Regular Sensing

Shaull Almagor*, Denis Kuperberg†, and Orna Kupferman*
*The Hebrew University, School of Computer Science and Engineering.
†The University of Warsaw, Faculty of Mathematics, Informatics, and Mechanics.

***Abstract*—The size of deterministic automata required for recognizing regular and ω-regular languages is a well-studied measure for the complexity of languages. We introduce and study a new complexity measure, based on the *sensing* required for recognizing the language. Intuitively, the sensing cost quantifies the detail in which a random input word has to be read in order to decide its membership in the language. Technically, we consider languages over an alphabet 2^P, for a finite set P of signals. A signal $p \in P$ is sensed in a state of the automaton if transitions from the state depend on its value. The *sensing cost of an automaton* is then its expected sensing, under a uniform distribution of the alphabet, and the *sensing cost of a language* is the infimum of the sensing costs of deterministic automata for the language. Beyond the theoretical interest in regular sensing, it corresponds to natural and practical questions in the design of finite-state monitors, as well as controllers and transducers.**

We show that for finite words, size and sensing are related, and minimal sensing is attained by minimal automata. Thus, a unique minimal-sensing deterministic automaton exists, and is based on the language's right-congruence relation. For infinite words, the minimal sensing may be the limit of an infinite sequence of automata. We show that the unique limit of such sequences can be characterized by the language's right-congruence relation, which enables us to find the sensing cost of ω-regular languages in polynomial time. Also, interestingly, the sensing cost is independent of the acceptance condition. This is in contrast with the size measure, where the size of a minimal deterministic automaton for an ω-regular language depends on the acceptance condition, a unique minimal automaton need not exists, and the problem of finding one is NP-complete. We also study the affect of standard operations (e.g., union, concatenation, etc.) on the sensing cost of automata and languages.

I. Introduction

Studying the complexity of a formal language, there are several complexity measures to consider. When the language is given by means of a Turing Machine, the traditional measures are time and space demands. Theoretical interest as well as practical considerations have motivated additional measures, such as randomness (the number of random bits required for the execution) [11] or communication complexity (number and length of messages required) [10]. For regular and ω-regular languages, given by means of finite-state automata, the classical complexity measure is the size of a minimal deterministic automaton that recognizes the language.

We introduce and study a new complexity measure, namely the *sensing cost* of the language. Intuitively, the sensing cost of a language measures the detail with which a random input word needs to be read in order to decide membership in the language. Sensing has been studied in several other CS contexts. In theoretical CS, in methodologies such as PCP and property testing, we are allowed to sample or query only part of the input [8]. In more practical applications, mathematical tools in signal processing are used to reconstruct information based on compressed sensing [6], and in the context of data streaming, one cannot store in memory the entire input, and therefore has to approximate its properties according to partial "sketches" [12].

Our interest in regular sensing is motivated by the use of finite-state automata (as well as monitors, controllers, and transducers) in reasoning about on-going behaviors of reactive systems. In particular, a big challenge in the design of monitors is an optimization of the sensing needed for deciding the correctness of observed behaviors. Our goal is to formalize regular sensing in the finite-state setting and to study the sensing complexity measure for regular and ω-regular languages.

A natural setting in which sensing arises is *synthesis*: given a specification over sets I and O of input and output signals, the goal is to construct a finite-state system that, given a sequence of input signals, generates a computation that satisfies the specification. In each moment in time, the system reads an assignment to the input signals, namely a letter in 2^I, which requires the activation of $|I|$ Boolean sensors. A well-studied special case of limited sensing is synthesis with *incomplete information*. There, the system can read only a subset of the signals in I, and should still generate only computations that satisfy the specification [9], [4]. A more sophisticated case of sensing in the context of synthesis is studied in [5], where the system can read some of the input signals some of the time. In more detail, sensing the truth value of an input signal has a cost, the system has a budget for sensing, and it tries to realize the specification while minimizing the required sensing budget. In [17], the authors study games with errors. Such games correspond to a synthesis setting in which there are positions during the interaction in which input signals are read by the system with an error. The games are characterized by the number or rate of errors that the system has to cope with, and by the ability of the system to detect whether a current input is erred. Finally, [2] studies a setting in which the truth values of the specification as well as the input signals are multi-valued. The signals are sensed with some noise and the goal is to synthesize a system that realizes the specification with the best value possible despite the noise.

We study the basic fundamental questions on regular sensing. We consider languages over alphabets of the form 2^P, for a finite set P of signals. Consider a deterministic automaton $\mathcal{A}$ over an alphabet 2^P. For a state q of $\mathcal{A}$, we say that a signal $p \in P$ is *sensed* in q if at least one transition taken from q depends on the truth value of p. The *sensing cost* of q is the number of signals it senses, and the sensing cost of a run is the average sensing cost of states visited along the run. We extend the definition to automata by assuming a uniform distribution of the inputs. Thus, the sensing cost of $\mathcal{A}$ is the limit of the expected sensing of runs over words of increasing length.[1] We show that this definition coincides with one that is based on the stationary distribution of the Markov chain induced by $\mathcal{A}$, which enables us to calculate the sensing cost of an automaton in polynomial time. The sensing cost of a language L, of either finite or infinite words, is then the infimum of the sensing costs of deterministic automata for L. In the case of infinite words, one can study different classes of automata, yet we show that the sensing cost is independent of the acceptance condition being used.

We start by studying the sensing cost of regular languages of finite words. For the complexity measure of size, the picture in the setting of finite words is very clean: each language L has a unique minimal deterministic automaton (DFA), namely the *residual automaton* $\mathcal{R}_L$ whose states correspond to the equivalence classes of the Myhill-Nerode right-congruence relation for L [13], [14]. We show that minimizing the state space of a DFA can only reduce its sensing cost. Hence, the clean picture of the size measure is carried over to the sensing measure: the sensing cost of a language L is attained in the DFA $\mathcal{R}_L$. In particular, since DFAs can be minimized in polynomial time, we can construct in polynomial time a minimally-sensing DFA, and can compute in polynomial time the sensing cost of languages given by DFAs. We proceed to study how classical operations (complementation, union, and concatenation) affect the sensing cost of languages, and we show that sensing is not monotonic with respect to abstractions: a language L' that abstracts a language L (that is, L' is obtained from L by existentially projecting some signals in some transitions in a DFA for L) may have a strictly bigger sensing cost.

We turn on to study the sensing cost of ω-regular languages, given by means of deterministic parity automata (DPAs). Recall the size complexity measure. There, the picture for languages of infinite words is not clean: A language needs not have a unique minimal DPA, and the problem of finding one is NP-complete [16]. It turns out that the situation is challenging also in the sensing measure. First, we show that different minimal DPAs for a language may have different sensing costs. In fact, bigger DPAs may have smaller sensing cost. Moreover, the sensing cost of a language may not be attained, and may only be the limit of a sequence of DPWs. To see this, consider for example the language $L \subseteq (2^{\{p\}})^\omega$ of all words in which p holds in infinitely many positions. For $n \geq 2$, an n-state DPA $\mathcal{A}_n$ for L may follow a "lazy sensing" strategy in which p is ignored for $n-1$ states and is being waited for only in the n-th state. As n tends to infinity, $\mathcal{A}_n$ is expected to spend more time in the chain of $n+1$ states where no sensors are required, so the sensing cost of $\mathcal{A}_n$ tends to 0. Thus, bigger DPAs have smaller sensing costs, and the sensing cost of L is obtained as the infimum of the sensing cost of the DPAs $(\mathcal{A}_n)_{n=2}^\infty$.

Our main result is that despite the above intricacy, we can compute the sensing cost of an ω-regular language in polynomial time. Indeed, we prove that the sensing cost of an ω-regular language L is the sensing cost of the residual automaton $\mathcal{R}_L$ for L. Unlike the case of finite words, it may not be possible to define L on top of $\mathcal{R}_L$. Interestingly, however, $\mathcal{R}_L$ does capture exactly the sensing required for recognizing L. The proof of this property of $\mathcal{R}_L$ is the main technical challenge of our contribution. The proof goes via a sequence $(\mathcal{B}_n)_{n=1}^\infty$ of DPWs whose sensing costs converge to that of L. The DPA $\mathcal{B}_n$ is obtained from a DPA $\mathcal{A}$ for L by a lazy sensing strategy that spends time is n copies of $\mathcal{R}_L$ between visits to $\mathcal{A}$. As n grows, less time is expected to be spent in $\mathcal{A}$, so $\mathcal{R}_L$ becomes the dominant component in the sensing cost of $\mathcal{B}_n$. The challenge is to define the visits in $\mathcal{A}$ so that even being sparse, it is possible to decide acceptance from the parity ranks visited during them. This is made possible by decomposing $\mathcal{A}$ according to its ergodic strongly connected components and strengthening the "flower lemma" of [15], exploring its consequences in the special case of strongly connected automata.

II. Preliminaries

A. Automata

A *deterministic automaton on finite words* (DFA, for short) is $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where Q is a set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to Q$ is a transition function, and $\alpha \subseteq Q$ is a set of accepting states. We sometimes refer to δ as a relation $\Delta \subseteq Q \times \Sigma \times Q$, with $\langle q, \sigma, q' \rangle \in \Delta$ iff $\delta(q, \sigma) = q'$. The run of $\mathcal{A}$ on a word $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_m \in \Sigma^*$ is the sequence of states $q_0, q_1, \ldots, q_m$ such that $q_{i+1} \in \delta(q_i, \sigma_{i+1})$ for all $i \geq 0$. The run is accepting if $q_m \in \alpha$. A word $w \in \Sigma^*$ is accepted by $\mathcal{A}$ if the run of $\mathcal{A}$ on w is accepting. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. For a state $q \in Q$, we use $\mathcal{A}^q$ to denote $\mathcal{A}$ with initial state q. We sometimes refer also to nondeterministic automata (NFAs), where $\delta : Q \times \Sigma \to 2^Q$ suggests several possible successor states, and there may be several initial states. Thus, an NFA may have several runs on an input word w, and it accepts w if at least one of them is accepting.

Consider a language $L \subseteq \Sigma^*$. For two finite words u_1 and u_2, we say that u_1 and u_2 are *right L-indistinguishable*, denoted $u_1 \sim_L u_2$, if for every $z \in \Sigma^*$, we have that $u_1 \cdot z \in L$ iff $u_2 \cdot z \in L$. Thus, $\sim_L$ is the Myhill-Nerode right congruence used for minimizing automata. For $u \in \Sigma^*$, let $[u]$ denote the equivalence class of u in $\sim_L$ and let $\langle L \rangle$ denote the set of all equivalence classes. Each class $[u] \in \langle L \rangle$ is associated with the *residual language* $L^u = \{w : uw \in L\}$. When L

[1]Alternatively, one could define the sensing cost of $\mathcal{A}$ as the cost of its "most sensing" run. Such a worst-case approach is taken in [5], where the sensing cost needs to be kept under a certain budget in all computations, rather than in expectation. We find the average-case approach we follow appropriate for sensing, as the cost of operating sensors may well be amortized over different runs of the system, and requiring the budget to be kept under a threshold in every run may be too restrictive. Thus, the automaton must answer correctly for every word, but the sensing should be low only on average, and it is allowed to operate an expensive sensor now and then. Our results can be easily extended to any distribution on the inputs that is given by a finite-state Markov chain.

is regular, the set $\langle L\rangle$ is finite, and induces the *residual automaton* of L, defined by $\mathcal{R}_L = \langle \Sigma, \langle L\rangle, \Delta_L, [\epsilon], \alpha\rangle$, with $\langle [u], a, [u \cdot a]\rangle \in \Delta_L$ for all $[u] \in \langle L\rangle$ and $a \in \Sigma$. Also, α contains all classes $[u]$ with $u \in L$. By [13], [14], the DFA $\mathcal{R}_L$ is well defined and is the unique minimal DFA for L.

A *deterministic automaton on infinite words* is $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha\rangle$, where Q, q_0, and δ are as in DFA, and α is an acceptance condition. The run of $\mathcal{A}$ on an infinite input word $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is defined as for automata on finite words, except that the sequence of visited states is now infinite. For a run $r = q_0, q_1, \ldots$, let $\mathit{inf}(r)$ denote the set of states that r visits infinitely often. Formally, $\mathit{inf}(r) = \{q : q = q_i \text{ for infinitely many } i\text{'s}\}$. We consider the following acceptance conditions. In a *Büchi* automaton, the acceptance condition is a set $\alpha \subseteq Q$ and a run r is accepting iff $\mathit{inf}(r) \cap \alpha \neq \emptyset$. Dually, in a *co-Büchi*, again $\alpha \subseteq Q$, but r is accepting iff $\mathit{inf}(r) \cap \alpha = \emptyset$. Finally, parity condition is a mapping $\alpha : Q \to [i, \ldots, j]$, for integers $i \leq j$, and a run r is accepting iff $\max_{q \in \mathit{inf}(r)}\{\alpha(q)\}$ is even. We use the acronyms NBA, DBA, NCA, DCA, NPA, and DPA to denote nondeterministic/deterministic Büchi/co-Büchi/parity word automata.

We extend the right congruence $\sim_L$ as well as the deftitnion of the residual automaton $\mathcal{R}_L$ to languages $L \subseteq \Sigma^\omega$. Here, however, $\mathcal{R}_L$ need not accept the language of L, and we ignore its acceptance condition.

B. Sensing

We study languages over an alphabet $\Sigma = 2^P$, for a finite set P of signals. A letter $\sigma \in \Sigma$ corresponds to a truth assignment to the signals. When we define languages over Σ, we use predicates on P in order to denote sets of letters. For example, if $P = \{a, b, c\}$, then the expression $(\texttt{True})^* \cdot a \cdot b \cdot (\texttt{True})^*$ describes all words over 2^P that contain a subword $\sigma_a \cdot \sigma_b$ with $\sigma_a \in \{\{a\}, \{a, b\}, \{a, c\}, \{a, b, c\}\}$ and $\sigma_b \in \{\{b\}, \{a, b\}, \{b, c\}, \{a, b, c\}\}$.

Consider an automaton $\mathcal{A} = \langle 2^P, Q, q_0, \delta, \alpha\rangle$. For a state $q \in Q$ and a signal $p \in P$, we say that p is *sensed in* q if there exists a set $S \subseteq P$ such that $\delta(q, S \setminus \{p\}) \neq \delta(q, S \cup \{p\})$. Intuitively, a signal is sensed in q if knowing its value may affect the destination of at least one transition from q. We use $\mathit{sensed}(q)$ to denote the set of signals sensed in q. The *sensing cost* of a state $q \in Q$ is $\mathit{scost}(q) = |\mathit{sensed}(q)|$. [2]

Consider a deterministic automaton $\mathcal{A}$ over $\Sigma = 2^P$ (and over finite or infinite words). For a finite run $r = q_0, \ldots, q_m$ of $\mathcal{A}$, we define the sensing cost of r, denoted $\mathit{scost}(r)$, as $\frac{1}{m}\sum_{i=0}^{m-1} \mathit{scost}(q_i)$. That is, $\mathit{scost}(r)$ is the average number of sensors that $\mathcal{A}$ uses during r. Now, for a finite word w, we define the sensing cost of w in $\mathcal{A}$, denoted $\mathit{scost}_{\mathcal{A}}(w)$, as the sensing cost of the run of $\mathcal{A}$ on w. Finally, the sensing cost of $\mathcal{A}$ is the expected sensing cost of words of length that tends to infinity, where we assume that the letters in Σ are uniformly distributed. Thus, $\mathit{scost}(\mathcal{A}) = \lim_{m\to\infty} |\Sigma|^{-m} \sum_{w:|w|=m} \mathit{scost}_{\mathcal{A}}(w)$. Note that the definition applies to automata on both finite and infinite words.

Two DFAs may recognize the same language and have different sensing costs. For example, if $P = \{a\}$, one DFA for $\texttt{True}^*$ may have a single state and sensing cost 0, while another DFA may decompose the language to $\texttt{True}^* \cdot a + \texttt{True}^* \cdot (\neg a)$, and thus need two states, both sensing a. In fact, as we demonstrate in Example 2.1 below, in the case of infinite words two different minimal automata for the same language may have different sensing costs.

For a language L of finite or infinite words, the sensing cost of L, denoted $\mathit{scost}(L)$ is the minimal sensing cost required for recognizing L by a deterministic automaton. Thus, $\mathit{scost}(L) = \inf_{\mathcal{A}:L(\mathcal{A})=L} \mathit{scost}(\mathcal{A})$. For the case of infinite words, we allow $\mathcal{A}$ to be a deterministic automaton of any type. In fact, as we shall see, unlike the case of succinctness, the sensing cost is independent of the acceptance condition used.

Example 2.1: Let $P = \{a\}$. Consider the language $L \subseteq (2^{\{a\}})^\omega$ of all words with infinitely many a and infinitely many $\neg a$. In Figure 1 we present two minimal DBAs for L with different sensing costs. While all the states of the second

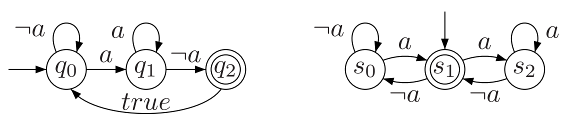

Fig. 1. Two minimal DBAs for L with different sensing costs.

automaton sense a, thus its sensing cost is 1, the signal a is not sensed in all the states of the first automaton, thus its sensing cost is strictly smaller than 1 (to be precise, it is $\frac{4}{5}$, as we shall see in Example 2.7). □

Remark 2.2: Our study of sensing considers deterministic automata. The notion of sensing is less natural in the nondeterministic setting. From a conceptual point of view, we want to capture the number of sensors required for an actual implementation for recognizing the language. Technically, guesses can reduce the number of required sensors. To see this, take $P = \{a\}$ and consider the language $L = \texttt{True}^* \cdot a$. A DFA for L needs two states, both sensing a. An NFA for L can guess the position of the letter before the last one, where it moves to the only state that senses a. The sensing cost of such an NFA is 0 (for any reasonable extension of the definition of cost on NFAs). □

C. Probability

Consider a directed graph $G = \langle V, E\rangle$. A *strongly connected component* (SCC) of G is a maximal (with respect to containment) set $C \subseteq V$ such that for all $x, y \in C$, there is a path from x to y. An SCC (or state) is *ergodic* if no other SCC is reachable from it, and is *transient* otherwise.

An automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha\rangle$ induces a directed graph $G_{\mathcal{A}} = \langle Q, E\rangle$ in which $\langle q, q'\rangle \in E$ iff there is a letter σ such that $q' \in \delta(q, \sigma)$. When we talk about the SCCs of $\mathcal{A}$, we refer to those of $G_{\mathcal{A}}$. Recall that we assume that the letters

[2] We note that, alternatively, one could define the *sensing level* of states, with $\mathit{slevel}(q) = \frac{|\mathit{sensed}(q)|}{|P|}$. Then, for all states q, we have that $\mathit{slevel}(q) \in [0, 1]$. All our results hold also for this definition, simply by dividing the sensing cost by $|P|$.

in Σ are uniformly distributed, thus $\mathcal{A}$ also corresponds to a Markov chain $M_{\mathcal{A}}$ in which the probability of a transition from state q to state q' is $p_{q,q'} = \frac{1}{|\Sigma|}|\{\sigma \in \Sigma : \delta(q,\sigma) = q'\}|$. Let $\mathcal{C}$ be the set of $\mathcal{A}$'s SCC, and $\mathcal{C}_e \subseteq \mathcal{C}$ be the set of its ergodic SCC's.

Consider an ergodic SCC $C \in \mathcal{C}_e$. Let P_C be the matrix describing the probability of transitions in C. Thus, the rows and columns of P_C are associated with states, and the value in coordinate q, q' is $p_{q,q'}$. By [7], there is a unique probability vector $\pi_C \in [0,1]^C$ such that $\pi_C P_C = \pi_C$. This vector describes the *stationary distribution* of C: for all $q \in C$ it holds that $\pi_C(q) = \lim_{m\to\infty} \frac{E^C_m(q)}{m}$, where $E^C_m(q)$ is the average number of occurrences of q in a run of M_A of length m that starts anywhere in C [7]. Thus, intuitively, $\pi_C(q)$ is the probability that a long run that starts in C ends in q. In order to extend the distribution to the entire Markov chain of $\mathcal{A}$, we have to take into account the probability of reaching each of the ergodic components. The *SCC-reachability distribution* of $\mathcal{A}$ is the function $\rho : \mathcal{C} \to [0,1]$ that maps each ergodic SCC C of $\mathcal{A}$ to the probability that M_A eventually reaches C, starting from the initial state. We can now define the *limiting distribution* $\pi : Q \to [0,1]$, as

$$\pi(q) = \begin{cases} 0 & \text{if } q \text{ is transient} \\ \pi_C(q)\rho(C) & \text{if } q \text{ is in some } C \in \mathcal{C}_e \end{cases}$$

Note that $\sum_{q\in Q} \pi(q) = 1$, and that if P is the matrix describing the transitions of M_A and π is viewed as a vector in $[0,1]^Q$, then $\pi P = \pi$. Intuitively, the limiting distribution of state q describes the probability of a run on a random and long input word to end in q. Formally, we have the following.

Lemma 2.3: Let $E_m(q)$ be the expected number of occurrences of a state q in a run of length m of M_A that starts in q_0. Then, $\pi(q) = \lim_{m\to\infty} \frac{E_m(q)}{m}$.

Proof: Let $q \in Q$, and consider a random infinite run r in M_A. If q is transient, then it is easy to see that $\lim_{m\to\infty} \frac{1}{m} E_m(q) = 0 = \pi(q)$, because with probability 1, the state q does not appear after some point in r. Otherwise, let $C \in \mathcal{C}_e$ be the ergodic SCC of q. The probability that r reaches C is given by $\rho(C)$. By the law of total expectation, and since q is reachable only if r reaches C, we have that $E_m(q) = \rho(C)E^C_{m-t}$ where t is the expected time by which r reaches C. From this we get that $\lim_{m\to\infty} \frac{E_m(q)}{m} = \rho(C)\lim_{m\to\infty} \frac{E^C_{m-t}}{m} = \rho(C)\lim_{m\to\infty} \frac{E^C_m}{m} = \rho(C)\pi_C(q)$. □

D. Computing The Sensing Cost of an Automaton

Consider a deterministic automaton $\mathcal{A} = \langle 2^P, Q, \delta, q_0, \alpha\rangle$. The definition of $scost(\mathcal{A})$ by means of the expected sensing cost of words of length that tends to infinity does not suggest an algorithm for computing it. In this section we show that the definition coincides with a definition that sums the costs of the states in $\mathcal{A}$, weighted according to the limiting distribution, and show that this implies a polynomial-time algorithm for computing $scost(\mathcal{A})$. This also shows that the cost is well-defined for all automata.

Theorem 2.4: For all automata $\mathcal{A}$, we have $scost(\mathcal{A}) = \sum_{q\in Q} \pi(q) \cdot scost(q)$, where π is the limiting distribution of $\mathcal{A}$.

Proof: By Lemma 2.3, we have $\pi(q) = \lim_{m\to\infty} \frac{E_m(q)}{m}$, where $E_m(q)$ is the expected number of occurrences of q in a random m-step run. This can be restated in our case as $\pi(q) = \lim_{m\to\infty} \frac{1}{m|\Sigma|^m} \sum_{w:|w|=m} Occ_w(q)$, where $Occ_w(q)$ is the number of occurrences of q in the run of $\mathcal{A}$ on w. By definition, $scost(\mathcal{A}) = \lim_{m\to\infty} |\Sigma|^{-m} \sum_{w:|w|=m} scost_{\mathcal{A}}(w)$, and also $scost_{\mathcal{A}}(w) = \sum_{q\in Q} scost(q) \cdot Occ_w(q)$. From this, we get

$$\begin{aligned} scost(\mathcal{A}) &= \lim_{m\to\infty} |\Sigma|^{-m} \sum_{w:|w|=m} \sum_{q\in Q} scost(q) \cdot Occ_w(q) \\ &= \sum_{q\in Q} scost(q) \cdot \lim_{m\to\infty} |\Sigma|^{-m} \sum_{w:|w|=m} Occ_w(q) \\ &= \sum_{q\in Q} scost(q) \cdot \pi(q). \end{aligned}$$

□

Remark 2.5: It is not hard to see that if $\mathcal{A}$ is strongly connected, then π is the unique stationary distribution of M_A and is independent of the initial state of $\mathcal{A}$. Accordingly, $scost(\mathcal{A})$ is also independent of $\mathcal{A}$'s initial state in this special case. □

Theorem 2.6: Given an automaton $\mathcal{A}$, the sensing cost $scost(\mathcal{A})$ can be calculated in polynomial time.

Proof: By Theorem 2.4, we have that $scost(\mathcal{A}) = \sum_{q\in Q} \pi(q) \cdot scost(q)$, where π is the limiting distribution of $\mathcal{A}$. By the definition of π, we have that $\pi(q) = \pi_C(q)\rho(C)$, if q is in some $C \in \mathcal{C}_e$. Otherwise, $\pi(q) = 0$. Hence, the computational bottleneck is the calculation of the SCC-reachability distribution $\rho : C \to [0,1]$ and the stationary distributions π_C for every $C \in \mathcal{C}_e$.It is well known that both can be computed in polynomial time via classic algorithms on matrices. For completeness, we give the details in Appendix A. For completeness, we give the details in Appendix A. □

Example 2.7: Recall the first DBAs described in Example 2.1. Its limiting distribution is $\pi(q_0) = \pi(q_1) = \frac{2}{5}$, $\pi(q_2) = \frac{1}{5}$. Accordingly, its cost is $1 \cdot \frac{2}{5} + 1 \cdot \frac{2}{5} + 0 \cdot \frac{1}{5} = \frac{4}{5}$.

Let $P = \{a, b\}$. Consider the DFA $\mathcal{A}_1$ appearing in Figure 2. Note that $L(\mathcal{A}_1) = (\mathtt{True})^* \cdot a \cdot b \cdot (\mathtt{True})^*$. It is easy to see that $sensed(q_0) = \{a\}$, $sensed(q_1) = \{b\}$, and $sensed(q_2) = \emptyset$. Accordingly, $scost(q_0) = scost(q_1) = 1$ and $scost(q_2) = 0$. Since the state q_2 forms the only ergodic SCC, the limiting distribution on the states of $\mathcal{A}$ is $\pi(q_0) = \pi(q_1) = 0$ and $\pi(q_2) = 1$. Hence, $scost(\mathcal{A}_1) = 0$.

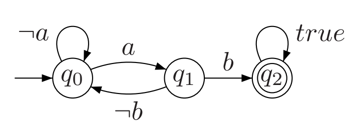

Fig. 2. The DFA $\mathcal{A}_1$.

Consider now the DFA $\mathcal{A}_2$, appearing in Figure 3, with $L(\mathcal{A}_2) = (\texttt{True})^* \cdot a \cdot b$. Here, $sensed(q_0) = \{a\}$, $sensed(q_1) = \{a, b\}$, and $sensed(q_2) = \{a\}$. Accordingly, $scost(q_0) = scost(q_2) = 1$ and $scost(q_2) = 2$. Since $\mathcal{A}_2$ is strongly connected, its limiting distribution is its unique stationary distribution, which can be calculated by solving the following system of equations, where x_i corresponds to $\pi(q_i)$:

- $x_0 = \frac{1}{2}x_0 + \frac{1}{4}x_1 + \frac{1}{2}x_2$.
- $x_2 = \frac{1}{2}x_1$.
- $x_1 = \frac{1}{2}x_0 + \frac{1}{4}x_1 + \frac{1}{2}x_2$.
- $x_0 + x_1 + x_2 = 1$.

Accordingly, $\pi(q_0) = \pi(q_1) = \frac{2}{5}$ and $\pi(q_2) = \frac{1}{5}$. We conclude that the sensing cost of $\mathcal{A}_2$ is $1 \cdot \frac{2}{5} + 2 \cdot \frac{2}{5} + 1 \cdot \frac{1}{5} = \frac{7}{5}$.

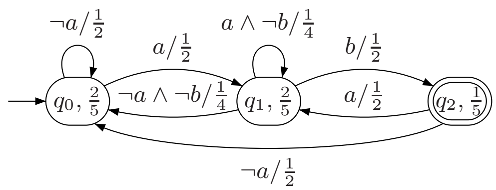

Fig. 3. The DFA $\mathcal{A}_2$ and its corresponding Markov chain.

□

III. The Sensing Cost of Regular Languages of Finite Words

In this section we study the setting of finite words. We show that there, sensing minimization goes with size minimization, which makes things clean and simple, as size minimization for DFAs is a feasible and well-studied problem. We also study theoretical properties of sensing. We show that, surprisingly, abstraction of signals may actually increase the sensing cost of a language, and we study the effect of classical operations on regular languages on their sensing cost.

A. Minimizing the Sensing Cost

Consider a regular language $L \subseteq \Sigma^*$, with $\Sigma = 2^P$. Recall that the residual automaton $\mathcal{R}_L = \langle \Sigma, \langle L \rangle, \Delta_L, [\epsilon], \alpha \rangle$ is the minimal-size DFA that recognizes L. We claim that $\mathcal{R}_L$ also minimizes the sensing cost of L.

Lemma 3.1: Consider a regular language $L \subseteq \Sigma^*$. For every DFA $\mathcal{A}$ with $L(\mathcal{A}) = L$, we have that $scost(\mathcal{A}) \geq scost(\mathcal{R}_L)$.

Proof: Consider a word $u \in \Sigma^*$. After reading u, the DFA $\mathcal{R}_L$ reaches the state $[u]$ and the DFA $\mathcal{A}$ reaches a state q with $L(\mathcal{A}^q) = L^u$. Indeed, otherwise we can point to a word with prefix u that is accepted only in one of the DFAs. We claim that for every state $q \in Q$ such that $L(\mathcal{A}^q) = L^u$, it holds that $sensed([u]) \subseteq sensed(q)$. To see this, consider a signal $p \in sensed([u])$. By definition, there exists a set $S \subseteq P$ and states u_1 andu_2 such that $([u], S \setminus \{p\}, [u_1]) \in \Delta_L$, $([u], S \cup \{p\}, [u_2]) \in \Delta_L$, yet $[u_1] \neq [u_2]$. By the definition of $\mathcal{R}_L$, there exists $z \in (2^P)^*$ such that, w.l.o.g, $z \in L^{u_1} \setminus L^{u_2}$. Hence, as $L(\mathcal{A}^q) = L^u$, we have that $\mathcal{A}^q$ accepts $(S \setminus \{p\}) \cdot z$ and rejects $(S \cup \{p\}) \cdot z$. Let $\delta_\mathcal{A}$ be the transition function of $\mathcal{A}$. By the above, $\delta_\mathcal{A}(q, S \setminus \{p\}) \neq \delta_\mathcal{A}(q, S \cup \{p\})$. Therefore, $p \in sensed(q)$, and we are done. Now, $sensed([u]) \subseteq sensed(q)$ implies that $scost(q) \geq scost([u])$.

Consider a word $w_1 \cdots w_m \in \Sigma^*$. Let $r = r_0, \ldots, r_m$ and $[u_0], \ldots, [u_m]$ be the runs of $\mathcal{A}$ and $\mathcal{R}_L$ on w, respectively. Note that for all $i \geq 0$, we have $u_i = w_1 \cdot w_2 \cdots w_i$. For all $i \geq 0$, we have that $L(\mathcal{A}^{r_i}) = L^{[u_i]}$, implying that then $scost(r_i) \geq scost([u_i])$. Hence, $scost_\mathcal{A}(w) \geq scost_{\mathcal{R}_L}(w)$. Since this holds for every word in Σ^*, it follows that $scost(\mathcal{A}) \geq scost(\mathcal{R}_L)$. □

Since $L(\mathcal{R}_L) = L$, then $scost(L) \leq scost(\mathcal{R}_L)$. This, together with Lemma 3.1, enables us to conclude with the following.

Theorem 3.2: For every regular language $L \subseteq \Sigma^*$, we have $scost(L) = scost(\mathcal{R}_L)$.

Finally, since DFAs can be size-minimized in polynomial time, Theorems 2.6 and 3.2 imply we can efficiently minimize also the sensing cost of a DFA and calculate the sensing cost of its language:

Theorem 3.3: Given an DFA $\mathcal{A}$, the problem of computing $scost(L(\mathcal{A}))$ can be solved in polynomial time.

B. On Monotonicity of Sensing

The example in Remark 2.2 suggests that there is a trade-off between guessing and sensing. Consider a DFA $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, with $\Sigma = 2^P$. For a state $q \in Q$ and a signal $p \in P$, let $\mathcal{A}_{q \downarrow p}$ be the NFA obtained from $\mathcal{A}$ by ignoring p in q. Thus, in state q, the NFA $\mathcal{A}_{q \downarrow p}$ guesses the value of p and proceeds to all the successors that are reachable with some value. Note that the guess introduces nondeterminism. Formally, $\mathcal{A}_{q \downarrow p} = \langle \Sigma, 2^Q, \{q_0\}, \delta', \alpha' \rangle$, where for every state $T \in 2^Q$ and letter $S \in 2^P$, we define $\delta(T, S) = \bigcup_{t \in T} \delta(t, S)$ if $q \notin T$, and $\delta(T, S) = \delta(q, S \setminus \{p\}) \cup \delta(q, S \cup \{p\}) \cup \bigcup_{t \in T \setminus \{q\}} \delta(t, S)$ if $q \in T$. Also, a state $T \subseteq Q$ is in α' iff $T \cap \alpha \neq \emptyset$. It is easy to see that $L(\mathcal{A}) \subseteq L(\mathcal{A}_{q \downarrow p})$. Since $\mathcal{A}_{q \downarrow p}$ is obtained from $\mathcal{A}$ by giving up some of its sensing, it is tempting to think that $scost(L(\mathcal{A}_{q \downarrow p})) \leq scost(L(\mathcal{A}))$. As we now show, however, sensing is not monotone. For two languages L and L', we say that L' is an abstraction of L if there is a DFA $\mathcal{A}$ such that $L(\mathcal{A}) = L$ and there is a state q and a signal p of $\mathcal{A}$ such that $L' = L(\mathcal{A}_{q \downarrow p})$.

Theorem 3.4: Sensing is not monotone. That is, there is a language L and an abstraction L' of L such that $scost(L) \leq scost(L')$.

Proof: Let $P = \{a, b, c\}$. Consider the language $L = a \cdot \texttt{True}^* \cdot b + (\neg a) \cdot \texttt{True}^* \cdot c$. It is not hard to see that $scost(L) = 1$. Indeed, a DFA for L has to sense a in its initial state and then has to always sense either b (in case a appears in the first letter) or c (otherwise).

Giving up the sensing of a in the initial state of a DFA for L we end up with the language $L' = (\texttt{True})^+ \cdot (b \vee c)$. It is not hard to see that $scost(L') = 2$. Indeed, every DFA for L' has to almost always sense both b and c. □

We conclude that replacing a sensor with non-determinism may actually result in a language for which we need more sensors. This corresponds to the known fact that abstraction of automata may result in bigger (in fact, exponentially bigger)

DFAs [3]. Also, while the above assumes an abstraction that over-approximates the original language, a dual argument could show that under-approximating the language (that is, defining $\mathcal{A}_{q\downarrow p}$ as a universal automaton) may result in a language with higher sensing cost.

C. Operations on DFAs and Their Sensing Cost

In this section we study the effect of actions on DFAs on their sensing cost. We start with complementation. For every regular language L, a DFA for $comp(L) = \Sigma^* \setminus L$ can be obtained from a DFA for L by only dualyzing the set of accepting states. In particular, this holds for $\mathcal{R}_L$, implying the following.

Lemma 3.5: For every regular language L we have that $scost(L) = scost(comp(L))$.

Next, we consider the union of two regular languages.

Lemma 3.6: For every regular languages $L_1, L_2 \subseteq (2^P)^*$, we have $scost(L_1 \cup L_2) \leq scost(L_1) + scost(L_2)$.

Proof: Consider the minimal DFAs $\mathcal{A}_1 = \langle 2^P, Q^1, \delta^1, q_0^1, \alpha^1\rangle$ and $\mathcal{A}_2 = \langle 2^P, Q^2, \delta^2, q_0^2, \alpha^2\rangle$ for L_1 and L_2, respectively. Let $\mathcal{B} = \langle 2^P, Q^1 \times Q^2, \delta, (q_0^1, q_0^2), (\alpha^1 \times Q^2) \cup (Q^1 \times \alpha^2)\rangle$ be their product DFA. Note that $L(\mathcal{B}) = L_1 \cup L_2$. We claim that for every state $\langle q, s\rangle \in Q^1 \times Q^2$, we have that $sensed(\langle q, s\rangle) \subseteq sensed(q) \cup sensed(s)$. Indeed, if $p \notin sensed(q) \cup sensed(s)$, then for every set $S \subseteq P \setminus \{p\}$, it holds that $\delta^1(q, S) = \delta^1(q, S \cup \{p\})$ and $\delta^2(s, S) = \delta^2(s, S \cup \{p\})$. Thus, $\delta(\langle q, s\rangle, S) = \delta(\langle q, s\rangle, S \cup \{p\})$, so $p \notin sensed\langle q, s\rangle$. We thus have that $scost(\langle q, s\rangle) \leq scost(q) + scost(s)$.

It follows that for every word $w \in (2^P)^*$, we have that $scost_{\mathcal{B}}(w) \leq scost_{\mathcal{A}_1}(w) + scost_{\mathcal{A}_2}(w)$. Indeed, in every state in the run of $\mathcal{B}$ on w, the sensing is at most the sum of the sensings in the corresponding states in the runs of $\mathcal{A}_1$ and $\mathcal{A}_2$ on w. Since this is true for every word in Σ^*, then taking the limit of the average cost yields the result. □

We now consider the concatenation of two languages. The following lemma shows that the sensing level may increase from 0 to 1 when concatenating languages.

Lemma 3.7: There are languages $L_1, L_2 \subseteq \Sigma^*$ such that $scost(L_1) = scost(L_2) = 0$, yet $scost(L_1 \cdot L_2) = 1$.

Proof: Let $P = \{a\}$, and consider the languages $L_1 = (2^P)^*$ and $L_2 = \{\{a\}\}$. It is not hard to see that $scost(L_1) = scost(L_2) = 0$. Indeed, a DFA for L_1 consists of a single accepting sink with no sensing, and a DFA for L_2 has a single ergodic component, which is a rejecting sink with no sensing. On the other hand $L_1 \cdot L_2$ consists of all words that end with $\{a\}$ and thus a DFA for it has to always sense a. □

IV. The Sensing Cost of ω-Regular Languages

For the case of finite words, we have a very clean picture: minimizing the state space of a DFA also minimizes its sensing cost. In this section we study the case of infinite words. There, the picture is much more complicated. In Example 2.1 we saw that different minimal DBAs may have a different sensing cost. We start this section by showing that even for languages that have a single minimal DBA, the sensing cost may not be attained by this minimal DBA, and in fact it may be attained only as a limit of a sequence of DBAs.

Example 4.1: Let $P = \{p\}$, and consider the language L of all words $w_1 \cdot w_2 \cdots$ such that $w_i = \{p\}$ for infinitely many i's. Thus, $L = (\texttt{True}^* \cdot p)^\omega$. A minimal DBA for L has two states. The minimal sensing cost for a two-state DBA for L is $\frac{2}{3}$ (the classical two-state DBA for L senses p in both states and thus has sensing cost 1. By taking $\mathcal{A}_1$ in the sequence we shall soon define we can recognize L by a two-state DBA with sensing cost $\frac{2}{3}$). Consider the sequence of DBAs $\mathcal{A}_m$ appearing in Figure 4. The DBA $\mathcal{A}_m$ recognizes $(\texttt{True}^{\geq m} \cdot p)^\omega$, which is equivalent to L, yet enables a "lazy" sensing of p. Formally, The stationary distribution π for $\mathcal{A}_m$ is such that $\pi(q_i) = \frac{1}{m+1}$ for $0 \leq i \leq m-1$ and $\pi(q_m) = \frac{2}{m+1}$. In the states $q_0, \ldots, q_{m-1}$ the sensing cost is 0 and in q_m it is 1. Accordingly, $scost(\mathcal{A}_m) = \frac{2}{m+1}$, which tends to 0 as m tends to infinity. □

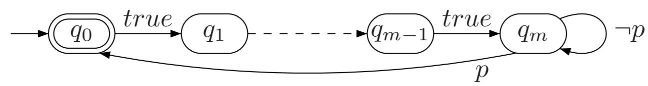

Fig. 4. The DBA $\mathcal{A}_m$.

Remark 4.2: Note that the language L can also be recognized by a one-state DBA in which the acceptance condition is defined with respect to transitions. Our definition of the sensing cost gives cost 0 to one-state automata. In a model with acceptance on the transitions, however, the sensing cost should be defined in a different way, and take into an account the influence of the signal on the transition being accepting. With such a definition, our results apply to both models of acceptance. □

A. Characterizing $scost(L)$ by the residual automaton for L

In this section we state and prove our main result, which characterizes the sensing cost of an ω-regular language by means of the residual automaton for the language:

Theorem 4.3: For every ω-regular language $L \subseteq \Sigma^\omega$, we have $scost(L) = scost(\mathcal{R}_L)$.

The proof is described in the following sections.

1) The cost of the residual automaton is necessary: We start by proving $scost(L) \geq scost(\mathcal{R}_L)$. The considerations here are similar to these used in the proof of Lemma 3.1 for the setting of finite words. We prove that for every DPA $\mathcal{A}$ with $L(\mathcal{A}) = L$, we have that $scost(\mathcal{A}) \geq scost(\mathcal{R}_L)$. Consider a word $w \in \Sigma^\omega$ and a prefix $u \in \Sigma^*$ of w. After reading u, the DPA $\mathcal{R}_L$ reaches the state $[u]$ and the DPA $\mathcal{A}$ reaches a state q with $L(\mathcal{A}^q) = L^u$. As in the case of finite words, for every state $q \in Q$ such that $L(\mathcal{A}^q) = L^u$, it holds that $sensed([u]) \subseteq sensed(q)$, implying that $scost(q) \geq scost([u])$. Now, since this holds for all prefixes u of w, it follows that $scost_{\mathcal{A}}(w) \geq scost_{\mathcal{R}_L}(w)$. Finally, since the latter holds for every word $w \in \Sigma^\omega$, it follows that $scost(\mathcal{A}) \geq scost(\mathcal{R}_L)$.

Note that the arguments in the proof are independent of the acceptance condition of $\mathcal{A}$ and apply also to stronger acceptance conditions, such as the Müller acceptance condition.

2) The cost of the residual automaton is sufficient: In this section we prove that $scost(L) \leq scost(\mathcal{R}_L)$. To show this, we construct a sequence $(\mathcal{B}_n)_{n\geq 1}$ of DPAs such that $L(\mathcal{B}_n) = L$ for every $n \geq 1$, and $\lim_{n\to\infty} scost(\mathcal{B}_n) = scost(\mathcal{R}_L)$. We note that since the DPAs $\mathcal{B}_n$ have the same acceptance condition as $\mathcal{A}$, there is no trade-off between sensing cost and acceptance condition. More precisely, if L can be recognized by a DPA with parity ranks $[i, j]$ (in particular, if L is DBW-recognizable), then the sensing cost for $L(\mathcal{A})$ can be obtained by a DPA with parity ranks $[i, j]$.

We first assume that $\mathcal{A}$ is strongly connected. We will later show how to drop this assumption.

Let $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \alpha_{\mathcal{A}} \rangle$ be a strongly connected DPA for L. We assume that $\mathcal{A}$ is minimally ranked. Thus, if $\mathcal{A}$ has parity ranks $\{0, 1, \ldots, k\}$, then there is no DPA for L with ranks $\{0, 1, \ldots, k-1\}$ or $\{1, 2, \ldots, k\}$. Also, if $\mathcal{A}$ has ranks $\{1, 2, \ldots, k\}$, we consider the complement DPA, which is $\mathcal{A}$ with ranks $\{0, 1, \ldots, k-1\}$. Since DPAs can be complemented by dualizing the acceptance condition, their sensing cost is preserved under complementation, so reasoning about the complementing DPA is sound. For $0 \leq i \leq k$, a cycle in $\mathcal{A}$ is called an *i-loop* if the maximal rank along the cycle is i. For $0 \leq i \leq j \leq k$, an $[i, j]$ - *flower* is a state $q_{\circledast} \in Q$ such that for every $i \leq r \leq j$, there is an r-loop that goes throughout $q_{\circledast}$.

The following lemma is a strengthening to strongly connected DPAs of a result from [15]:

Lemma 4.4: Consider a strongly-connected minimally-ranked DPA $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \alpha_{\mathcal{A}} \rangle$ with ranks $\{0, \ldots, k\}$. Then, there is a DPA $\mathcal{D} = \langle \Sigma, Q, q_0, \Delta, \alpha_{\mathcal{D}} \rangle$ such that all the following hold.

1) For every state $s \in Q$, we have $L(\mathcal{A}^s) = L(\mathcal{D}^s)$. In particular, $\mathcal{A}$ and $\mathcal{D}$ are equivalent.
2) There exist $m \in \mathbb{N}$ such that $\mathcal{D}$ has ranks $\{0, ..., 2m + k\}$ and has a $[2m, 2m + k]$ flower.

Proof: We first claim that $\mathcal{A}$ does not have an equivalent DPA with ranks $\{1, \ldots, k+1\}$. Indeed, assume by contradiction that there is a DPA $\mathcal{B}$ with ranks $\{1, \ldots, k + 1\}$ that recognizes $L(\mathcal{A})$. Let $u \in \Sigma^*$ be such that the run of $\mathcal{B}$ on u ends in an ergodic SCC C of $\mathcal{B}$. Since $\mathcal{A}$ is strongly connected, there is a word $v \in \Sigma^*$ such that the run of $\mathcal{A}$ on the word uv ends in the initial state of $\mathcal{A}$. That is, $L^{uv} = L$. Since $L(\mathcal{B}) = L(\mathcal{A})$, this implies that the run of $\mathcal{B}$ on uv ends in a state q such that $L(\mathcal{B}^q) = L$. Since after reading u, the run is in the ergodic component C, we have $q \in C$. Thus, $\mathcal{B}^q$ is a strongly connected DPA equivalent to $\mathcal{A}$. By our assumption, $L(\mathcal{A})$ cannot be recognized by a DPA with ranks in $\{1, ..., k\}$, and thus the ranks in $\mathcal{B}^q$ are $\{1, ..., k + 1\}$. In particular, there is a state q' in $\mathcal{B}$ with rank $k + 1$.

Let s be a state in $\mathcal{A}$ with rank k. Let $u_0 \in \Sigma^*$ be such that the run of $\mathcal{A}$ on u_0 reaches s, and let $u_1 \in \Sigma^*$ be such that the run of $\mathcal{B}$ on u_0u_1 reaches q'. We can construct in this manner an infinite word $w = u_0u_1u_2 \cdots$ such that for all $i \geq 0$, the run of $\mathcal{A}$ on $u_0, ..., u_{2i}$ reaches s and the run of $\mathcal{B}$ on $u_0, ..., u_{2i+1}$ reaches q'. Thus, the maximal rank in the run of $\mathcal{A}$ (resp. $\mathcal{B}$) on w is k (resp. $k + 1$). However, k and $k + 1$ have different parity, so $L(\mathcal{A}) \neq L(\mathcal{B})$, which contradicts our assumption. We conclude that $L(\mathcal{A})$ is not recognizable by a DPA with ranks $\{1, ..., k + 1\}$.

Now, [15] proves the claim for $\mathcal{A}$ that need not be strongly connected and has no equivalent DPA with ranks $\{1, \ldots, k + 1\}$. There, the DPA $\mathcal{D}$ has ranks in $\{0, ..., 2m + k + 1\}$, and has a $[2m, 2m+k]$-flower $q_{\circledast}$. We argue that since $\mathcal{A}$ is strongly connected, $\mathcal{D}$ has only ranks in $\{0, ..., 2m + k\}$.

By [15], if there exists $m \in \mathbb{N}$ and a DPA $\mathcal{D}$ that recognizes $L(\mathcal{A})$ and has a $[2m, 2m + k + 1]$-flower, then $L(\mathcal{A})$ cannot be recognized by a DPA with ranks $\{1, ..., k + 2\}$. Observe that in this case, $L(\mathcal{A})$ cannot be recognized by a DPA with ranks $\{0, ..., k\}$ as well, as then, by increasing the ranks by 2 we would get a DPA with ranks $\{2, ..., k + 2\}$, contradicting the fact $L(\mathcal{A})$ cannot be recognized by a DPA with ranks in $\{1, ..., k + 2\}$. Hence, as $\mathcal{A}$ with ranks $\{0, ..., k\}$ does exist, the DPA $\mathcal{D}$ cannot have a $[2m, 2m + k + 1]$-flower.

Now, in our case, the DPA $\mathcal{A}$, and therefore also $\mathcal{D}$, is strongly-connected. Thus, if $\mathcal{D}$ has a state with rank $2m+k+1$, then the state $q_{\circledast}$ is in the same component with this state, and is therefore a $[2m, 2m+k+1]$ flower. By the above, however, $\mathcal{D}$ cannot have a $[2m, 2m + k + 1]$ flower, implying that $\mathcal{D}$ has ranks in $\{0, ..., 2m + k\}$. ☐

Let $\mathcal{A}$ and $\mathcal{D}$ be as in Lemma 4.4, and $q_{\circledast}$ be the $[2m, 2m + k]$-flower in $\mathcal{D}$. Note that $\mathcal{A}$ and $\mathcal{D}$ have the same structure and differ only in their acceptance condition. Let $\Omega = \{0, ..., 2m + k\}$. For a word $w \in \Sigma^*$, let $\rho = s_1, s_1, ..., s_n$ be the run of $\mathcal{D}$ on w. If ρ ends in $q_{\circledast}$, we define the *$q_{\circledast}$-loop-abstraction of* w to be the rank-word $\text{abs}(w) \in \Omega^*$ of maximal ranks between successive visits to $q_{\circledast}$. Formally, let $w = y_0 \cdot y_1 \cdots y_t$ be a partition of w such that $\mathcal{D}$ visits the state $q_{\circledast}$ after reading the prefix $y_0 \cdots y_j$, for all $0 \leq j \leq t$, and does not visit $q_{\circledast}$ in other positions. Then, $\text{abs}(y_i)$, for $0 \leq i \leq t$, is the maximal rank read along y_i, and $\text{abs}(w) = \text{abs}(y_0) \cdot \text{abs}(y_1) \cdots \text{abs}(y_t)$. Recall that $\mathcal{R}_L = \langle \Sigma, \langle L \rangle, \Delta_L, [\epsilon], \alpha \rangle$, where $\langle L \rangle$ are the equivalence classes of the right-congruence relation on L, thus each state $[u] \in \langle L \rangle$ is associated with the language L^u of words w such that $uw \in L$. We define a function $\varphi : Q \to \langle L \rangle$ that maps states of $\mathcal{A}$ to languages in $\langle L \rangle$ by $\varphi(q) = L(\mathcal{A}^q)$. Observe that φ is onto. We define a function $\gamma : \langle L \rangle \to Q$ that maps languages in $\langle L \rangle$ to states of $\mathcal{A}$ by arbitrarily choosing for every language $L^u \in \langle L \rangle$ a state in $\varphi^{-1}(L^u)$.

We define a sequence of words $u_{2m}, \ldots, u_{2m+k} \in \Omega^*$ as follows. The definition proceeds by an induction. Let $M = |Q| + 1$. First, $u_{2m} = (2m)^M$. Then, for $2m < i \leq 2m + k$, we have $u_i = (i \cdot u_{i-1})^{M-1} \cdot i$. For example, if $m = 2$ and $|Q| = 2$, then $u_4 = 444$, $u_5 = 544454445$, $u_6 = 6544454445654445444 56$, and so on. Let $\mathcal{P}$ be a DFA that accepts a (finite) word $w \in \Sigma^*$ iff the run of $\mathcal{D}$ on w ends in $q_{\circledast}$ and u_{2m+k} is a suffix of $abs(w)$, for the word $u_{2m+k} \in \Omega^*$

defined above. In Appendix B we describe how to construct $\mathcal{P}$, essentially by combining a DFA over that alphabet Ω that recognizes $\Omega^* \cdot u_{2m+k}$ with a DFA with state space $Q \times \Omega$ that records the highest rank visited between successive visits to $q_{\circledast}$ and thus abstracts words in Σ^*.

We can now turn to the construction of the DFAs $\mathcal{B}_n$. Recall that $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \alpha_{\mathcal{A}} \rangle$, and let $\mathcal{P} = \langle \Sigma, Q_{\mathcal{P}}, t_0, \Delta_{\mathcal{P}}, \{t_{acc}\} \rangle$. For $n \geq 1$, we define $\mathcal{B}_n = \langle \Sigma, Q_n, \langle q_0, t_0 \rangle, \Delta_n, \alpha_n \rangle$ as follows. The states of $\mathcal{B}_n$ are $Q_n = (\langle L \rangle \times \{1, \ldots, n\}) \cup (Q \times (Q_{\mathcal{P}} \setminus \{t_{acc}\}))$, where t_{acc} is the unique accepting state of $\mathcal{P}$. We refer to the two components in the union as the $\mathcal{R}_L$-*component* and the $\mathcal{D}$-*component*, respectively. The transitions of $\mathcal{B}_n$ are defined as follows.

- Inside the $\mathcal{R}_L$-component: for every transition $\langle [u], a, [u'] \rangle \in \Delta_L$ and $i \in \{1, \ldots, n-1\}$, there is a transition $\langle ([u], i), a, ([u'], i+1) \rangle \in \Delta_n$.
- From the $\mathcal{R}_L$-component to the $\mathcal{D}$-component: for every transition $\langle [u], a, [u'] \rangle \in \Delta_L$, there is a transition $\langle ([u], n), a, (\gamma([u']), t_0) \rangle \in \Delta_n$.
- Inside the $\mathcal{D}$-component: for every transitions $\langle q, a, q' \rangle \in \Delta$ and $\langle t, a, t' \rangle \in \Delta_{\mathcal{P}}$ with $t' \neq t_{acc}$, there is a transition $\langle (q, t), a, (q', t') \rangle \in \Delta_n$.
- From the $\mathcal{D}$-component to the $\mathcal{R}_L$-component: for every transitions $\langle q, a, q' \rangle \in \Delta$ and $\langle t, a, t_{acc} \rangle \in \Delta_{\mathcal{P}}$, there is a transition $\langle (q, t), a, (\varphi(q'), 0) \rangle \in \Delta_n$.

The acceptance condition of $\mathcal{B}_n$ is induced by that of $\mathcal{A}$. Formally $\alpha_n(q, t) = \alpha_{\mathcal{A}}(q)$, for states $(q, t) \in Q \times Q_{\mathcal{P}}$, and $\alpha_n([u], i) = 0$ for states $([u], i) \in \langle L \rangle \times \{1, \ldots, n\}$.

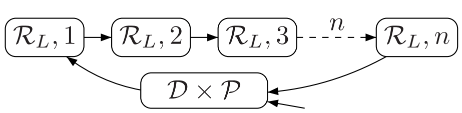

Fig. 5. The DPA $\mathcal{B}_n$.

The idea behind the construction of $\mathcal{B}_n$ is as follows. The automaton $\mathcal{B}_n$ stays in $\mathcal{R}_L$ for n steps, then proceeds to a state in $\mathcal{D}$ with the correct residual language, and simulates $\mathcal{D}$ until the ranks corresponding to the word u_{2m+k} have been seen. It then goes back to $\mathcal{R}_L$, by projecting the current state of $\mathcal{D}$ onto its residual in $\langle L \rangle$. The bigger n is, the more time a run spends in the $\mathcal{R}_L$-component, making $\mathcal{R}_L$ the more dominant factor in the sensing cost of $\mathcal{B}_n$. As n tends to infinity, the sensing cost of $\mathcal{B}_n$ tends to that of $\mathcal{R}_L$. The technical challenge is to define $\mathcal{P}$ in such a way so that even though the run spends less time in the $\mathcal{D}$ component, we can count on the ranks visited during this short time in order to determine whether the run is accepting. We are now going to formalize this intuition, and we start with the most challenging part of the proof, namely the equivalence of $\mathcal{B}_n$ and $\mathcal{A}$. The proof is decomposed into the three lemmas 4.5, 4.6, and 4.7.

Lemma 4.5: Consider a word $u \in \Sigma^*$ such that the run of $\mathcal{B}_n$ on u reaches the $\mathcal{D}$-component in state $\langle q, t \rangle$. Then, $L(\mathcal{D}^q) = L(A^q) = L^u$.

Proof: We prove a stronger claim, namely that if the run of $\mathcal{B}_n$ on u ends in the $\mathcal{R}_L$-component in a state $\langle s, i \rangle$, then $s = [u]$, and if the run ends in the $\mathcal{D}$-component in a state $\langle q, t \rangle$, then $L(\mathcal{A}^q) = L^u$. The proof proceeds by induction on $|u|$ and is detailed in Appendix C. By Lemma 4.4, for all states $q \in Q$, we have $L(\mathcal{A}^q) = L(\mathcal{D}^q)$, so the claim follows. □

Lemma 4.6: If the run of $\mathcal{B}_n$ on a word $w \in \Sigma^\omega$ visits the $\mathcal{R}_L$-component finitely many times, then $w \in L$ iff $w \in L(\mathcal{B}_n)$.

Proof: Let $u \in \Sigma^*$ be a prefix of w such that the run of $\mathcal{B}_n$ on w stays forever in the $\mathcal{D}$-component after reading u. Let $(q, t) \in Q_n$ be the state reached by $\mathcal{B}_n$ after reading u. By Lemma 4.5, we have $L(\mathcal{A}^q) = L^u$. Since the run of $\mathcal{B}_n$ from (q, t) stays in the $\mathcal{D}$-components where it simulates the run of $\mathcal{A}$ from q, then $\mathcal{A}^q$ accepts the suffix $w^{|u|}$ iff $\mathcal{B}_n^{(q,t)}$ accepts $w^{|u|}$. It follows that $w \in L$ iff $w \in L(\mathcal{B}_n)$. □

The complicated case is when the run of $\mathcal{B}_n$ on w does visit the $\mathcal{R}_L$-component infinitely many times. Here is where the the special structure of $\mathcal{P}$ enters the pictures and guarantees that the sparse visits in the $\mathcal{D}$-component are sufficient for determining acceptance.

Lemma 4.7: If the run of $\mathcal{B}_n$ on a word $w \in \Sigma^\omega$ visits the $\mathcal{R}_L$-component infinitely many times, then $w \in L$ iff $w \in L(\mathcal{B}_n)$.

Proof: Let $\tau = s_1, s_2, s_3, \ldots$ be the run of $\mathcal{B}_n$ on w and let $\rho = q_1, q_2, q_3 \ldots$ be the run of $\mathcal{A}$ on w. We denote by $\tau[i, j]$ the infix $s_i, ..., s_j$ of τ. We also extend $\alpha_{\mathcal{D}}$ to (infixes of) runs by defining $\alpha_{\mathcal{D}}(\tau[i, j]) = \alpha_{\mathcal{D}}(s_i), ..., \alpha_{\mathcal{D}}(s_j)$. For a rank-word $u \in \Omega^*$, we say that an infix $\tau[i, j]$ is a u-*infix* if $\alpha_{\mathcal{D}}(\tau[i, j]) = u$.

If $v = \tau[i, j]$, for some $0 \leq i \leq j$, is a part of a run of $\mathcal{D}$ that consists of loops around $q_{\circledast}$, we define the *loop type of* v to be the word in Ω^* that describes the highest rank of each simple loop around $q_{\circledast}$ in v. An infix of τ whose loop type is u_i for some $2m \leq i \leq 2m + k$ is called a u_i-*loop-infix*.

By our assumption, τ contains infinitely many u_{2m+k}-infixes. Indeed, by the definition of $\mathcal{P}$, otherwise τ get trapped in the $\mathcal{D}$-component. We proceed by establishing a connection between u_i-loop-infixes of τ and the corresponding infixes of ρ, for all $2m \leq i \leq 2m + k$.

Let $i \in \{2m, \ldots, 2m+k\}$, and consider a u_i-loop-infix, By the definition of u_i, such a u_i-loop-infix consists of a sequence of $M = |Q| + 1$ i-loops in τ, with loops of lower ranks between them. We can write $w = xvw'$, where $v = w[c, d]$ is the sub word that corresponds to the u_i-loop-infix. Let $u_i' = \alpha_{\mathcal{A}}(\rho[c, d])$ be the ranks of ρ in the its part that corresponds to v.

By our choice of M, we can find two indices $c \leq j < l \leq d$ such that the pairs $\langle (q_j, t), q_j' \rangle$ and $\langle (q_l, t'), q_l' \rangle$ reached by (τ, ρ) in indices j and l, respectively, satisfy $q_j = q_l = q_{\circledast}$ and $q_j' = q_l'$. Additionally, being a part of the run on a u_i-loop-infix, the highest rank seen between q_j and q_l in τ is i.

We write $v = v_1v_2v_3$, where $v_1 = v[1, j]$, $v_2 = v[j+1, l]$, and $v_3 = v[l+1, |v|]$. Thus, the loop type of v_2 is in $(iu_{i-1})^+i$, with the convention $u_{2m-1} = \epsilon$.

Consider the runs μ and η of $\mathcal{D}^{q_j}$ and of $\mathcal{A}^{q'_j}$ on v_2^ω, respectively. These runs are loops labeled by v_2, where the highest rank in μ is i. By Lemma 4.5, $L(\mathcal{D}^{q_j}) = L(\mathcal{D}^{q'_j}) = L(\mathcal{A}^{q'_j})$, so the highest rank in η must have same parity as i.

Thus, we showed that for every $i \in \{2m, ..., 2m+k\}$, and for every u_i-loop-infix v of τ, there is an infix of v with loop-type in $(iu_{i-1})^+i$, such that the infix of ρ corresponding to v has highest rank of same parity as i.

We want to show that rank k is witnessed on ρ during every u_{2m+k}-infix of τ. Assume by way of contradiction that this is not the case. This means that there is some u_{2m+k}-infix v' in τ such that all ranks visited in ρ along v' are at most $k-2$. Indeed, since the highest rank has to be of the same parity as $2m+k$, which has the same parity as k, it cannot be $k-1$. By the same argument, within v' there is an infix v'' of u_{2m+k-1} of the form $((2m+k-1)(u_{2m+k-2}))^+(2m+k-1)$ in which the highest rank in ρ is of the same parity as $k-1$. As v'' is also an infix of v', the highest rank in ρ along v'' is at most $k-2$. Thus, the highest rank along v'' is at most $k-3$. By continuing this argument by induction down to 0, we reach a contradiction (in fact it is reached at level 1), as no rank below 0 is available.

We conclude that the run ρ witnesses a rank k in any u_k-infix of τ. Since τ contains infinitely many u_k-infixes, then ρ contains infinitely many ranks k, and, depending on the parity of k, either both ρ and τ are rejecting or both are accepting.

This concludes the proof that $w \in L$ iff $w \in L(\mathcal{B}_n)$. □

We proceed to show that the sensing cost of the sequence of DPAs $\mathcal{B}_n$ indeed converges to that of $\mathcal{R}_L$.

Lemma 4.8: $\lim_{n\to\infty} scost(\mathcal{B}_n) = scost(\mathcal{R}_L)$.

Proof: Since $\mathcal{D}$ is strongly connected, then $q_{❁}$ is reachable from every state in $\mathcal{D}$. Also, since $q_{❁}$ is a $[2m, 2m+k]$-flower, we can construct a sequence of loops around $q_{❁}$ whose ranks correspond to the word u_{2m+k}. Thus, t_{acc} is reachable from every state in the $\mathcal{D}$-component.

Since $\mathcal{A}$ is strongly connected, so is $\mathcal{R}_L$. This, together with the fact t_{acc} is reachable from every state in a $\mathcal{D}$-component, implies that $\mathcal{B}_n$ is strongly connected. So, a run of $\mathcal{B}_n$ is expected to traverse both components infinitely often, making the $\mathcal{R}_L$-component more dominant as n grows, implying that $\lim_{n\to\infty} scost(\mathcal{B}_n) = scost(\mathcal{R}_L)$. Formalizing this intuition involves a careful analysis of $\mathcal{B}_n$'s Markov chain, as detailed below.

Consider the Markov chain that corresponds to $\mathcal{B}_n$, and let T_n be its transition matrix. For a vector $v = (v_1, \ldots, v_m)$, let $\|v\| = \sum_{i=1}^m v_i$. The sensing cost of $\mathcal{B}_n$ is computed using the limiting distribution π_n of $\mathcal{B}_n$. Since $\mathcal{B}_n$ is strongly connected, it has a unique stationary distribution. Thus π_n is obtained as a solution of the equation $\pi_n T_n = \pi_n$, subject to the constraint $\|\pi_n\| = 1$. We denote by $x_n = (x_{n,1}, \ldots, x_{n,d})$ the sub-vector of π_n that corresponds to the $\mathcal{D}$-component, and denote by $y_{n,i}$ the sub-vector that corresponds to the i-th $\mathcal{R}_L$-component. For every $1 \leq i < n$, it is easy to see that $\|y_{n,i}\| = \|y_{n,i+1}\|$. Indeed, all the transitions from the i-th copy of $\mathcal{R}_L$ are to the $(i+1)$-th copy. Thus, $\|y_{n,i}\|$ is independent of i. Let $a_n = \|y_{n,1}\| \geq 0$ and $b_n = \|x_n\| \geq 0$. Observe that for every n, we have that $na_n + b_n = 1$, so in particular, $\lim_{n\to\infty} a_n = 0$.

Let $\epsilon > 0$. By the definition of $\mathcal{P}$, we always enter the first $\mathcal{R}_L$-component in the state $[q_{❁}]$ of $\mathcal{R}_L$ – the state corresponding to $L(\mathcal{A}^{q_{❁}})$. Let τ_0 be the distribution over the states of $\mathcal{R}_L$ in which $[q_{❁}]$ is assigned probability 1 and the other states of $\mathcal{R}_L$ are assigned 0, and let $\theta = (\theta_1, \ldots, \theta_l)$ be the unique stationary distribution of $\mathcal{R}_L$. Let R be the matrix associated with the Markov chain of $\mathcal{R}_L$, and let $\tau_i = \tau_0 R^i$ for every $i \geq 1$. By [7], there exists n_0 such that for all index $i \geq n_0$ and $1 \leq j \leq l$, we have that $|\tau_{i,j} - \theta_j| \leq \epsilon$. Note that for all n and i, it holds that $y_{n,i} = \tau_i$.

Let $\{q_1, \ldots, q_d\}$ be the states in the $\mathcal{D}$-component. Since $\mathcal{P}$ is strongly connected, then for every $1 \leq i, j \leq d$ there is a path from q_i to q_j with at most $d-1$ transitions. Since there are at most $|\Sigma|$ edges leaving each state, the probability of taking each edge along such a path is at least $\mu = \frac{1}{|\Sigma|}$. Therefore, the probability of reaching q_j from q_i is at least μ^{d-1}. Consider the maximal entry in x_n (w.l.o.g $x_{n,1}$). It holds that $x_{n,1} \geq \frac{\|x_n\|}{d} = \frac{b_n}{d}$. Therefore, for all $1 \leq j \leq d$, we have $x_{n,j} \geq \mu^{d-1} x_{n,1} \geq \frac{\mu^{d-1}}{d} b_n$.

Recall that t_{acc} is reachable from all the states in the $\mathcal{D}$-component. Therefore, there is at least one transition from some state q_j of the $\mathcal{D}$-component to the first $\mathcal{R}_L$-component. This means that $a_n \geq \mu \cdot x_{n,j} \geq \frac{\mu^d}{d} b_n$, implying that $b_n \leq \mu^{-d} \cdot d \cdot a_n$, which tens to 0 when n tends to ∞.

We now consider the cost of $\mathcal{B}_n$, for $n \geq n_0$. Clearly, the maximal cost of a state is $|P|$. Let c_j be the cost of the state indexed j in $\mathcal{R}_L$, and let $\tau_i = (\tau_{i,1} \ldots, \tau_{i,l})$. Then,

$$\begin{aligned} &scost(B_n) \leq b_n|P| + n_0 a_n|P| + a_n \textstyle\sum_{i=n_0}^n \sum_{j=1}^d \tau_{i,j} c_j \\ &\leq b_n|P| + n_0 a_n|P| + a_n \textstyle\sum_{i=n_0}^n \sum_{j=1}^d (\theta_j + \epsilon) c_j. \end{aligned}$$

Therefore, when $n \to \infty$, as $a_n \to 0$ and $b_n \to 0$, we get $scost(B_n) \leq (n-n_0)a_n \sum_{j=1}^d \theta_j c_j + O(\epsilon) + o(1)$. But we know $na_n + b_n = 1$, and $b_n \to 0$, so $na_n \to 1$, and therefore $(n-n_0)a_n \to 1$. We get $scost(B_n) \leq scost(\mathcal{R}_L) + O(\epsilon) + o(1)$. Furthermore, by Lemmas 4.6 and 4.7, for all n we have $L(\mathcal{B}_n) = L(\mathcal{A})$, thus $scost(\mathcal{R}_L) \leq scost(B_n)$.

Since the above holds for all $\epsilon > 0$, we conclude that $\lim_{n\to\infty} scost(B_n) = scost(\mathcal{R}_L)$. □

Lemmas 4.6, and 4.7 put together ensure that for strongly connected automata, we have that $L(\mathcal{B}_n) = L$, so with Lemma 4.8, we get $scost(L) = scost(\mathcal{R}_L)$.

It is left to remove the assumption about $\mathcal{A}$ being strongly connected. Assume then that $\mathcal{A}$ is not a strongly connected DPA, and let $C_1, \ldots, C_l$ be its SCCs. For each $1 \leq i \leq l$, let R_i be the residual automaton of C_i, with no state specified as an initial state (since C_i is strongly connected, then, by Remark 2.5, the sensing cost of $L(C_i)$ is independent of its initial state). We can assume that all the R_i are distinct, as otherwise intersecting components could have been merged in $\mathcal{A}$. Let $(\mathcal{B}_{1,n},\ldots,\mathcal{B}_{l,n})_{n\geq 1}$ be a sequences of DPAs we have constructed above for strongly connected DPAs, with $\mathcal{B}_{i,n}$ corresponding to C_i. In particular, for every $1 \leq i \leq l$, we have $\lim_{n\to\infty} scost(\mathcal{B}_{i,n}) = scost(L(C_i))$. Let $\mathcal{A}_n$ be the DPA obtained from $\mathcal{A}$ by replacing each SCC C_i by $\mathcal{B}_{i,n}$, with the entry points to $\mathcal{B}_{i,n}$ being chosen to preserve the correct residual language. It is easy to see that for all $n \geq 1$, we have $L(\mathcal{A}_n) = L(\mathcal{A})$. Let ρ (resp. ρ_n) be the SCC-reachability distribution of $\mathcal{A}$ (resp. $\mathcal{A}_n$). For every $1 \leq i \leq l$ and $n \geq 1$, we have that $\rho(C_i) = \rho_n(\mathcal{B}_{i,n})$. Therefore, $scost(\mathcal{A}_n) = \sum_{i=1}^{l} \rho(C_i) scost(\mathcal{B}_{i,n})$. When n tends to ∞, we get $\sum_{i=1}^{l} \rho(C_i) scost(C_i)$.

Finally, let $\mathcal{A}_R$ be the DPA obtained from $\mathcal{A}$ by replacing each SCC C_i by its residual automaton R_i, again keeping the entry points to R_i consistent with residuals. Since the SCC-reachability distribution in $\mathcal{A}$ and $\mathcal{A}_R$ coincide, it follows that $scost(\mathcal{A}_R) = scost(\mathcal{R}_L)$. Since $scost(\mathcal{A}_R) = \sum_{i=1}^{l} \rho(C_i) scost(C_i)$, we can conclude that $scost(\mathcal{A}) = scost(\mathcal{R}_L)$ and we are done.

Remark 4.9: It is easy to see that all our results can be easily extended to a setting with a non-uniform distribution on the letters, or with a different cost for each input in each state. ☐

V. Directions for Future Research

Regular sensing is a basic notion, which we introduced and studied for languages of finite and infinite words. In this section we discuss possible extensions and variants of our definition and contribution.

Open systems: Our setting assumes that all the signals in P are generated by the environment and read by the automaton. This corresponds, for example, to the case the automaton is a *monitor* that observes behaviors and decides whether they are correct. In the setting of open systems we partition P into a set I of input signals, generated by the environment, and set O of output signals, generated by the system. Given a specification over $I \cup O$, the goal is to construct a finite-state system, a.k.a. a *transducer*, that, given a sequence of input signals, generates a computation that satisfies the specification. Studying sensing for open systems, we define the sensing cost of a language as the minimal sensing cost required for a transducer that realizes it. Note that here, sensing is measured only with respect to the signals in I. Also, the transducer does not have to generate all the words in the language – it only has to associate a computation in the language with each input sequence. These two differences may lead to significantly different results for seeing in the setting of open systems.

Trade-off between sensing and quality: The key idea in the proof of Theorem 4.3 is that when we reason about languages of infinite words, it is sometimes possible to delay the sensing and only sense in "sparse" intervals. This sort of lazy sensing is sound, as eventualities are allowed to be satisfied in an unboundedly-far future (see also Example 4.1). In practice, however, it is often desirable to satisfy eventualities quirky. This is formalized in multi-valued formalisms such as LTL with future discounting [1], where formulas assign higher satisfaction values to computations that satisfy eventualities fast. Our study here suggests that lower sensing leads to lower satisfaction values. An interesting problem is to study and formalize this intuitive trade-off between sensing and quality.

Transient cost: In our definition of sensing, transient states are of no importance. Consequently, for example, all safety languages have sensing cost 0, as the probability of a safety property not being violated is 0, and once it is violated, no sensing is required. An alternative definition of sensing cost may take transient states into an account. One way to do it is to define the sensing cost of a run as the discounted sum of the sensing costs of the states it visits. Thus, for example, the sensing cost of a run $q_0, q_1, \ldots$ is $\sum_{i\geq 0} 2^{-i} \cdot sensed(|q_i|)$. One could also use different types of discounting. This way, transient states affect the sensing cost, ergodic sets are less dominant, and the sensing cost of safety languages is typically positive. For example, with the above suggested discounting, the sensing cost of the language $\{p^\omega\}$ over the alphabet $2^{\{p\}}$ is $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots = 1$.

Beyond regular: Our definition of sensing cost can be adapted to measure more complex models, such as push-down automata or Turing machines. It would be interesting to see the trade-off between sensing and classical complexity measures in these models.

References

[1] S. Almagor and O. Kupferman. Discounting in LTL In *Proc. 20th TACAS*, To appear, 2014.

[2] S. Almagor and O. Kupferman. Latticed-LTL synthesis in the presence of noisy inputs. In *Proc. 17th FOSSACS*, To appear, 2014.

[3] G. Avni and O. Kupferman. When does abstraction help? *IPL*, 113:901–905, 2013.

[4] K. Chatterjee and R. Majumdar. Minimum attention controller synthesis for ω-regular objectives. In *FORMATS*, pages 145–159, 2011.

[5] K. Chatterjee, R. Majumdar, and T. A. Henzinger. Controller synthesis with budget constraints. In *Proc 11th HSCC, LNCS 4981*, pages 72–86, 2008.

[6] D.L. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52:1289–1306, 2006.

[7] C. Grinstead and J. Laurie Snell. 11:Markov chains. In *Introduction to Probability*. American Mathematical Society, 1997.

[8] G. Kindler. *Property Testing, PCP, and Juntas*. PhD thesis, Tel Aviv University University, 2002.

[9] O. Kupferman and M.Y. Vardi. Church's problem revisited. *The Bulletin of Symbolic Logic*, 5(2):245 – 263, 1999.

[10] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.

[11] C. Mauduit and A. Sárköz. On finite pseudorandom binary sequences. i. measure of pseudorandomness, the legendre symbol. *Acta Arith.*, 82(4):365–377, 1997.

[12] S. Muthukrishnan. Theory of data stream computing: where to go. In *Proc. 30th PODS*, pages 317–319, 2011.

[13] J. Myhill. Finite automata and the representation of events. TR WADD TR-57-624, pages 112–137, Wright Patterson AFB, Ohio, 1957.

[14] A. Nerode. Linear automaton transformations. *Proc. American Mathematical Society*, 9(4):541–544, 1958.

[15] D. Niwinski and I. Walukiewicz. Relating hierarchies of word and tree automata. In *Proc. 15th STACS, LNCS 1373*, 1998.

[16] S. Schewe. Beyond Hyper-Minimisation—Minimising DBAs and DPAs is NP-Complete. In *Proc. FSTTCS*, LIPIcs 8, pages 400-411, 2010.

[17] Y. Velner and A. Rabinovich. Church synthesis problem for noisy input. In *Proc. 14th FOSSACS*, pages 275–289, 2011.

Appendix

A. Calculating the SCC-reachability distribution

The stationary distribution π_C of each ergodic SCC C can be computed in polynomial time by solving a system of linear equations.

We show that the SCC-reachability distribution $\rho : \mathcal{C} \to [0,1]$ can also be calculated in polynomial time.First, if the initial state q_0 is in an ergodic SCC, the result is trivial. Otherwise, we proceed as follows. We associate with $\mathcal{A}$ the Markov chain M'_A, in which we contract each ergodic SCC of $\mathcal{A}$ to a single state. That is, M'_A is obtained from M_A by replacing each $C \in \mathcal{C}_e$ by a single state q_C. Notice that M'_A is an *absorbing* Markov chain, thus it reaches a sink state with probability 1. Indeed, the probability of reaching an ergodic SCC in M_A is 1, and every SCC in M_A becomes a sink state in M'_A.

By indexing the rows and columns in the transition matrix of M'_A such that transient states come before ergodic states, we can put the matrix in a normal form $\begin{pmatrix} T & E \\ 0 & I \end{pmatrix}$, where T describes the transitions between transient states, E from transient to ergodic states, and I is the identity matrix of size $|\mathcal{C}_e|$. Note that, indeed, there are no transitions from ergodic states to transient ones, which explains the 0 matrix in the bottom left, and that I captures the fact the ergodic states are sinks. By [7], the entry at coordinates (q_t, q_C) in the matrix $B = (I - T)^{-1}E$ is the probability of reaching the sink q_C starting from the transient state q_t. Therefore, for every $C \in \mathcal{C}_e$, we have that $\rho(C) = B_{(q_0, q_C)}$.

B. The construction of the auxiliary DFA $\mathcal{P}$

Let $\mathcal{H}_{2m+k} = \langle \Omega, Q', q'_0, \Delta', \alpha' \rangle$ be the minimal DFA that recognizes the language $\Omega^* \cdot u_{2m+k}$. We can define $\mathcal{H}_{2m+k}$ so that α' contains a single state q'_{acc}. Indeed, there is a single accepting Myhill-Nerode class of the language $\Omega^* \cdot u_{2m+k}$.

Let $\mathcal{H}$ be the DFA with state space $Q \times \Omega$ and alphabet Σ that maintains in its state the highest rank seen since the last occurrence of $q_{\circledast}$ (or since the beginning of the word, if no $q_{\circledast}$ has been seen) in the run of $\mathcal{D}$ on the word. Thus, $\mathcal{H}$ is in state $\langle q, i \rangle$ iff the highest rank that was visited by $\mathcal{D}$ since the last visit to $q_{\circledast}$ is i. Observe that simulating $\mathcal{H}$ when $\mathcal{D}$ is in an r-loop that started from $q_{\circledast}$, means that the next visit to $q_{\circledast}$ will make $\mathcal{H}$ reach the state $\langle q_{\circledast}, r \rangle$.

Formally, $\mathcal{H} = \langle \Sigma, Q \times \Omega, \langle q_0, 0 \rangle, \Delta_{\mathcal{H}}, Q \times \Omega \rangle$, where $\Delta_{\mathcal{H}}$ is defined as follows.

- For every state $\langle q, i \rangle$ where $q \neq q_{\circledast}$, and for every $\sigma \in \Sigma$, we have $\langle \langle q, i \rangle, \sigma, \langle s, \max\{i, i'\} \rangle \rangle \in \Delta_{\mathcal{H}}$ where s is such that $\langle q, \sigma, s \rangle \in \Delta$, and $i' = \alpha_{\mathcal{D}}(s)$.
- For a state $\langle q_{\circledast}, i \rangle$ and for $\sigma \in \Sigma$, we have $\langle \langle q_{\circledast}, i \rangle, \sigma, \langle s, i' \rangle \rangle \in \Delta_{\mathcal{H}}$ where s is such that $\langle q, \sigma, s \rangle \in \Delta$, and $i' = \alpha_{\mathcal{D}}(s)$.

We obtain $\mathcal{P}$ by composing $\mathcal{H}$ with $\mathcal{H}_{2m+k}$ as follows. In every step of a run of $\mathcal{D}$, the DFA $\mathcal{P}$ advances in the DFA $\mathcal{H}$, while the DFA $\mathcal{H}_{2m+k}$ only advances when we visit $q_{\circledast}$, and it advances according to the highest rank stored in $\mathcal{H}$.

Formally, $\mathcal{P} = \langle \Sigma, Q_{\mathcal{P}}, t_0, \Delta_{\mathcal{P}}, \{t_{acc}\} \rangle$, where $Q_{\mathcal{P}} = Q \times \Omega \times Q'$, $t_0 = \langle q_0, 0, q'_0 \rangle$, $t_{acc} = \langle q_{\circledast}, 2m{+}k, q'_{acc} \rangle$ and the transition relation is defined as follows. For every state $\langle q, i, s \rangle \in q_{\mathcal{P}}$ and letter $\sigma \in \Sigma$, we have $\langle \langle q, i, s \rangle, \sigma, \langle q', i', s' \rangle \rangle \in \Delta_{\mathcal{P}}$, where $\langle q', i' \rangle$ is such that $\langle \langle q, i \rangle, \sigma, \langle q', i' \rangle \rangle \in \Delta_{\mathcal{H}}$, and s' is such that $\langle s, i', s' \rangle \in \Delta'$ if $q' = q_{\circledast}$, while $s' = s$ if $q \neq q_{\circledast}$.

C. Details of the proof of Lemma 4.5

We complete the proof by induction.

For $u = \epsilon$, the claim is trivial, as $\mathcal{B}_n$ starts in $\langle q_0, t_0 \rangle$. Consider the word $u \cdot \sigma$ for $u \in \Sigma^*$ and $\sigma \in \Sigma$. By the induction hypothesis, if the run of $\mathcal{B}_n$ on u ends in an $\mathcal{R}_L$ component in state $\langle s, i \rangle$, then $s = [u]$. If $i < n$, then, by the definition of $\mathcal{R}_L$, the next state is $\langle [u \cdot \sigma], i+1 \rangle$, we are done. If $i = n$ then the next state is $\langle \gamma([u \cdot \sigma]), t_0 \rangle$. By the definition of γ, we have $L(\mathcal{A}^{\gamma([u \cdot \sigma])}) = L^{u \cdot \sigma}$, so we are done.

We continue to the case the run of $\mathcal{B}_n$ on u ends in the $\mathcal{D}$-component. If the run ends in a state $\langle p, t \rangle$ such that $\langle t, \sigma, t_{acc} \rangle \notin \Delta_{\mathcal{P}}$, then, by the induction hypothesis, we have that $L(\mathcal{A}^p) = L^u$. Reading σ, we move to a state $\langle p', t' \rangle$ such that $\langle p, \sigma, p' \rangle \in \Delta$, thus $L(\mathcal{A}^{p'}) = L^{u \cdot \sigma}$, and we are done. Otherwise, $\langle t, \sigma, t_{acc} \rangle \in \Delta_{\mathcal{P}}$ and the next state of $\mathcal{B}_n$ is $\langle \varphi(p'), 1 \rangle$. By the definition of φ, we have $\varphi(p') = [u \cdot \sigma]$, and we are done.