# Regular Sensing

**Shaull Almagor[1], Denis Kuperberg[2], and Orna Kupferman[1]**

1 **The Hebrew University, Israel**
2 **The University of Warsaw, Poland**

## Abstract

The size of deterministic automata required for recognizing regular and $\omega$-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. 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 attained only by an infinite sequence of automata. We show that the optimal limit cost of such sequences can be characterized by the language's right-congruence relation, which enables us to find the sensing cost of $\omega$-regular languages in polynomial time.

**1998 ACM Subject Classification** F.4.3 Formal Languages, B.8.2 Performance Analysis and Design Aids, F.1.1 Models of Computation

**Keywords and phrases** Automata, regular languages, $\omega$-regular languages, complexity, sensing, minimization

**Digital Object Identifier** 10.4230/LIPIcs.FSTTCS.2014.161

## 1 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) [9] or communication complexity (number and length of messages required) [8]. For regular and $\omega$-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 [6]. In more practical applications, mathematical tools in signal processing are used to reconstruct information based on compressed sensing [4], 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" [10].

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 $\omega$-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 [7, 2]. A more sophisticated case of sensing in the context of synthesis is studied in [3], 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.

We study the 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.[1] Thus, the sensing cost of $\mathcal{A}$ is the limit of the expected sensing of runs over words of increasing length.[2] 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 $L$. 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 then study the sensing cost of $\omega$-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

[1] Our study and results apply also to a non-uniform distribution on the letters, given by a Markov chain (see Remark 19).

[2] 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 [3], 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.

problem of finding one is NP-complete [12]. 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 costs.

Before describing our results, let us describe a motivating example that demonstrates the intricacy in the case of $\omega$-regular languages. Consider a component in a vacuum-cleaning robot that monitors the dust collector and checks that it is empty infinitely often. The proposition *empty* indicates whether the collector is empty and a sensor needs to be activated in order to know its truth value. One implementation of the component would sense *empty* throughout the computation. This corresponds to the classical two-state DPA for "infinitely often *empty*". A different implementation can give up the sensing of *empty* for some fixed number $k$ of states, then wait for *empty* to hold, and so forth. The bigger $k$ is, the lazier is the sensing and the smaller the sensing cost is. As the example demonstrates, there may be a trade-off between the sensing cost of an implementation and its size. Other considerations, like a preference to have eventualities satisfied as soon as possible, enter the picture too.

Our main result is that despite the above intricacy, the sensing cost of an $\omega$-regular language $L$ is the sensing cost of the residual automaton $\mathcal{R}_L$ for $L$. It follows that the sensing cost of an $\omega$-regular language can be computed in polynomial time. 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 DPAs 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 in $n$ copies of $\mathcal{R}_L$ between visits to $\mathcal{A}$, but spends enough time in $\mathcal{A}$ to ensure that the language is $L$. It is worth noting that this result is far from being intuitive. Indeed, first, as mentioned above, the extra expressive power that is added to the setting by the acceptance condition of DPAs makes the residual automaton irrelevant in the context of size minimization. Moreover, in the context of sensing, there need not be a single DPA that attains the minimal sensing cost. It is thus surprising that $\mathcal{R}_L$, which has no acceptance condition, captures the sensing cost of all DPAs. We believe that this reflects a general property of deterministic parity automata that could be useful outside of the scope of sensing. Intuitively, it means that we can "lose track" of the run of a deterministic automaton for arbitrary long periods, just keeping the residual in memory, and still be able to recognize the wanted language.

Due to lack of space, some of the proofs are omitted and can be found in the full version, in the authors' home pages.

## 2 Preliminaries

### 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 finite set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to Q$ is a total transition function, and $\alpha \subseteq Q$ is a set of accepting states. We sometimes refer to $\delta$ 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 $q_{i+1} = \delta(q_i, \sigma_{i+1})$ 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. 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* $u^{-1}L = \{w : uw \in L\}$. When $L$ 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, $\alpha$ contains all classes $[u]$ with $u \in L$. 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 $\delta$ are as in DFA, and $\alpha$ 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 extend the right congruence $\sim_L$ as well as the definition 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.

## 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 $\Sigma$, 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 $\mathit{scost}(q) = |\mathit{sensed}(q)|$. [3]

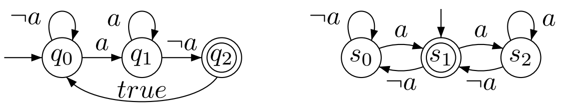
Consider a deterministic automaton $\mathcal{A}$ over $\Sigma = 2^P$ (and over finite or infinite words). For a finite run $r = q_1, \ldots, q_m$ of $\mathcal{A}$, we define the sensing cost of $r$, denoted $\mathit{scost}(r)$, as $\frac{1}{m}\sum_{i=1}^{m} \mathit{scost}(q_i)$. That is, $\mathit{scost}(r)$ 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 $\Sigma$ 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. In fact, as we demonstrate in Example 1 below, in the case of infinite words two different minimal automata for the same language may have different sensing costs.

[3] 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 $|P|$.

For a language $L$ of finite or infinite words, the sensing cost of $L$, denoted $scost(L)$ is the minimal sensing cost required for recognizing $L$ by a deterministic automaton. Thus, $scost(L) = \inf_{\mathcal{A}:L(\mathcal{A})=L} scost(\mathcal{A})$. 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 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 the following figure we present two minimal DBAs (deterministic Büchi automata) for $L$ with different sensing costs.



While all the states of the second 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 7).

▶ **Remark 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). ◀

## 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 $\sigma$ 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 in $\Sigma$ 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 [5], 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 $C$ [5]. 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 $\pi$ 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, whose proof appears in the full version.

▶ **Lemma 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, $\pi(q) = \lim_{m\to\infty} \frac{E_m(q)}{m}$.*

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

▶ **Remark 5.** *It is not hard to see that if $\mathcal{A}$ is strongly connected, then $\pi$ 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 6.** *Given an automaton $\mathcal{A}$, the sensing cost $scost(\mathcal{A})$ can be calculated in polynomial time.*

**Proof.** By Theorem 4, we have that $scost(\mathcal{A}) = \sum_{q\in Q} \pi(q) \cdot scost(q)$, where $\pi$ is the limiting distribution of $\mathcal{A}$. By the definition of $\pi$, 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 $\pi_C$ 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 the full version. ◀

▶ **Example 7.** Recall the first DBA described in Example 1. Its limiting distribution is $\pi(q_0) = \pi(q_1) = \frac{2}{5}$, $\pi(q_2) = \frac{1}{5}$. Accordingly, its cost is $1 \cdot \frac{2}{5} + 1 \cdot \frac{2}{5} + 0 \cdot \frac{1}{5} = \frac{4}{5}$.

Additional examples can be found in the full version.

## 3 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. These last two contributions can be found in the full version.

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 8.** *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 $L(\mathcal{A}^q) = u^{-1}L$. 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 $L(\mathcal{A}^q) = u^{-1}L$, 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 words $u_1$ and $u_2$ 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, $z \in u_1^{-1}L \setminus u_2^{-1}L$. Hence, as $L(\mathcal{A}^q) = u^{-1}L$, 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 $L(\mathcal{A}^{r_i}) = u_i^{-1}L$, 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 $\Sigma^*$, 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 8, enables us to conclude the following.

▶ **Theorem 9.** *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 6 and 9 imply we can efficiently minimize also the sensing cost of a DFA and calculate the sensing cost of its language:

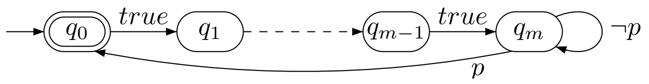
▶ **Theorem 10.** *Given a DFA $\mathcal{A}$, the problem of computing $scost(L(\mathcal{A}))$ can be solved in polynomial time.*

## 4 The Sensing Cost of $\omega$-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 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 11.** 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 the figure below. 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 $\pi$ for $\mathcal{A}_m$ is such that $\pi(q_i) = \frac{1}{m+2}$ for $0 \leq i \leq m-1$ and

$\pi(q_m) = \frac{2}{m+2}$. In the states $q_0, \ldots, q_{m-1}$ the sensing cost is 0 and in $q_m$ it is 1. Accordingly, $scost(\mathcal{A}_m) = \frac{2}{m+2}$, which tends to 0 as $m$ tends to infinity.



## 4.1 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 $\omega$-regular language by means of the residual automaton for the language:

▶ **Theorem 12.** *For every $\omega$-regular language $L \subseteq \Sigma^\omega$, we have $scost(L) = scost(\mathcal{R}_L)$.*

The proof is described over the following section. The first direction, showing that $scost(L) \geq scost(\mathcal{R}_L)$, is proved by similar considerations to those used in the proof of Lemma 8 for the setting of finite words, and can be found in the full version.

Our main effort is to prove that $scost(L) \leq scost(\mathcal{R}_L)$. To show this, we construct, given a DPA $\mathcal{A}$ such that $L(\mathcal{A}) = L$, 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 DBA-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 complemented 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 $q_{❀} \in Q$ such that for every $i \leq r \leq j$, there is an $r$-loop that goes through $q_{❀}$.

The following is an adaptation of a result from [11] to strongly connected DPAs:

▶ **Lemma 13.** *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 exists $m \in \mathbb{N}$ such that $\mathcal{D}$ has ranks $\{0, ..., 2m + k\}$ and has a $[2m, 2m + k]$ flower.*

**Proof.** We start with the following claim, whose proof appears in the full version.

▶ **Claim 14.** *$\mathcal{A}$ does not have an equivalent DPA with ranks $\{1, \ldots, k + 1\}$.*

Now, [11] proves the lemma for $\mathcal{A}$ that needs 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 $q_{❀}$. We argue that since $\mathcal{A}$ is strongly connected, $\mathcal{D}$ has only ranks in $\{0, ..., 2m + k\}$.

By [11], 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 by increasing the ranks by 2 we 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 $q_{❀}$ 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 13, and $q_{❀}$ 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 $\rho = s_1, s_2, ..., s_n$ be the run of $\mathcal{D}$ on $w$. If $\rho$ ends in $q_{❀}$, we define the *$q_{❀}$-loop-abstraction of* $w$ to be the rank-word $\text{abs}(w) \in \Omega^*$ of maximal ranks between successive visits to $q_{❀}$. Formally, let $w = y_0 \cdot y_1 \cdots y_t$ be a partition of $w$ such that $\mathcal{D}$ visits the state $q_{❀}$ after reading the prefix $y_0 \cdots y_j$, for all $0 \leq j \leq t$, and does not visit $q_{❀}$ 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 $u^{-1}L$ 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 $\varphi$ 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 $u^{-1}L \in \langle L \rangle$ a state in $\varphi^{-1}(u^{-1}L)$.

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 $u_4 = 444$, $u_5 = 544454445$, $u_6 = 6544454445654445444456$, 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 $q_{❀}$ and $u_{2m+k}$ is a suffix of $abs(w)$, for the word $u_{2m+k} \in \Omega^*$ defined above. In the full version we describe how to construct $\mathcal{P}$, essentially by combining a DFA over that alphabet $\Omega$ 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 $q_{❀}$ and thus abstracts words in $\Sigma^*$.

We can now turn to the construction of the DPAs $\mathcal{B}_n$. 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 $\langle (q, t), a, (\varphi(q'), 1) \rangle \in \Delta_n$.

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 $([u], i) \in \langle L \rangle \times \{1, \ldots, n\}$.

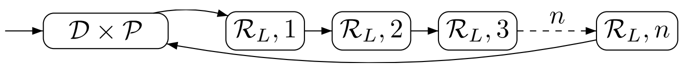


**Figure 1** The DPA $\mathcal{B}_n$.

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 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 15, 16, and 17.

▶ **Lemma 15.** *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, $L(\mathcal{D}^q) = L(\mathcal{A}^q) = u^{-1}L$.*

**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 $L(\mathcal{A}^q) = u^{-1}L$. The proof proceeds by induction on $|u|$ and is detailed in the full version. By Lemma 13, for every $q \in Q$, we have $L(\mathcal{A}^q) = L(\mathcal{D}^q)$, so the claim follows. ◀

▶ **Lemma 16.** *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 15, we have $L(\mathcal{A}^q) = u^{-1}L$. 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. This is where the special structure of $\mathcal{P}$ guarantees that the sparse visits in the $\mathcal{D}$-component are sufficient for determining acceptance.

▶ **Lemma 17.** *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 $\tau = s_1, s_2, s_3, \dots$ be the run of $\mathcal{B}_n$ on $w$ and let $\rho = q_1, q_2, q_3 \dots$ be the run of $\mathcal{A}$ on $w$. We denote by $\tau[i,j]$ the infix $s_i, \dots, s_j$ of $\tau$. We also extend $\alpha_{\mathcal{D}}$ to (infixes of) runs by defining $\alpha_{\mathcal{D}}(\tau[i,j]) = \alpha_{\mathcal{D}}(s_i), \dots, \alpha_{\mathcal{D}}(s_j)$. 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 $0 \le i \le j$, is a part of a run of $\mathcal{D}$ that consists of loops around $q_{❀}$, we define the *loop type of $v$* to be the word in $\Omega^*$ that describes the highest rank of each simple loop around $q_{❀}$ in $v$. An infix of $\tau$ whose loop type is $u_i$ for some $2m \le i \le 2m+k$ is called a *$u_i$-loop-infix.*

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

Let $i \in \{2m, \dots, 2m+k\}$, 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 $\tau$, 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 $\rho$ in its part that corresponds to $v$.

By our choice of $M$, we can find two indices $c \le j < l \le d$ such that the pairs $\langle (q_j, t), q'_j \rangle$ and $\langle (q_l, t'), q'_l \rangle$ reached by $(\tau, \rho)$ in indices $j$ and $l$, respectively, satisfy $q_j = q_l = q_{❀}$ 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 $\tau$ is $i$. We write $v = v_1 v_2 v_3$, 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 $\mu$ and $\eta$ of $\mathcal{D}^{q_j}$ and of $\mathcal{A}^{q'_j}$ on $v_2^\omega$, respectively. These runs are loops labeled by $v_2$, where the highest rank in $\mu$ is $i$. By Lemma 15, $L(\mathcal{D}^{q_j}) = L(\mathcal{D}^{q'_j}) = L(\mathcal{A}^{q'_j})$, so the highest rank in $\eta$ must have same parity (odd or even) as $i$.

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

We want to show that rank $k$ is witnessed on $\rho$ during every $u_{2m+k}$-infix of $\tau$. Assume by way of contradiction that this is not the case. This means that there is some $u_{2m+k}$-infix $v'$ in $\tau$ such that all ranks visited in $\rho$ 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 $\rho$ is of the same parity as $k-1$. As $v''$ is also an infix of $v'$, the highest rank in $\rho$ 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 $\rho$ witnesses a rank $k$ in any $u_k$-infix of $\tau$. Since $\tau$ contains infinitely many $u_k$-infixes, then $\rho$ contains infinitely many ranks $k$, and, depending on the parity of $k$, either both $\rho$ and $\tau$ 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 18.** $\lim_{n\to\infty} scost(\mathcal{B}_n) = scost(\mathcal{R}_L)$.

**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. This implies that $\mathcal{B}_n$ is strongly connected, and therefore, 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 in the full version. ◀

Lemmas 16, and 17 put together ensure that for strongly connected automata, we have that $L(\mathcal{B}_n) = L$, so with Lemma 18, we get $scost(L) = scost(\mathcal{R}_L)$.

It is left to remove the assumption about $\mathcal{A}$ being strongly connected. The proof is detailed in the full version, and uses the above result on each ergodic component of $\mathcal{A}$.

▶ **Remark 19.** *All our results can be easily extended to a setting with a non-uniform distribution on the letters given by any Markov chain, or with a different cost for each input in each state. We can also use a decision tree to read the inputs instead of reading them* *simulatenously, defining for instance a cost of* 1.5 *if the state starts by reading a, then if a is true it also reads b.* ◀

## 5 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. In the setting of open systems, we partition $P$ into a set $I$ of input signals, generated by the environment, and a set $O$ of output signals, generated by the system. Then, we define the sensing cost of a specification as the minimal sensing cost required for a transducer that realizes it, where 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 than those presented in the paper.

**Trade-off between sensing and quality:** The key idea in the proof of Theorem 12 is that when we reason about languages of infinite words, it is sometimes possible to delay the sensing and only sense in "sparse" intervals. In practice, however, it is often desirable to satisfy eventualities quickly. 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 $\sum_{i\geq 0} 2^{-i} \cdot sensed(|q_i|)$ of the sensing costs of the states $q_0, q_1, \ldots$ it visits.

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

## References

1 S. Almagor, U. Boker, and O. Kupferman. Discounting in LTL. In *TACAS*, Lecture Notes in Computer Science. Springer, 2014.

2 K. Chatterjee and R. Majumdar. Minimum attention controller synthesis for omega-regular objectives. In *FORMATS*, pages 145–159, 2011.

3 K. Chatterjee, R. Majumdar, and T. A. Henzinger. Controller synthesis with budget constraints. In *HSCC*, volume 4981 of *Lecture Notes in Computer Science*, pages 72–86. Springer, 2008.

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

**5** C. Grinstead and J. Laurie Snell. *Introduction to Probability*, chapter 11 (Markov Chains), pages 405–470. American Mathematical Society, 1997.

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

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

**8** E. Kushilevitz and N. Nisan. *Communication complexity.* Cambridge University Press, 1997.

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

**10** S. Muthukrishnan. Theory of data stream computing: where to go. In *Proc. 30th Symposium on Principles of Database Systems*, pages 317–319, 2011.

**11** D. Niwinski and I. Walukiewicz. Relating hierarchies of word and tree automata. In *STACS*, volume 1373 of *Lecture Notes in Computer Science.* Springer, 1998.

**12** S. Schewe. Beyond Hyper-Minimisation – Minimising DBAs and DPAs is NP-Complete. In *FSTTCS*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 400–411, 2010.