Sensing as a Complexity Measure[⋆]

Shaull Almagor, Denis Kuperberg, and Orna Kupferman

[1] Department of Computer Science, Oxford University, UK
[2] CNRS, ENS Lyon, Universit de Lyon, LIP
[3] School of Engineering and Computer Science, The Hebrew University, Israel

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. We study the sensing cost of regular and ω-regular languages, as well as applications of the study in practice, especially in the monitoring and synthesis of reactive systems.

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) [12] or communication complexity (number and length of messages required) [11]. 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 [9]. 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" [13].

Our interest in regular sensing is motivated by the use of finite-state automata 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

[⋆] The paper gives an overview of the technical results in the papers [2] and [3]. The research leading to these results has received funding from the European Research Council under the European Union's 7th Framework Programme (FP7/2007-2013, ERC grant no 278410).

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.

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 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 [15]. 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.

To see the intricacy in the case of ω-regular languages, consider a component in a vacuum-cleaning robot that monitors the dust collector and checks that it is empty

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

[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 [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.

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 ω-regular language L is the sensing cost of the residual automaton $\mathcal{R}_L$ for L. It follows that the sensing cost of an ω-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 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.

In the context of formal methods, sensing has two appealing applications. The first is *monitoring*: we are given a computation and we have to decide whether it satisfies a specification. When the computations are over 2^P, we want to design a monitor that minimizes the expected average number of sensors used in the monitoring process. Monitoring is especially useful when reasoning about *safety* specifications [8]. There, every computation that violates the specification has a bad prefix – one all whose extensions are not in L. Hence, as long as the computation is a prefix of some word in L, the monitor continues to sense and examine the computation. Once a bad prefix is detected, the monitor declares an error and no further sensing is required. The second application is *synthesis*. Here, the set P of signals is partitioned into sets I and O of input and output signals, respectively. We are given a specification L over the alphabet $2^{I \cup O}$, and our goal is to construct an *I/O transducer* that realizes L. That is, for every sequence of assignments to the input signals, the transducer generates a sequence of assignments to the output signals so that the obtained computation is in L [14]. Our goal is to construct a transducer that minimizes the expected average number of sensors (of input signals) that are used along the interaction.

The definition of sensing cost described above falls short in the above two applications. For the first, the definition above does not distinguish between words in the language and words not in the language, whereas in monitoring we care only for words in the language. In particular, according to the definition above, the sensing cost of a safety language is always 0. For the second, the definition above considers automata and does not partition P into I and O, whereas synthesis refers to I/O-transducers. Moreover, unlike automata, correct transducers generate only computations in the language, and they need not generate all words in the language – only these that ensure receptiveness with respect to all sequences of inputs.

We thus continue and study sensing in the context of monitoring and synthesis. We suggest definitions that capture the intuition of "required number of sensors" in these settings and solve the problems of generating monitors and transducers that minimize sensing. For both settings, we focus on safety languages.

Consider, for example, a traffic monitor that has access to various sensors on roads and whose goal is to detect accidents. Once a road accident is detected, an alarm is raised to the proper authorities and the monitoring is stopped until the accident has been taken care of. The monitor can read the speed of cars along the roads, as well as the state of traffic lights. An accident is detected when some cars do not move eventhough no traffic light is stopping them. Sensing the speed of every car and checking every traffic light requires huge sensing. Our goal is to find a monitor that minimizes the required sensing and still detects all accidents. In the synthesis setting, our goal is extended to designing a transducer that controls the traffic lights according to the speed of the traffic in each direction, and satisfies some specification (say, give priority to slow traffic), while minimizing the sensing of cars.

We revise our definition as follows. Let us start with monitoring. Recall that the definition of sensing above assumes a uniform probability on the assignments to the signals, whereas in monitoring we want to consider instead more intricate probability spaces – ones that restrict attention to words in the language. As we show, there is more than one way to define such probability spaces, each leading to a different measure. We study two such measures. In the first, we sample a word randomly, letter by letter, according to a given distribution, allowing only letters that do not generate bad prefixes. In the second, we construct a sample space directly on the words in the language. We show that in both definitions, we can compute the sensing cost of the language in polynomial time, and that the minimal sensing cost is attained by a minimal-size automaton. Thus, luckily enough, even though different ways in which a computation may be given in an online manner calls for two definitions of sensing cost, the design of a minimally-sensing monitor is the same in the two definitions.

Let us continue to synthesis. Recall that there, 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 [10, 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.

The main challenge there is that we no longer need to consider all words in the language. This introduces a new degree of freedom, which requires different techniques than those used for the definition above. In particular, while a minimal-size transducer for a safety language can be defined on top of the state space of a minimal-size deterministic automaton for the language, this is not the case when we seek minimally-sensing transducers. In fact, we show that a minimally-sensing transducer for a safety language might be exponentially bigger than a minimal-size automaton for the language. Consequently, the problems of computing the minimal sensing cost and finding a minimally-sensing transducer are EXPTIME-complete even for specifications given by means of deterministic safety automata. On the positive side, a transducer that attains the minimal sensing cost always exists for safety specifications.

2 Preliminaries

Automata A *deterministic automaton on finite words* (DFA) 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 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 $q_{i+1} = \delta(q_i, \sigma_{i+1})$ 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, α 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 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 $\mathit{inf}(r) \cap \alpha \neq \emptyset$. A *looping* automaton is a special case of the Büchi condition, with $\alpha = Q$. Finally, a 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, NLA, DLA, NPA, and DPA to denote nondeterministic/deterministic Büchi/looping/parity word automata.

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 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 $sensed(q)$ to denote the set of signals sensed in q. The *sensing cost* of a state $q \in Q$ is $scost(q) = |sensed(q)|$. [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 $scost(r)$, as $\frac{1}{m} \sum_{i=0}^{m-1} scost(q_i)$. That is, $scost(r)$ 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 $scost_{\mathcal{A}}(w)$, 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, $scost(\mathcal{A}) = \lim_{m \to \infty} |\Sigma|^{-m} \sum_{w : |w| = m} scost_{\mathcal{A}}(w)$. 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.

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 for L with different sensing costs.

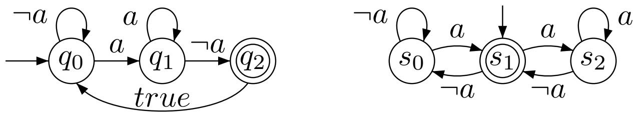

Fig. 1. Two minimal DBAs for L with different sensing costs.

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). $\square$

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

Remark 1. 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). $\square$

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 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 1. *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 1. *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 $\mathcal{A}$.*

Remark 2. 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. $\square$

Theorem 2. *Given an automaton $\mathcal{A}$, the sensing cost $scost(\mathcal{A})$ can be calculated in polynomial time.*

Example 2. Let $P = \{a, b\}$. Consider the DFA $\mathcal{A}_1$ appearing in Figure 2. Note that $L(\mathcal{A}_1) = (\texttt{True})^* \cdot a \cdot b \cdot (\texttt{True})^*$. 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, $scost(\mathcal{A}_1) = 0$.

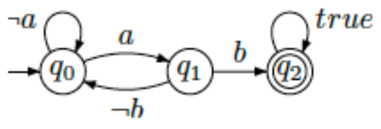

Fig. 2. The DFA $\mathcal{A}_1$.

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 $scost(q_2) = 2$.

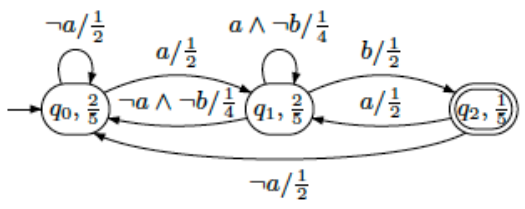

Fig. 3. The DFA $\mathcal{A}_2$ and its corresponding Markov chain.

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 $\pi(q_i)$:

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

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 $1 \cdot \frac{2}{5} + 2 \cdot \frac{2}{5} + 1 \cdot \frac{1}{5} = \frac{7}{5}$. $\square$

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.

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

Since $L(\mathcal{R}_L) = L$, then $scost(L) \leq scost(\mathcal{R}_L)$. This, together with Lemma 2, enables us to conclude the following.

Theorem 3. *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 and 3 imply we can efficiently minimize also the sensing cost of a DFA and calculate the sensing cost of its language:

Theorem 4. *Given a DFA $\mathcal{A}$, the problem of computing $scost(L(\mathcal{A}))$ can be solved in polynomial time.*

4 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 1 we saw that different minimal DBAs may have a different sensing cost. We start 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 3. 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. hfill $\square$

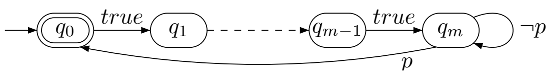

Fig. 4. The DBA $\mathcal{A}_m$.

Still we can characterize the sensing cost of an ω-regular language by means of the residual automaton for the language:

Theorem 5. *For every ω-regular language $L \subseteq \Sigma^\omega$, we have $scost(L) = scost(\mathcal{R}_L)$.*

Trade-off between sensing and quality: The key idea in the proof of Theorem 5 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 3). 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.

5 Monitoring

As described in Section 1, the definition of sensing above takes into an account all words in $(2^P)^\omega$, regardless their membership in the language. In monitoring, we restrict attention to words in the language, as once a violation is detected, no further sensing is required. In particular, in safety languages, violation amounts to a detection of a bad prefix, and indeed safety languages are the prominent class of languages for which monitoring is used [8].

As it turns out, however, there are many approaches to define the corresponding probability space. We suggest here two. We focus on safety languages, namely these recognizable by DLAs. Let $\mathcal{A}$ be a DLA and let $L = L(\mathcal{A})$.

1. **[Letter-based]** At each step, we uniformly draw a "safe" letter – one with which we are still generating a word in $pref(L)$, thereby iteratively generating a random word in L.
2. **[Word-based]** At the beginning, we uniformly draw a word in L.

We denote the sensing cost of $\mathcal{A}$ in the letter- and word-based approaches $lcost(\mathcal{A})$ and $wcost(\mathcal{A})$, respectively. The two definitions yield two different probability measures on L, as demonstrated in Example 4 below.

Example 4. Let $P = \{a\}$ and consider the safety language $L = a^\omega + (\neg a) \cdot (\mathit{True})^\omega$. That is, if the first letter is $\{a\}$, then the suffix should be $\{a\}^\omega$, and if the first letter is $\emptyset$, then all suffixes result in a word in L. Consider the DLA $\mathcal{A}$ for L in Figure 5.

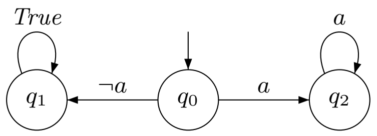

Fig. 5. A DLA for $a^\omega + (\neg a) \cdot (\mathit{True})^\omega$.

In the letter-based definition, we initially draw a letter from $2^{\{a\}}$ uniformly, i.e., either a or $\neg a$ w.p. $\frac{1}{2}$. If we draw $\neg a$, then we move to q_1 and stay there forever. If we draw a, then we move to q_2 and stay there forever. Since $scost(q_1) = 0$ and $scost(q_2) = 1$, and we reach q_1 and q_2 w.p $\frac{1}{2}$, we get $lcost(\mathcal{A}) = \frac{1}{2}$.

In the word-based definition, we assign a uniform probability to the words in L. In this case, almost all words are not a^ω, and thus the probability of a^ω is 0. This means that we will get to q_1 w.p. 1, and thus $wcost(\mathcal{A}) = 0$. $\square$

As a more realistic example, recall our traffic monitor in Section 1. There, the behavior of the cars is the random input, and the two approaches can be understood as follows. In the letter-based approach, we assume that the drivers do their best to avoid accidents regardless of the history of the traffic and the traffic lights so far. Thus, after every safe prefix, we assume that the next input is also safe. In the word-based approach, we assume that the city is planned well enough to avoid accidents. Thus, we a-priori set the distribution to safe traffic behaviors according to their likelihood.

We now define the two approaches formally.

The Letter-Based Approach Consider a DLA $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, Q \rangle$. For a state $q \in Q$, let $avail(q)$ be the set of letters available in q, namely letters that do not cause $\mathcal{A}$ to get stuck. Formally, $avail(q) = \{\sigma \in \Sigma : \delta(q, \sigma) \text{ is defined }\}$. We model the drawing of available letters by the Markov chain $\mathcal{M}_\mathcal{A} = \langle Q, P \rangle$, where the probability of a transition from state q to state q' in $\mathcal{M}_\mathcal{A}$ is $P(q, q') = \frac{|\{\sigma \in \Sigma : \delta(q,\sigma) = q'\}|}{|avail(q)|}$. Let π be the limiting distribution of $\mathcal{M}_\mathcal{A}$. We define $lcost(\mathcal{A}) = \sum_{q \in Q} \pi(q) \cdot scost(q)$.

Since computing the limiting distribution can be done in polynomial time, we have the following.

Theorem 6. *Given a* DLA $\mathcal{A}$*, the sensing cost* $lcost(\mathcal{A})$ *can be calculated in polynomial time.*

The Word-Based Approach Consider a DLA $\mathcal{A} = \langle 2^P, Q, q_0, \delta, Q \rangle$ recognizing a non-empty safety language L. From Section 2, $scost(\mathcal{A}) = \lim_{n\to\infty} \frac{1}{|\Sigma|^n} \sum_{u\in\Sigma^n} scost_{\mathcal{A}}(u)$, which is proven to coincide with $\mathbb{E}[scost_{\mathcal{A}}(u)]$ where $\mathbb{E}$ is the expectation with respect to the standard measure on Σ^ω. Our goal here is to replace this standard measure with one that restricts attention to words in L. Thus, we define $wcost(\mathcal{A}) = \mathbb{E}[scost(u) \mid u \in L]$. For $n \geq 0$, let $pref(L, n)$ be the set of prefixes of L of length n. Formally, $pref(L, n) = pref(L) \cap \Sigma^n$. As in the case of the standard measure, the expectation-based definition coincides with one that that is based on a limiting process: $wcost(\mathcal{A}) = \lim_{n\to\infty} \frac{1}{|pref(L,n)|} \sum_{u\in pref(L,n)} scost_{\mathcal{A}}(u)$. Thus, the expressions for $scost$ and $wcost$ are similar, except that in the expectation-based definition we add conditional probability, restricting attention to words in L, and in the limiting process we replace Σ^n by $pref(L, n)$.

Note that the term $\frac{1}{|pref(L,n)|}$ is always defined, as L is a non-empty safety language. In particular, the expectation is well defined even if L has measure 0 in Σ^ω.

Theorem 7. *Given a DLW $\mathcal{A}$, we can compute $wcost(\mathcal{A})$ in polynomial time.*

References

1. S. Almagor, U. Boker, and O. Kupferman. Discounting in LTL. In *Proc. 20th TACAS*, LNCS 8413, pages 424–439. Springer, 2014.
2. S. Almagor, D. Kuperberg, and O. Kupferman. Regular sensing. In *Proc. 34th FST & TCS*, *LIPIcs* 29, pages 161–173, 2014.
3. S. Almagor, D. Kuperberg, and O. Kupferman. The sensing cost of monitoring and synthesis. In *Proc. 34th FST & TCS*, *LIPIcs* 35, pages 380–393, 2015.
4. K. Chatterjee and R. Majumdar. Minimum attention controller synthesis for omega-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 International Workshop on Hybrid Systems: Computation and Control*, LNCS 4981, pages 72–86. Springer, 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. K. Havelund and G. Rosu. Efficient monitoring of safety properties. *STT&T*, 6(2):18–173, 2004.
9. G. Kindler. *Property Testing, PCP, and Juntas*. PhD thesis, Tel Aviv University University, 2002.
10. O. Kupferman and M.Y. Vardi. Church's problem revisited. *The Bulletin of Symbolic Logic*, 5(2):245 – 263, 1999.
11. E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.
12. C. Mauduit and A. Sárköz. On finite pseudorandom binary sequences. i. measure of pseudorandomness, the legendre symbol. *Acta Arith.*, 82(4):365–377, 1997.
13. S. Muthukrishnan. Theory of data stream computing: where to go. In *Proc. 30th PODS*, pages 317–319, 2011.
14. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th POPL*, pages 179–190, 1989.
15. S. Schewe. Beyond Hyper-Minimisation—Minimising DBAs and DPAs is NP-Complete. In *Proc. 30th FST & TCS*, LIPIcs 8, pages 400–411, 2010.