The Sensing Cost of Monitoring and Synthesis

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

1 **The Hebrew University, Israel**
2 **IRIT/Onera, Toulouse, France**

Abstract

In [2], we introduced *sensing* as a new complexity measure for the complexity of regular languages. Intuitively, the sensing cost quantifies the detail in which a random input word has to be read by a deterministic automaton in order to decide its membership in the language. In this paper, we consider sensing in two principal applications of deterministic automata. The first is *monitoring*: we are given a computation in an on-line manner, and we have to decide whether it satisfies the specification. The second is *synthesis*: we are given a sequence of inputs in an on-line manner and we have to generate a sequence of outputs so that the resulting computation satisfies the specification. In the first, our goal is to design a monitor that handles all computations and minimizes the expected average number of sensors used in the monitoring process. In the second, our goal is to design a transducer that realizes the specification for all input sequences and minimizes the expected average number of sensors used for reading the inputs.

We argue that the two applications require new and different frameworks for reasoning about sensing, and develop such frameworks. We focus on safety languages. We show that for monitoring, minimal sensing is attained by a monitor based on the minimal deterministic automaton for the language. For synthesis, however, the setting is more challenging: minimizing the sensing may require exponentially bigger transducers, and the problem of synthesizing a minimally-sensing transducer is EXPTIME-complete even for safety specifications given by deterministic automata.

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, ω-regular languages, complexity, sensing, minimization

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2015.380

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 requirements. Theoretical interest as well as practical considerations have motivated additional measures, such as randomness (the number of random bits required for the execution) [11] or communication complexity (number and length of messages required) [10]. For 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.

In [2], we introduced and studied a new complexity measure, namely the *sensing cost* of the language. Intuitively, the sensing cost of a language measures the detail with which a random input word needs to be read in order to decide membership in the language. Sensing has been studied in several other CS contexts. In theoretical CS, in methodologies such as PCP and property testing, we are allowed to sample or query only part of the input [8]. In more practical applications, mathematical tools in signal processing are used to reconstruct information based on compressed sensing [3], and in the context of data streaming, one

35th IARCS Annual Conf. Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015).
Editors: Prahladh Harsha and G. Ramalingam; pp. 380–393

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

cannot store in memory the entire input, and therefore has to approximate its properties according to partial "sketches" [12].

Our study in [2] considered regular and ω-regular languages, where sensing is defined as follows. Consider a deterministic automaton $\mathcal{A}$ over an alphabet 2^P, for a finite set P of *signals*. 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 given distribution of the inputs. The sensing cost of a language with respect to this distribution is then the infimum sensing cost of an automaton for the language. For simplicity, we focus on the uniform distribution, and we refer to the sensing cost of an automaton without parameterizing it by a distribution. As detailed in Remark 1, all our results can be extended to a setting with a parameterized distribution.

In [2], we showed that computing the sensing cost of a language can be done in polynomial time. We further showed that while in finite words the minimal sensing cost is always attained, this is not the case for infinite words. For example, recognizing the language L over $2^{\{p\}}$ of all words with infinitely many p's, one can give up sensing of p for unboundedly-long intervals, thus the sensing cost of L is 0, yet every deterministic automaton $\mathcal{A}$ that recognizes L must sense p infinitely often, causing the sensing cost of $\mathcal{A}$ to be strictly greater than 0.

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 [7]. 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 [13]. 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 in [2] falls short in the above two applications. For the first, the definition in [2] 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 in [2], the sensing cost of a safety language is always 0. For the second, the definition in [2] 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.

In this work we 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 even-though 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 can now describe our model and results. Let us start with monitoring. Recall that the definition of sensing in [2] 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.

Next, we proceed to study sensing for synthesis. The main challenge there is that we no longer need to consider all words in the language. Also, giving up sensing has a flavor of synthesis with incomplete information [9]: the transducer has to realize the specification no matter what the incomplete information is. This introduces a new degree of freedom, which requires different techniques than those used in [2]. 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. This is different also from the results in [2] and even these in the monitoring setting, where a minimally-sensing automaton or monitor for a safety language coincides with the minimal-size automaton for it. 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. For general ω-regular specifications, even decidability of computing the optimal sensing cost remains open.

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 and Transducers

A *deterministic automaton on infinite words* 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, $\delta : Q \times \Sigma \nrightarrow Q$ is a partial transition function, and α is an acceptance condition. We sometimes refer to δ as a relation $\Delta \subseteq Q \times \Sigma \times Q$, with $\langle q, \sigma, q' \rangle \in \Delta$ iff $\delta(q, \sigma) = q'$. A run of $\mathcal{A}$ on a word $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is a sequence of states $q_0, q_1, \ldots$ such that $q_{i+1} = \delta(q_i, \sigma_{i+1})$ for all $i \geq 0$. Note that since δ is deterministic and partial, $\mathcal{A}$ has at most one run on a word. A run is accepting if it satisfies the acceptance

condition. A word $w \in \Sigma^\omega$ is accepted by $\mathcal{A}$ if $\mathcal{A}$ has an accepting run on w. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. We denote by $\mathcal{A}^q$ the automaton $\mathcal{A}$ with the initial state set to q.

In a deterministic *looping* automaton (DLW), every run is accepting. Thus, a word is accepted if there is a run of the automaton on it.[1] Since every run is accepting, we omit the acceptance condition and write $\mathcal{A} = \langle \Sigma, Q, q_0, \delta \rangle$.

For finite sets I and O of input and output signals, respectively, an I/O *transducer* is $\mathcal{T} = \langle I, O, Q, q_0, \delta, \rho \rangle$, where Q is a finite set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times 2^I \to Q$ is a total transition function, and $\rho : Q \to 2^O$ is a labeling function on the states. The run of $\mathcal{T}$ on a word $w = i_0 \cdot i_1 \cdots \in (2^I)^\omega$ is the sequence of states $q_0, q_1, \ldots$ such that $q_{k+1} = \delta(q_k, i_k)$ for all $k \geq 0$. The *output* of $\mathcal{T}$ on w is then $o_1, o_2, \ldots \in (2^O)^\omega$ where $o_k = \rho(q_k)$ for all $k \geq 1$. Note that the first output assignment is that of q_1, and we do not consider $\rho(q_0)$. This reflects the fact that the environment initiates the interaction. The *computation of* $\mathcal{T}$ *on* w is then $\mathcal{T}(w) = i_0 \cup o_1, i_1 \cup o_2, \ldots \in (2^{I \cup O})^\omega$.

Note that the structure of each I/O-transducer $\mathcal{T}$ induces a DLW $\mathcal{A}_\mathcal{T}$ over the alphabet 2^I with a total transition relation. Thus, the language of the DLW is $(2^I)^\omega$, reflecting the receptiveness of $\mathcal{T}$.

Safety Languages

Consider a language $L \subseteq \Sigma^\omega$. A finite word $x \in \Sigma^*$ is a *bad prefix* for L if for every $y \in \Sigma^\omega$, we have that $x \cdot y \notin L$. That is, x is a bad prefix if all its extensions are words not in L. The language L is then a *safety* language if every word not in L has a bad prefix. For a language L, let $\mathit{pref}(L) = \{x \in \Sigma^* : \text{ there exists } y \in \Sigma^\omega \text{ such that } x \cdot y \in L\}$ be the set of prefixes of words in L. Note that each word in Σ^* is either in $\mathit{pref}(L)$ or is a bad prefix for L. Since the set $\mathit{pref}(L)$ for a safety language L is *fusion closed* (that is, a word is in L iff all its prefixes are in $\mathit{pref}(L)$), an ω-regular language is safety iff it can be recognized by a DLW [15].

Consider a safety language L over sets I and O of input and output signals. We say that L is I/O*-realizable* if there exists an I/O transducer $\mathcal{T}$ all whose computations are in L. Thus, for every $w \in (2^I)^\omega$, we have that $\mathcal{T}(w) \in L$. We then say that $\mathcal{T}$ I/O*-realizes* L. When I and O are clear from the context, we omit them. The *synthesis* problem gets as input a safety language L over $I \cup O$, say by means of a DLW, and returns an I/O-transducer that realizes L or declares that L is not I/O-realizable.

Sensing

In [2], we defined regular sensing as a measure for the number of sensors that need to be operated in order to recognize a regular language. 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, and sensing a signal amounts to knowing its assignment. Describing sets of letters in Σ, it is convenient to use Boolean assertions over P. For example, when $P = \{a, b\}$, the assertion $\neg b$ stands for the set $\{\emptyset, \{a\}\}$ of two letters.

For completeness, we bring here the definitions from [2]. Consider a language L and a deterministic automaton $\mathcal{A} = \langle 2^P, Q, q_0, \delta, \alpha \rangle$ such that $L(\mathcal{A}) = L$. We assume that δ is total. 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

[1] For readers familiar with the Büchi acceptance condition, a looping automaton is a special case of Büchi with $\alpha = Q$.

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 $scost(q) = |sensed(q)|$. [2]

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 (see Remark 1 below). Thus, $scost(\mathcal{A}) = \lim_{m\to\infty} |\Sigma|^{-m} \sum_{w\in\Sigma^m} scost_{\mathcal{A}}(w)$.

Note that the definition applies to automata on both finite and infinite words, and it corresponds to the closed setting: the automaton gets as input words over 2^P and uses sensors in order to monitor the input words and decide their membership in L. We define the *sensing cost* of a language L to be the minimal cost of an automaton for L. A-priori, the minimal cost might not be attained by a single automaton, thus we define $scost(L) = \inf\{scost(\mathcal{A}) : \mathcal{A} \text{ is an automaton for } L\}$.

▶ **Remark 1** (On the choice of uniform distribution). *The choice of a uniform distribution on the letters in Σ may be unrealistic in practice. Indeed, in real scenarios, the distribution on the truth assignments to the underlying signals may be complicated. Generally, such a distribution can be given by a Markov chain (in monitoring) or by an MDP (in synthesis). As it turns out, adjusting our setting and algorithms to handle such distributions involves only a small technical elaboration, orthogonal to the technical challenges that exists already in a uniform distribution.*

Accordingly, throughout the paper we assume a uniform distribution on the truth assignments to the signals. In the full version we describe how our setting and algorithms are extended to the general case.

The definition of sensing in [2] essentially considers the sensing required in the Ergodic SCC of a deterministic automaton for the language. Since in safety languages, the Ergodic SCCs are accepting or rejecting sinks, which require no sensing, we have the following, which implies that the definition in [2] is not too informative for safety languages.

▶ **Lemma 2.** *For every safety language $L \subseteq \Sigma^\omega$, we have $scost(L) = 0$.*

Markov Chains and Decision Processes

A Markov chain $\mathcal{M} = \langle S, P\rangle$ consists of a finite state space S and a stochastic transition matrix $P : S \times S \to [0,1]$. That is, for all $s \in S$, we have $\sum_{s'\in S} P(s,s') = 1$. Given an initial state s_0, consider the vector v^0 in which $v^0(s_0) = 1$ and $v^0(s) = 0$ for every $s \neq s_0$. The *limiting distribution* of $\mathcal{M}$ is $\lim_{n\to\infty} \frac{1}{n}\sum_{m=0}^{n} v^0 P^m$. The limiting distribution satisfies $\pi P = \pi$, and can be computed in polynomial time [5].

A Markov decision process (MDP) is $\mathcal{M} = \langle S, s_0, (A_s)_{s\in S}, \mathrm{P}, cost\rangle$ where S is a finite set of states, $s_0 \in S$ is an initial state, A_s is a finite set of actions that are available in state $s \in S$. Let $A = \bigcup_{s\in S} A_s$. Then, $\mathrm{P} : S \times A \times S \nrightarrow [0,1]$ is a partial transition probability function, defining for every two states $s, s' \in S$ and action $a \in A_s$, the probability of moving from s to s' when action a is taken. Accordingly, $\sum_{s'\in S} \mathrm{P}(s,a,s') = 1$. Finally, $cost : S \times A \nrightarrow \mathbb{N}$ is a

[2] We note that, alternatively, one could define the *sensing level* of states, with $slevel(q) = \frac{scost(q)}{|P|}$. 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 $|P|$.

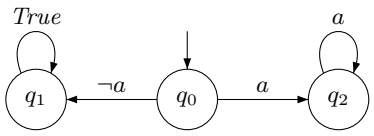

Figure 1 A DLW for $a^\omega + (\neg a) \cdot (True)^\omega$.

partial cost function, assigning each state s and action $a \in A_s$, the cost of taking action a in state s.

An MDP can be thought of as a game between a player who chooses the actions and nature, which acts stochastically according to the transition probabilities.

A *policy* for an MDP $\mathcal{M}$ is a function $f : S^* \times S \to A$ that outputs an action given the history of the states, such that for $s_0, \ldots, s_n$ we have $f(s_0, \ldots, s_n) \in A_{s_n}$. Policies correspond to the strategies of the player. The *cost* of a policy f is the expected average cost of a random walk in $\mathcal{M}$ in which the player proceeds according to f. Formally, for $m \in \mathbb{N}$ and for a sequence of states $\tau = s_0, \ldots, s_{m-1}$, we define $\mathrm{P}_f(\tau) = \prod_{i=1}^{m-1} \mathrm{P}(s_{i-1}, f(s_0 \cdots s_{i-1}), s_i)$. Then, $cost_m(f, \tau) = \frac{1}{m} \sum_{i=1}^{m} cost(s_i, f(s_1 \cdots s_i))$ and we define the cost of f as $cost(f) = \liminf_{m\to\infty} \frac{1}{m} \sum_{\tau:|\tau|=m} cost_m(f, \tau) \cdot \mathrm{P}_f(\tau)$.

A policy is *memoryless* if it depends only on the current state. We can describe a memoryless policy by $f : S \to A$. A memoryless policy f induces a Markov chain $\mathcal{M}^f = \langle S, P_f \rangle$ with $P_f(s, s') = \mathrm{P}(s, f(s), s')$. Let π be the limiting distribution of $\mathcal{M}^f$. It is not hard to prove that $cost(f) = \sum_{s\in S} \pi_s cost(s, f(s))$. Let $cost(\mathcal{M}) = \inf\{cost(f) : f \text{ is a policy for } \mathcal{M}\}$. That is, $cost(M)$ is the expected cost of a game played on $\mathcal{M}$ in which the player uses an optimal policy.

▶ **Theorem 3.** *Consider an MDP $\mathcal{M}$. Then, $cost(\mathcal{M})$ can be attained by a memoryless policy, which can be computed in polynomial time.*

3 Monitoring

As described in Section 2, the definition of sensing in [2] 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 [7].

As it turns out, however, there are many approaches to define the corresponding probability space. We suggest here two. Let $\mathcal{A}$ be a DLW 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 $L = a^\omega + (\neg a) \cdot (True)^\omega$. 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 DLW $\mathcal{A}$ for L in Figure 1.

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 $wcost(\mathcal{A}) = 0$.

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 DLW $\mathcal{A} = \langle \Sigma, Q, \delta, q_0 \rangle$. 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 5.** *Given a* DLW $\mathcal{A}$*, the sensing cost* $lcost(\mathcal{A})$ *can be calculated in polynomial time.*

The Word-Based Approach

Consider a DLW $\mathcal{A} = \langle 2^P, Q, q_0, \delta \rangle$ recognizing a non-empty safety language L. From [2], we have $scost(\mathcal{A}) = \lim_{n\to\infty} \frac{1}{|\Sigma|^n} \sum_{u \in \Sigma^n} scost_\mathcal{A}(u)$, which coincides 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 6.** *Given a DLW* $\mathcal{A}$*, we can compute* $wcost(\mathcal{A})$ *in polynomial time.*

Proof. We will use here formal power series on one variable z, a classical tool for graph and automata combinatorics. They can be thought of as polynomials of infinite degree.

For states $p, q \in Q$ and for $n \in \mathbb{N}$, let $\#paths(p, q, n)$ denote the number of paths (each one labeled by a distinct word) of length n from p to q in $\mathcal{A}$. We define the generating functions:

$C_{p,q}(z) = \sum_{n\in\mathbb{N}} \#paths(p,q,n)z^n$ and $F_q(z) = C_{q_0,q}(z)\sum_{p\in Q} C_{q,p}(z)$. Let $[z^n]F_q(z)$ be the coefficient of z^n in $F_q(z)$. By the definition of $C_{q_0,q}$, we get

$$[z^n]F_q(z) = \sum_{k=0}^{n} \#paths(q_0,q,k) \sum_{p\in Q} \#paths(q,p,n-k).$$

Therefore, $[z^n]F_q(z)$ is the total number of times the state q is used when listing all paths of length n from q_0.

Thus, we have $\sum_{u\in pref(L,n)} scost(u) = \frac{1}{n}\sum_{q\in Q} scost(q)[z^n]F_q(z)$. Finally, let $S(z) = \sum_{p\in p} C_{q_0,p}(z)$. Then, $wcost(\mathcal{A}) = \lim_{n\to\infty} \frac{1}{n\cdot[z^n]S(z)} \sum_{q\in Q} scost(q)[z^n]F_q(z)$. In the full version we use techniques from [4] and [14] to compute the latter limit in polynomial time, by asymptotic estimations of the coefficients, thus concluding the proof. ◀

Sensing cost of languages

For a safety language L, we define $lcost(L) = \inf\{lcost(\mathcal{A}) : \mathcal{A} \text{ is a DLW for } L\}$, and similarly for $wcost(L)$. Different DLWs for a language L may have different sensing costs. We show that the minimal sensing cost in both approaches is attained at the minimal-size DLW. We first need some definitions and notations.

Consider a safety language $L \subseteq \Sigma^\omega$. 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 $z \in \Sigma^\omega$, 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 DFAs. 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* $u^{-1}L = \{w : uw \in L\}$. Note that for safety languages, there is at most one class $[u]$, namely the class of bad prefixes, such that $u^{-1}L = \emptyset$. We denote this class $[\bot]$. When $L \neq \emptyset$ is a regular safety language, the set $\langle L\rangle$ is finite, and induces the *residual automaton* of L, defined by $\mathcal{R}_L = \langle \Sigma, \langle L\rangle \setminus \{[\bot]\}, \delta_L, [\epsilon]\rangle$, with $\delta_L([u],a) = [u\cdot a]$ for all $[u] \in \langle L\rangle \setminus \{[\bot]\}$ and $a \in \Sigma$ such that $[u\cdot a] \neq [\bot]$. The automaton $\mathcal{R}_L$ is well defined and is the unique minimal-size DLW for L.

Consider a DLW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta\rangle$ such that $L(\mathcal{A}) = L$. For a state $s = [u] \in \langle L\rangle \setminus \{[\bot]\}$, we associate with s a set $states(\mathcal{A}, s) = \{q \in Q : L(\mathcal{A}^q) = u^{-1}L\}$. That is, $states(\mathcal{A}, s) \subseteq Q$ contains exactly all state that $\mathcal{A}$ can be in after reading a word that leads $\mathcal{R}_L$ to $[u]$.

The following claims are simple exercises.

▶ **Proposition 7.** *Consider a safety language L and a DLW $\mathcal{A}$ for it.*

1. *The set $\{states(\mathcal{A}, s) : s \in \langle L\rangle \setminus \{[\bot]\}\}$ forms a partition of the states of $\mathcal{A}$.*
2. *For every state $s \in \langle L\rangle \setminus \{[\bot]\}$ of $\mathcal{R}_L$, letter $\sigma \in \Sigma$, and state $q \in states(\mathcal{A}, s)$, we have $\delta(q,\sigma) \in states(\mathcal{A}, \delta_L(s,\sigma))$.*

▶ **Lemma 8.** *Consider a safety language $L \subseteq \Sigma^\omega$. For every* DLW *$\mathcal{A}$ with $L(\mathcal{A}) = L$, we have that $lcost(\mathcal{A}) \geq lcost(\mathcal{R}_L)$ and $wcost(\mathcal{A}) \geq wcost(\mathcal{R}_L)$*

Proof. We outline the key points in the proof for *lcost*. The arguments for *wcost* are similar. For a detailed proof see the full version.

Recall that the states of $\mathcal{R}_L$ are $\langle L\rangle \setminus \{[\bot]\}$. We start by showing that for every $s \in \langle L\rangle \setminus \{[\bot]\}$ and for every $q \in states(\mathcal{A}, s)$ we have that $scost(q) \geq scost(s)$. Next, we consider the Markov chains $\mathcal{M}_\mathcal{A}$ and $\mathcal{M}_{\mathcal{R}_L}$. Using Proposition 7 we show that if π and τ are the limiting distributions of $\mathcal{M}_\mathcal{A}$ and $\mathcal{M}_{\mathcal{R}_L}$ respectively, then for every $s \in \langle L\rangle \setminus \{[\bot]\}$ we have that $\tau(s) = \sum_{q\in states(\mathcal{A},s)} \pi(q)$. Finally, since Q is partitioned by $\{states(\mathcal{A}, s)\}_s$ we conclude that $lcost(\mathcal{A}) \geq lcost(\mathcal{R}_L)$. ◀

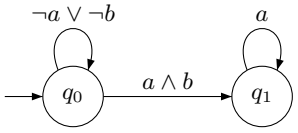

Figure 2 A DLW for $(\neg a \vee \neg b)^\omega + (\neg a \vee \neg b)^* \cdot (a \wedge b) \cdot a^\omega$.

Lemma 8 and Theorems 5 and 6 allow us to conclude with the following.

▶ **Theorem 9.** *Given a* DLW $\mathcal{A}$*, we can compute* $lcost(L(\mathcal{A}))$ *and* $wcost(L(\mathcal{A}))$ *in polynomial time.*

▶ **Example 10.** Consider the DLW $\mathcal{A}$ over the alphabet $2^{\{a,b\}}$ appearing in Figure 2.

Clearly, $\mathcal{A}$ is a minimal automaton for $L = (\neg a \vee \neg b)^\omega + (\neg a \vee \neg b)^* \cdot (a \wedge b) \cdot a^\omega$. By Lemma 8, we can calculate the sensing cost of $\mathcal{A}$ in order to find the sensing cost of L.

Clearly, $scost(q_0) = 2$ and $scost(q_1) = 1$. We start by computing $lcost(\mathcal{A})$. The corresponding Markov chain $M_\mathcal{A}$ has only one ergodic component $\{q_1\}$, so we obtain $lcost(\mathcal{A}) = scost(q_1) = 1$. The computation of $wcost(\mathcal{A})$ is more intricate. In the full version we show that $wcost(\mathcal{A}) = 2$. We remark that unlike in the other versions of sensing cost, transient components can play a role in *wcost*. In particular, If the self-loop on q_0 has been labeled by two rather than three letters, then we would have gotten $wcost(\mathcal{A}) = \frac{3}{2}$.

4 Synthesis

In the setting of synthesis, the signals in P are partitioned into sets I and O of input and output signals. An I/O-transducer $\mathcal{T}$ senses only input signals and we define its sensing cost as the sensing cost of the DLW $\mathcal{A}_\mathcal{T}$ it induces.

We define the *I/O-sensing cost* of a realizable specification $L \in (2^{I \cup O})^\omega$ as the minimal cost of an I/O-transducer that realizes L. Thus, $scost_{I/O}(\mathcal{A}) = \inf\{scost(\mathcal{T}) : \mathcal{T}$ is an I/O-transducer that realizes $L\}$. In this section we consider the problem of finding a minimally-sensing I/O-transducer that realizes L.

The realizability problem for DLW specifications can be solved in polynomial time. Indeed, given a DLW $\mathcal{A}$, we can view $\mathcal{A}$ as a game between a system, which controls the outputs, and an environment, which controls the inputs. We look for a strategy for the system that never reaches an undefined transition. This amounts to solving a turn-based safety game, which can be done in polynomial time.

When sensing is introduced, it is not enough for the system to win this game, as it now has to win while minimizing the sensing cost. Intuitively, not sensing some inputs introduces incomplete information to the game: once the system gives up sensing, it may not know the state in which the game is and knows instead only a set of states in which the game may be. In particular, unlike usual realizability, a strategy that minimizes the sensing need not use the state space of the DLW. We start with an example illustrating this.

▶ **Example 11.** Consider the DLW $\mathcal{A}$ appearing in Figure 3. The DLW is over $I = \{p, q\}$ and $O = \{a\}$. A realizing transducer over the structure of $\mathcal{A}$ (see $\mathcal{T}_1$ in Figure 4) senses p and q, responds with a if $p \wedge q$ was sensed and responds with $\neg a$ if $\neg p \wedge \neg q$ was sensed. In case other inputs are sensed, the response is arbitrary (denoted $*$ in the figure). As $\mathcal{T}_1$ demonstrates, every transducer that is based on the structure of $\mathcal{A}$ senses two input signals (both p and q) every second step, thus its sensing cost is 1. As demonstrated by the transducer $\mathcal{T}_2$ in Figure 5, it is possible to realize $\mathcal{A}$ with sensing cost of $\frac{1}{2}$ by only sensing p every second

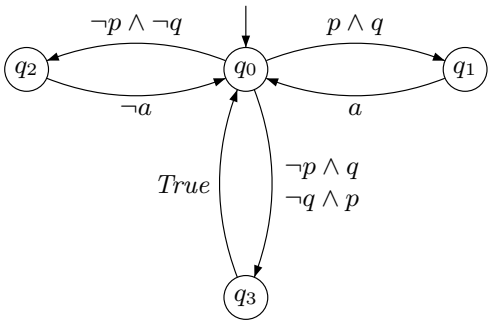

Figure 3 The DLW $\mathcal{A}$ in Example 11.

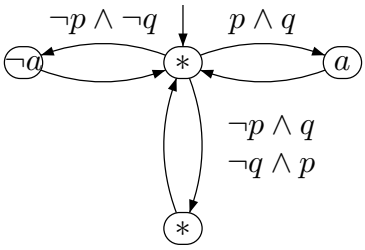

Figure 4 The transducer $\mathcal{T}_1$ for $\mathcal{A}$.

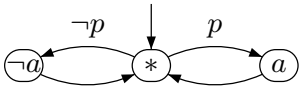

Figure 5 The transducer $\mathcal{T}_2$ for $\mathcal{A}$.

step. Indeed, knowing the value of p is enough in order to determine the output. Note that $\mathcal{T}_2$ may output sometimes a and sometimes $\neg a$ after reading assignments that causes $\mathcal{A}$ to reach q_3. Such a behavior cannot be exhibited by a transducer with the state-structure of $\mathcal{A}$.

Solving games with incomplete information is typically done by some kind of a subset-construction, which involves an exponential blow up. Unlike usual games with incomplete information, here the strategy of the system should not only take care of the realizability but also decides which input signals should be sensed, where the goal is to obtain a minimally sensing transducer. In order to address these multiple objectives, we first construct an MDP in which the possible policies are all winning for the system, and corresponds to different choices of sensing. An optimal policy in this MDP then induces a minimally-sensing transducer.

▶ **Theorem 12.** *Consider a* DLW $\mathcal{A}$ *over* $2^{I \cup O}$. *If* $\mathcal{A}$ *is realizable, then there exists an MDP* $\mathcal{M}$ *in which an optimal strategy corresponds to a minimally-sensing* I/O*-transducer that realizes* $\mathcal{A}$. *The MDP* $\mathcal{M}$ *has size exponential in* $|\mathcal{A}|$ *and can be computed in time exponential in* $|\mathcal{A}|$.

Proof. Consider a DLW $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta \rangle$. We obtain from $\mathcal{A}$ an MDP $\mathcal{M} = \langle \mathcal{S}, \text{START}, A, \text{P}, \mathit{cost} \rangle$, where $\mathcal{S} = (2^Q \times \{0, 1, \bot\}^I) \cup \{\text{START}\}$, and $A = 2^I \times 2^O$. Intuitively, when $\mathcal{M}$ is in state $\langle S, \ell \rangle$, for $S \subseteq Q$ and $\ell : I \to \{0, 1, \bot\}$, then $\mathcal{A}$ can be in every state in S, and for each input signal $b \in I$, we have that either b is true ($\ell(b) = 1$), b is false ($\ell(b) = 0$), or b is not sensed ($\ell(b) = \bot$). The action (o, i) means that we now output o and in the next state we will sense only inputs in i. For $\star \in \{\bot, 0, 1\}$, we define $\ell_\star = \{b \in I : \ell(b) = \star\}$.

We define the actions so that an action $\langle o, i \rangle$ is available in state $\langle S, \ell \rangle$ if for every $q \in S$ and $i' \subseteq \ell_\bot$, we have that $\delta(q, \ell_1 \cup i' \cup o)$ is defined. That is, an action is available if its o component does not cause $\mathcal{A}$ to get stuck no matter what the assignment to the signals that are not sensed is.

The transition probabilities are defined as follows. Consider a state $\langle S, \ell \rangle$, and an available action $\langle o, i \rangle$. Let $S' = \bigcup_{q \in S} \bigcup_{i' \subseteq \ell_\bot} \{\delta(q, \ell_1 \cup i' \cup o)\}$. Recall that by taking action $\langle o, i \rangle$, we decide that in the next state we will only sense signals in i. For $i \subseteq I$, we say that an assignment $\ell' : I \to \{0, 1, \bot\}$ *senses* i if $\ell'_1 \cup \ell'_0 = i$. Note that there are $2^{|i|}$ assignments that sense i. Accordingly, we have $\text{P}(\langle S, \ell \rangle, \langle o, i \rangle, \langle S', \ell' \rangle) = 2^{-|i|}$ for every $\ell' : I \to \{0, 1, \bot\}$ that senses i. That is, a transition from $\langle S, \ell \rangle$ with $\langle o, i \rangle$ goes to the set of all possible successors of S under inputs that are consistent with ℓ and the output assignment o, and the ℓ' component

is selected with uniform distribution among all assignments that sense i. The cost function depends on the number of signals we sense, thus $cost(\langle S, \ell\rangle) = |\ell_1 \cup \ell_0|$.

Finally, in the state START we only choose an initial set of input signals to sense. Thus, for every ℓ such that $\ell_1 \cup \ell_0$, we have $\mathrm{P}(\text{START}, \langle o, i\rangle, \langle\{q_0\}, \ell\rangle) = 2^{-|i|}$. Note that START is not reachable from any state in $\mathcal{M}$, and thus its cost is irrelevant. We arbitrarily set $cost(\text{START}) = 0$.

In the full version we prove that $cost(\mathcal{M}) = scost_{I,O}(\mathcal{A})$ and that a minimal-cost policy f in $\mathcal{M}$ induces a minimally-sensing I/O-transducer that realizes $\mathcal{A}$. Intuitively, we prove this by showing a correspondence between transducers and policies, such that the sensing cost of a transducer $\mathcal{T}$ equals the value of the policy it corresponds to in $\mathcal{M}$.

Finally, we observe that the size of $\mathcal{M}$ is single exponential in the size of $\mathcal{A}$, and that we can construct $\mathcal{M}$ in time exponential in the size of $\mathcal{A}$. ◀

▶ **Theorem 13.** *A minimally-sensing transducer for a realizable* DLW $\mathcal{A}$ *has size tightly exponential in* $|\mathcal{A}|$.

Proof. The upper bound follows from Theorem 3 applied to the MDP constructed in Theorem 12.

For the lower bound, we describe a family of realizable DLWs $\mathcal{A}_1, \mathcal{A}_2, \ldots$ such that for all $k \geq 1$, the DLW $\mathcal{A}_k$ has $1 + \sum_{i=1}^{k} p_i$ states, yet a minimally-sensing transducer for it requires at least $\prod_{i=1}^{k} p_i$ states, where $p_1, p_2, \ldots$ are prime numbers. Intuitively, $\mathcal{A}_k$ is constructed as follows. In the initial state q_{reset}, the inputs signals determine a number $1 \leq i \leq k$, and $\mathcal{A}_k$ moves to component i, which consists of a cycle of length p_i. In every state j in component i, the output signals must acknowledge that $\mathcal{A}_k$ is in state $0 \leq j < p_i$ of component i. Furthermore, we force a sensing of 1 in every state except for q_{reset} by requiring a signal to be acknowledged in every step. Finally, we can go back to q_{reset} only with a special output signal, which can be outputted only in state 0 of an i component.

Thus, a realizing transducer essentially only chooses which signals to read in q_{reset}. We show that 0 bits can be read, but in that case we need $\prod_{i=1}^{k} p_i$ states. Indeed, the transducer needs to keep track of the location in all the i components simultaneously, which means keeping track of the modulo from each p_i. Since every combination of such modulos is possible, the transducer needs $\prod_{i=1}^{k} p_i$ states. In the full version we formalize this intuition. ◀

We now turn to study the complexity of the problem of finding a minimally-sensing transducer. By the construction in Theorem 12 and the polynomial time algorithm from Theorem 3, we have the following.

▶ **Theorem 14.** *Consider a realizable* DLW $\mathcal{A}$ *over* $2^{I \cup O}$. *We can calculate* $cost_{I,O}(\mathcal{A})$ *and return a minimally-sensing* I/O*-transducer that realizes* $\mathcal{A}$ *in time exponential in* $|\mathcal{A}|$.

In order to complete the picture, we consider the corresponding decision problem. Given a DLW $\mathcal{A}$ over $2^{I \cup O}$ and a threshold γ, the sensing problem in the open setting is to decide whether $cost_{I,O}(\mathcal{A}) < \gamma$.

▶ **Theorem 15.** *The sensing problem in the open setting is EXPTIME-complete.*

Proof. The upper bound follows from Theorem 14. For the lower bound, we show that the problem is EXPTIME hard even for a fixed γ. Given a DLW specification $\mathcal{A}$ over $2^{I \cup O}$, we show that it is EXPTIME-hard to decide whether there exists a transducer $\mathcal{T}$ that realizes $\mathcal{A}$ with $scost(\mathcal{T}) < 1$. We show a reduction from the problem of deciding the nonemptiness of an intersection of finite deterministic tree automata proved to be EXPTIME-hard in [6].

The idea is similar to that of Theorem 13, where a reset state is used to select an object, and a transducer can ignore the inputs in this state by using a response which is acceptable in every possible selected object.

A deterministic automaton on finite trees (DFT) is $\mathcal{U} = \langle \Sigma, Q, \delta, q_0, F\rangle$, where Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to Q \times Q$ is a transition function, and $F \subseteq Q$ is a set of accepting states. We refer to the left and right components of δ as $\delta_\triangleleft$ and $\delta_\triangleright$. For example, when $\delta(q,\sigma) = \langle q_l, q_r\rangle$, we write $\delta_\triangleleft(q,\sigma) = q_l$. An DFT runs on Σ-*trees*. A (binary) Σ-tree is $T = \langle \tau, \ell\rangle$ where $\tau \subseteq \{\triangleleft,\triangleright\}^*$ is prefix-closed: for every $x \cdot \sigma \in \tau$ it holds that $x \in \tau$, and $\ell : \tau \to \Sigma$ is a labeling function. For simplicity, we require that for every $x \in \tau$, either $\{x\triangleleft, x\triangleright\} \subseteq \tau$, or $\{x\triangleleft, x\triangleright\} \cap \tau =$, in which case x is a leaf. Given a tree $T = \langle \tau, \ell\rangle$, the *run* of $\mathcal{U}$ on T is a Q-tree $\langle \tau, \ell'\rangle$ where $\ell'(\epsilon) = q_0$, and for every $x \in \tau$ such that x is not a leaf, we have $\delta(\ell'(x), \ell(x)) = \langle \ell'(x\triangleleft), \ell'(x\triangleright)\rangle$. A run is *accepting* if every leaf is labeled by an accepting state. A Σ-tree T is accepted by $\mathcal{U}$ if the run of $\mathcal{U}$ on T is accepting.

The nonempty-intersection problem gets as input DFTs $\mathcal{U}_1, \ldots, \mathcal{U}_n$, and decides whether their intersection is nonempty, that is $\bigcap_{t=1}^n L(\mathcal{U}_t) \neq \emptyset$. Given $\mathcal{U}_1, \ldots, \mathcal{U}_n$, we construct a specification DLW $\mathcal{A}$ such that $\bigcap_{t=1}^n L(\mathcal{U}_t) \neq \emptyset$ iff $scost(\mathcal{A}) < 1$. We assume w.l.o.g. that $L(\mathcal{U}_t) \neq \emptyset$ for all $1 \leq t \leq n$.

We construct $\mathcal{A}$ as follows. Initially, the inputs specify an index $1 \leq t \leq n$. Then, the transducer should respond with a tree in $L(\mathcal{U}_t)$. This is done by challenging the transducer with a branch in the tree, until some reset input signal is true, and the process repeats. Now, if $\bigcap_{t=1}^n L(\mathcal{U}_t) \neq \emptyset$, the transducer can ignore the input signals that specify the index t and just repeatedly output a tree in the intersection. On the other hand, if $\bigcap_{t=1}^n L(\mathcal{U}_t) = \emptyset$, the transducer must sense some information about the specified index.[3]

We now formalize this intuition. For $1 \leq t \leq n$, let $\mathcal{U}_t = \langle 2^J, Q^t, \delta^t, q_0^t, F^t\rangle$. Note that we assume w.l.o.g that the alphabet of all the DFTs is 2^J. We construct a specification DLW $\mathcal{A} = \langle 2^{I \cup O}, Q, q_0, \delta\rangle$ as follows. The set of states of $\mathcal{A}$ is $Q = \bigcup_{t=1}^n Q^t \cup \{\textsc{reset}\}$. Assume w.l.o.g that $n = 2^k$ for some $k \in \mathbb{N}$. We define $I = \{b_1, \ldots, b_k\} \cup \{dI\}$ and $O = J \cup \{dO, e\}$. The input signal dI and the output signal dO denote the direction of branching in the tree. For clarity, in an input letter $i \in I$ we write $i(dI) = \triangleleft$ (and $i(dI) = \triangleright$) to indicate that $dI \notin i$ (and $dI \in i$). We use a similar notation for dO.

We define the transition function as follows. In state $\textsc{reset}$, we view the inputs $b_1, \ldots, b_k$ as a binary encoding of a number $t \in \{1, \ldots, n\}$. Then, $\delta(\textsc{reset}, t) = q_0^t$. Next, consider a state $q \in Q^t$, and consider letters $i \subseteq I$ and $o \subseteq O$. We define δ as follows:

$$\delta(q, i \cup o) = \begin{cases} \textsc{reset} & q \in F \wedge e \in o \wedge o(dO) = i(dI) \\ \delta^t_\triangleleft(q, o \cap J) & e \notin o \wedge o(dO) = i(dI) = \triangleleft \\ \delta^t_\triangleright(q, o \cap J) & e \notin o \wedge o(dO) = i(dI) = \triangleright \end{cases}$$

Note that $\delta(q, i \cup o)$ is undefined when $o(dO) \neq i(dI)$ or when $q \notin F$ and $e \in o$. Intuitively, in state $\textsc{reset}$, an index $1 \leq t \leq n$ is chosen. From then on, in a state $q \in Q^t$, we simulate the run of $\mathcal{U}_t$ on the left or right branch of the tree, depending on the signal dI. The next letter is outputted in o, and additionally, we require that dO matches dI.

We claim that $scost(\mathcal{A}) < 1$ iff $\bigcap_{t=1}^n L(\mathcal{U}_t) \neq \emptyset$. In the first direction, assume that $\bigcap_{t=1}^n L(\mathcal{U}_t) \neq \emptyset$, and let T be a tree such that $T \in \bigcap_{t=1}^n L(\mathcal{U}_t) \neq \emptyset$. Consider the following

[3] Note that since a tree in the intersection of DFTs may be exponentially bigger than the DFTs, the lower bound here also suggests an alternative lower bound to the exponential size of a minimally-sensed transducer, now with a polynomial set of signals (as opposed to the proof of Theorem 13).

transducer $\mathcal{T}$: in the state RESET it does not sense any inputs, and then it outputs a branch of T according to the signal dI, while always acknowledging the dI bit with the correct dO. When the end of the branch is reached, it outputs e. Since T is accepted by every DFT U^t, it follows that $\mathcal{T}$ realizes $\mathcal{A}$. Moreover, let l be the longest branch in T, then every l steps at most, $\mathcal{T}$ visits a state corresponding to RESET, in which it senses nothing. Thus, $\mathcal{T}$ senses 1 for at most l steps, and then 0. It follows that $scost(\mathcal{T}) \leq \frac{l}{l+1} = 1 - \frac{1}{l+1} < 1$.

Conversely, observe that in every state $q \in Q \setminus \{\text{RESET}\}$, a realizing transducer must sense at least 1 signal, namely dI. Thus, the only way to get sensing cost of less than 1 is to visit RESET infinitely often (in fact, with bounded sparsity), and to sense 0 in RESET. However, sensing 0 in RESET means that the next state could be the initial state of any of the n DFTs. Moreover, visiting RESET again means that at some point e was outputted in an accepting state of one of the DFTs. Thus, the transducer outputs a tree that is accepted in every DFT, so $\bigcap_{t=1}^{n} L(\mathcal{U}_t) \neq \emptyset$.

Finally, observe that the reduction is clearly polynomial, and thus we conclude that deciding whether $scost(\mathcal{A}) < 1$ is EXPTIME-hard. ◀

5 Discussion and Future Research

Sensing is a basic measure of the complexity of monitoring and synthesis. In monitoring safety properties, the definition of sensing presented in [2] is not informative, as it gives sensing cost 0 to properties that are satisfied with probability 0. We argue that in the context of monitoring, the definition of sensing cost should consider only computations that satisfy the property, and we study the complexity of computing the sensing cost of a property in the new definition. We distinguish between two approaches to define a probabilistic measure with respect to the set of computations that satisfy a property. We show that while computing the sensing cost according to the new definitions is technically more complicated than in [2], the minimal sensing is still attained by a minimal-size automaton, and it can still be computed in polynomial time.

In synthesis, we introduce a new degree of freedom, namely choosing the outputs when realizing a specification. We study the complexity of finding a minimal-sensing transducer for safety specifications. We show that the minimal-sensing transducer is not necessarily minimal in size. Moreover, interestingly, unlike the case of traditional synthesis, a minimal-sensing transducer need not even correspond to a strategy embodied in the specification deterministic automaton. On the positive side, we show that a minimal-sensing transducer always exists (for a realizable safety specification) and that its size is at most exponential in the size of the minimal-size transducer. We also provide matching lower bounds.

We now turn to discuss some future directions for research.

Non-safety properties. We focus on safety properties. The study in [2] completes the monitoring picture for all other ω-regular properties. We plan to continue the study of synthesis of ω-regular properties. An immediate complication in this setting is that a finite minimal-sensing transducer does not always exists. Indeed, even in the monitoring setting studied in [2], a minimal-sensing automaton does not always exist. Even the decidability of computing the optimal cost remains open.

A trade-off between sensing and quality. Reducing the sensing cost of a transducer can often be achieved by *delaying* the sensing of some letter, thus sensing it less often. This, however, means that eventualities may take longer to be fulfilled, resulting in transducers

of lower quality [1]. We plan to formalize and study the trade-off between the sensing and quality and relate it to the trade-offs between size and sensing, as well as between size and quality.

Acknowledgment. We thank Elie de Panafieu for helpful discussions.

References

1 S. Almagor, U. Boker, and O. Kupferman. Discounting in LTL. In *20th TACAS*, 2014.
2 S. Almagor, D. Kuperberg, and O. Kupferman. Regular sensing. In *34th FSTTCS*, pages 161–173, 2014.
3 D.L. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52:1289–1306, 2006.
4 P. Flajolet and R. Sedgewick. Analytic combinatorics: functional equations, rational and algebraic functions. 2001.
5 C. Grinstead and J. Laurie Snell. 11:Markov chains. In *Introduction to Probability*. American Mathematical Society, 1997.
6 J. Goubault. Rigid E-Unifiability is DEXPTIME-Complete. In *9th LICS*, pages 498–506, 1994.
7 K. Havelund and G. Rosu. Efficient monitoring of safety properties. *Software Tools for Technology Transfer*, 6(2):18–173, 2004.
8 G. Kindler. *Property Testing, PCP, and Juntas*. PhD thesis, Tel Aviv University, 2002.
9 O. Kupferman and M.Y. Vardi. Synthesis with incomplete information. In *Advances in Temporal Logic*, pages 109–127. Kluwer Academic Publishers, 2000.
10 E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.
11 C. Mauduit and A. Sárköz. On finite pseudorandom binary sequences. i. measure of pseudorandomness, the legendre symbol. *Acta Arith.*, 82(4):365–377, 1997.
12 S. Muthukrishnan. Theory of data stream computing: where to go. In *Proc. 30th PODS*, pages 317–319, 2011.
13 A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th POPL*, pages 179–190, 1989.
14 M.F. Roy and A. Szpirglas. Complexity of the computation on real algebraic numbers. *J. Symb. Comput.*, 10(1):39–52, 1990.
15 A.P. Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6:495–511, 1994.