u, v \in \Sigma^*$, $u\,v \in L$ implies $u\,a\,v \in L$ for all $a \in \Sigma$. Therefore, $L(A)$ is not upward-closed – for $A = (\Sigma, Q, \delta, q_{\text{init}}, F)$ – if and only if there are states $p, q \in Q$, a letter a, and words u, v such that $\delta(q_{\text{init}}, u) = p$, $\delta(p, a) = q$, $\delta(p, v) \in F$ and $\delta(q, v) \notin F$. If such words exist, one can, in particular, find witnesses with $|u| < n$ and $|v| < n^2$ where $n = |Q|$ is the number of states of A. Hence checking that $L(A)$ is not upward-closed can be performed in nondeterministic logarithmic space by guessing u, a, and v within the above length bounds, finding p and q by running $u\,a$ from q_{init}, then running v from both p and q. Since coNL = NL, we conclude that upward-closedness too is in NL.

For NL-hardness, one may reduce from vacuity of DFAs, a well-known NL-hard problem that is essentially equivalent to GAP, the Graph Accessibility Problem. Note that for any DFA, and in fact any NFA, A with n states the following equivalences hold:

$$L(A) \cap \Sigma^{<n} \text{ is upward-closed} \iff L(A) \cap \Sigma^{<n} = \varnothing \iff L(A) = \varnothing \,.$$

This provides the required reduction since, given a DFA A, one easily builds a DFA for $L(A) \cap \Sigma^{<n}$ in logspace. $\square$

7.2. Deciding equivalence modulo closure

The question whether $\downarrow L(A) = \downarrow L(B)$ or, similarly, whether $\uparrow L(A) = \uparrow L(B)$, is relevant in some settings where closures are used to build regular over-approximations of more complex languages.

Bachmeier *et al.* recently showed that the above two questions are coNP-complete when A and B are NFAs [3, Section 5], hence "easier" than deciding whether $L(A) = L(B)$. Here we give an improved version of their result.

Proposition 7.3 (after [3]) *1. Deciding whether $\downarrow L(A) \subseteq \downarrow L(B)$ or whether $\uparrow L(A) \subseteq \uparrow L(B)$ is* coNP-*complete when A and B are NFAs.*

2. Deciding $\downarrow L(A) = \downarrow L(B)$ or $\uparrow L(A) = \uparrow L(B)$ is coNP-*hard even when A and B are DFAs over a two-letter alphabet.*

3. These problems are NL-*complete when restricting to NFAs over a 1-letter alphabet.*

PROOF. 1. Let $B = (\Sigma, Q, \delta, I, F)$ and $n_B = |Q|$. Assume that $\downarrow L(A) \nsubseteq \downarrow L(B)$ and pick a shortest witness $x = x_1 \cdots x_\ell \in \Sigma^*$ with $x \in \downarrow L(A)$ and $x \notin \downarrow L(B)$. We claim that $|x| < n_B$: indeed in the deterministic powerset automaton obtained from $B^\downarrow$, the unique run $S_0 \xrightarrow{x_1} S_1 \xrightarrow{x_2} \cdots \xrightarrow{x_\ell} S_\ell$ of x is such that $Q = S_0 \supseteq S_1 \supseteq S_2 \cdots \supseteq S_\ell \neq \varnothing$ (recall the proof of Lemma 3.4). If $S_{i-1} = S_i$ for some i, a shorter witness is obtained by omitting the ith letter in x: this does not affect membership in $\downarrow L(A)$ since this language is downward-closed. One concludes that the S_i have strictly diminishing sizes, hence $\ell < n_B$. This leads to an NP algorithm deciding $\downarrow L(A) \nsubseteq \downarrow L(B)$: guess x in $\Sigma^{<n_B}$ and check in polynomial time that it is accepted by $A^\downarrow$ and not by $B^\downarrow$.

For upward closure the reasoning is even simpler: a shortest witness x with $x \in \uparrow L(A)$ and $x \notin \uparrow L(B)$ has length $|x| < n_A$: if x is longer, a pumping lemma allows one to find a subword $x' \in \uparrow L(A)$, and $x' \notin \uparrow L(B)$ since $x \notin \uparrow L(B)$.

2. coNP-hardness is shown by reduction from validity of DNF-formulae. Consider an arbitrary DNF formula $\phi = C_1 \vee C_2 \vee \cdots \vee C_m$ consisting of m conjunctions of literals where k Boolean variables $v_1, \ldots, v_k$ may appear, for example, $\phi = (v_1 \wedge \neg v_2 \wedge v_4) \vee (v_2 \wedge \cdots) \vee \cdots$. The language of all the valuations, seen as words in $\{0, 1\}^k$, under which ϕ holds true is recognized by an NFA that has size $O(|\phi|^2)$. We slightly modify this language so that we can use a DFA instead of an NFA. Let $L_\phi = \{1^\ell 0\, x_1 \cdots x_k \in \{0, 1\}^* \mid 0 \leqslant \ell < m \wedge x_1 \cdots x_k \models C_{\ell+1}\}$. We build a DFA A_ϕ, having $m(k+2)$ states, that recognizes L_ϕ: see Figure 8 where, for the sake of readability, the picture uses wavy edges where A_ϕ recognizes a $1^\ell 0$ prefix, and standard edges where it recognizes the encoding of a valuation $x_1 \cdots x_k$ proper.

Now let B_ϕ be a DFA for $L_\phi \cup 1^m 0 (0 + 1)^k$, where all valuations are allowed after the $1^m 0$ prefix, and observe that $\uparrow L(A_\phi) = \uparrow L(B_\phi)$ if and only if $1^m 0 (0 + 1)^k \subseteq \uparrow L(A_\phi)$. However, $1^m 0\, x_1 \cdots x_k \in \uparrow L(A_\phi)$ requires that $1^\ell 0\, x_1 \cdots x_k \in L(A_\phi)$ for some $\ell \leqslant m$. Finally, $\uparrow L(A_\phi) = \uparrow L(B_\phi)$ if and only if all valuations make ϕ true, that is, if ϕ is valid. Since A_ϕ and B_ϕ are built in logspace from ϕ, this completes the reduction for equality of upward closures.

For downward closures, we modify A_ϕ by adding a transition $c_m \xrightarrow{1} c_1$ so that the resulting A'_ϕ accepts all words $1^\ell 0\, x_1 \cdots x_k$ such that $x_1 \cdots x_k$ makes $C_{\ell'+1}$ true for $\ell' = \ell \mod m$. For B we now take a DFA for $1^*0(0+1)^k$ and
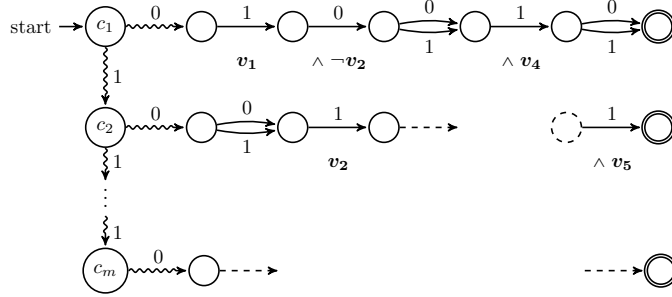
Figure 8: DFA A_ϕ for $\phi = (v_1 \wedge \neg v_2 \wedge v_4) \vee (v_2 \wedge \cdots \wedge v_5) \vee \cdots \vee C_m$ with $k = 5$ variables.

see that $\downarrow L(A'_\phi) = \downarrow L(B)$ if and only if all valuations make ϕ true.

3. In the 1-letter case, comparing upward or downward closures amounts to comparing the length of the shortest or longest word, respectively, accepted by the automata. This is easily done in nondeterministic logspace. And since $\uparrow L(A) = \downarrow L(A) = \varnothing$ if and only if $L(A) = \varnothing$, NL-hardness is shown by reduction from emptiness of NFAs, that is, a question "is there a path from an initial state to an accepting state" which is just another version of GAP, the Graph Accessibility Problem. □

A special case of language comparison is testing for universality. The question whether $\uparrow L(A) = \Sigma^*$ is trivial since it amounts to asking whether ε is accepted by A. For downward closures one has the following:

Proposition 7.4 (after [42]) *Deciding whether $\downarrow L(A) = \Sigma^*$ when A is an NFA over Σ is* NL*-complete.*

PROOF. Rampersad *et al.* show that the problem can be solved in linear time [42, Section 4.4]. Actually the characterization they use, namely "$\downarrow L(A) = \Sigma^*$ if and only if $A = (\Sigma, Q, \delta, I, F)$ has a state $q \in Q$ with $I \xrightarrow{*} q \xrightarrow{*} F$ and such that for any $a \in \Sigma$ there is a path of the form $q \xrightarrow{*} \xrightarrow{a} \xrightarrow{*} q$ from q to itself", is a FO + TC sentence on A seen as a directed labeled graph, hence can be checked in NL [28]. NL-hardness can be shown by reduction from emptiness of NFAs, for example, by adding loops $p \xrightarrow{a} p$ on any accepting state $p \in F$ and for every $a \in \Sigma$. □

8. Concluding remarks

We considered the state complexity of "closure languages" obtained by starting from an arbitrary regular language and closing it with all its subwords or all its superwords. These closure operations are essential when reasoning with subwords [32]. We completed the known results on closures by providing exact state complexities in the case of unbounded alphabets, and by demonstrating an

exponential lower bound on downward closures even in the case of a two-letter alphabet.

We also considered the dual notion of computing interiors. The nondeterministic state complexity of interiors is a new problem that we introduced in this article and for which we show doubly-exponential upper and lower bounds. From this we can deduce an exponential state complexity for the upward and downward closures of languages represented via alternating automata.

These results contribute to a more general research agenda: what are the "best" data structures and algorithms for reasoning with subwords and superwords? The algorithmics of subwords and superwords has mainly been developed in string matching and combinatorics [4, 15]. When considering subwords and superwords for sets of strings rather than individual strings – in matching and combinatorics [45] but also in other fields like model-checking and constraint solving [27, 32] –, there are many different ways of representing downward-closed and upward-closed sets. Automata-based representation are not always the preferred option; see, for example, the SREs used for downward-closed languages in [1]. The existing trade-offs between all the available options are not yet well understood and certainly deserve more scrutiny. In this direction, let us mention [7, Theorem 2.1(3)] showing that if $n_{\mathrm{D}}(L) = n$ then $\min(L) \stackrel{\text{def}}{=} \{x \in L \mid \forall y \in L : y \sqsubseteq x \implies y = x\} = L \smallsetminus (L \sqcup \Sigma)$ may have $n_{\mathrm{N}}(\min(L)) = (n-2)2^{n-3} + 2$, to be contrasted with $n_{\mathrm{N}}(\uparrow L) \leqslant n$. This suggests that it is more efficient to represent $\uparrow L$ directly than by its minimal elements.

Acknowledgments.

References

[1] P. A. Abdulla, A. Collomb-Annichini, A. Bouajjani, and B. Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65, 2004.

[2] M. Arfi. Polynomial operations on rational languages. In *Proc. STACS '87*, volume 247 of *Lecture Notes in Computer Science*, pages 198–206. Springer, 1987.

[3] G. Bachmeier, M. Luttenberger, and M. Schlund. Finite automata for the sub- and superword closure of CFLs: Descriptional and computational complexity. In *Proc. LATA 2015*, volume 8977 of *Lecture Notes in Computer Science*, pages 473–485. Springer, 2015.

[4] R. A. Baeza-Yates. Searching subsequences. *Theoretical Computer Science*, 78(2):363–376, 1991.

[5] N. Bertrand and Ph. Schnoebelen. Computable fixpoints in well-structured symbolic model checking. *Formal Methods in System Design*, 43(2):233–267, 2013.

[6] M. P. Bianchi, M. Holzer, S. Jakobi, C. Mereghetti, B. Palano, and G. Pighizzini. On inverse operations and their descriptional complexity. *Journal of Automata, Languages and Combinatorics*, 17(2–4):61–81, 2012.

[7] J.-C. Birget. Partial orders on words, minimal elements of regular languages and state complexity. *Theoretical Computer Science*, 119(2):267–291, 1993.

[8] J.-C. Birget. The state complexity of $\overline{\Sigma^* \overline{L}}$ and its connection with temporal logic. *Information Processing Letters*, 58(4):185–188, 1996.

[9] J. A. Brzozowski, G. Jirásková, and Baiyu Li. Quotient complexity of ideal languages. *Theoretical Computer Science*, 470:36–52, 2013.

[10] J. A. Brzozowski, J. Shallit, and Zhi Xu. Decision problems for convex languages. *Information and Computation*, 209(3):353–367, 2011.

[11] A. K. Chandra and L. J. Stockmeyer. Alternation. In *Proc. FOCS '76*, pages 98–108. IEEE Comp. Soc. Press, 1976.

[12] Th. Colcombet. Unambiguity in automata theory. In *Proc. DCFS 2015*, volume 9118 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015.

[13] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, London, UK, 1971.

[14] B. Courcelle. On constructing obstruction sets of words. *EATCS Bulletin*, 44:178–185, 1991.

[15] C. H. Elzinga, S. Rahmann, and Hui Wang. Algorithms for subsequence combinatorics. *Theoretical Computer Science*, 409(3):394–404, 2008.

[16] Jie Fu, J. Heinz, and H. G. Tanner. An algebraic characterization of strictly piecewise languages. In *Proc. TAMC 2011*, volume 6048 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2011.

[17] H. Gruber and M. Holzer. Finding lower bounds for nondeterministic state complexity is hard. In *Proc. DLT 2006*, volume 4036 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2006.

[18] H. Gruber, M. Holzer, and M. Kutrib. The size of Higman-Haines sets. *Theoretical Computer Science*, 387(2):167–176, 2007.

[19] H. Gruber, M. Holzer, and M. Kutrib. More on the size of Higman-Haines sets: Effective constructions. *Fundamenta Informaticae*, 91(1):105–121, 2009.

[20] Ch. Haase, S. Schmitz, and Ph. Schnoebelen. The power of priority channel systems. *Logical Methods in Comp. Science*, 10(4:4), 2014.

[21] P. Habermehl, R. Meyer, and H. Wimmel. The downward-closure of Petri net languages. In *Proc. ICALP 2010*, volume 6199 of *Lecture Notes in Computer Science*, pages 466–477. Springer, 2010.

[22] L. H. Haines. On free monoids partially ordered by embedding. *Journal of Combinatorial Theory*, 6(1):94–98, 1969.

[23] P.-C. Héam. On shuffle ideals. *RAIRO Theoretical Informatics and Applications*, 36(4):359–384, 2002.

[24] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc. (3)*, 2(7):326–336, 1952.

[25] Š. Holub, T. Masopust, and M. Thomazo. Alternating towers and piecewise testable separators. arXiv:1409.3943 [cs.FL], September 2014.

[26] M. Holzer and M. Kutrib. Nondeterministic descriptional complexity of regular languages. *Int. J. Foundations of Computer Science*, 14(6):1087–1102, 2003.

[27] P. Hooimeijer and M. Veanes. An evaluation of automata algorithms for string analysis. In *Proc. VMCAI 2011*, volume 6538 of *Lecture Notes in Computer Science*, pages 248–262. Springer, 2011.

[28] N. Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16(4):760–778, 1987.

[29] M. Ito, L. Kari, and G. Thierrin. Shuffle and scattered deletion closure of languages. *Theoretical Computer Science*, 245(1):115–133, 2000.

[30] J. Kahn. Entropy, independent sets and antichains: A new approach to Dedekind's problem. *Proc. Amer. Math. Soc.*, 130(2):371–378, 2002.

[31] Jui-Yi Kao, N. Rampersad, and J. Shallit. On NFAs where all states are final, initial, or both. *Theoretical Computer Science*, 410(47–49):5010–5021, 2009.

[32] P. Karandikar and Ph. Schnoebelen. Decidability in the logic of subsequences and supersequences. In *Proc. FST&TCS 2015*, Leibniz International Proceedings in Informatics. Leibniz-Zentrum für Informatik, December 2015. To appear.

[33] L. Kari, G. Paun, G. Thierrin, and Sheng Yu. At the crossroads of DNA computing and formal languages: Characterizing RE using insertion-deletion systems. In *DNA Based Computers III*, volume 48 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 329–347. American Mathematical Society, 1999.

[34] D. Kleitman. On Dedekind's problem: The number of monotone Boolean functions. *Proc. Amer. Math. Soc.*, 21(3):677–682, 1969.

[35] J. van Leeuwen. Effective constructions in well-partially-ordered free monoids. *Discrete Mathematics*, 21(3):237–252, 1978.

[36] Hing Leung. Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. *SIAM J. Computing*, 27(4):1073–1082, 1998.

[37] Hing Leung. Descriptional complexity of NFA of different ambiguity. *Int. J. Foundations of Computer Science*, 16(5):975–984, 2005.

[38] S. Lombardy and J. Sakarovitch. The universal automaton. In J. Flum, E. Grädel, and T. Wilke, editors, *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 457–504. Amsterdam University Press, 2008.

[39] A. Okhotin. On the state complexity of scattered substrings and superstrings. *Fundamenta Informaticae*, 99(3):325–338, 2010.

[40] G. Paun. *Marcus Contextual Grammars*, volume 67 of *Studies in Linguistics and Philosophy*. Springer, 1997.

[41] J.-É. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory of Computing Systems*, 30(4):383–422, 1997.

[42] N. Rampersad, J. Shallit, and Zhi Xu. The computational complexity of universality problems for prefixes, suffixes, factors, and subwords of regular languages. *Fundamenta Informaticae*, 116(1–4):223–236, 2012.

[43] J. Rogers, J. Heinz, G. Bailey, M. Edlefsen, M. Visscher, D. Wellcome, and S. Wibel. On languages piecewise testable in the strict sense. In *Proc. MOL 2010*, volume 6149 of *Lecture Notes in Computer Science*, pages 255–265. Springer, 2010.

[44] A. Salomaa, K. Salomaa, and Sheng Yu. State complexity of combined operations. *Theoretical Computer Science*, 383(2–3):140–152, 2007.

[45] Z. Tronícek and A. Shinohara. The size of subsequence automaton. *Theoretical Computer Science*, 341(1–3):379–384, 2005.

[46] Sheng Yu. State complexity: Recent results and open problems. *Fundamenta Informaticae*, 64(1–4):471–480, 2005.

[47] G. Zetzsche. Computing downward closures for stacked counter automata. In *Proc. STACS 2015*, volume 30 of *Leibniz International Proceedings in Informatics*, pages 743–756. Leibniz-Zentrum für Informatik, 2015.