Classification Under Uncertainty

Orna Kupferman
School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel

Ofer Leshkowitz
School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel

Abstract

Consider a fixed number of disjoint regular languages $L_1, \ldots, L_k \subseteq \Sigma^*$. A *classifier* for $L_1, \ldots, L_k$ is a transducer that receives each moment t in time an input letter $\sigma_t \in \Sigma$ and outputs an index in $\{1, \ldots, k\}$ such that if the word $\sigma_1 \cdots \sigma_t$, generated so far, is in some (unique) language L_i, then this index is i. Classifiers arise naturally in runtime monitoring and online stream processing, where a system must continuously determine which of several specifications or behaviors is currently being realized. The problem of generating classifiers of minimal size has been well studied.

In many applications, the input alphabet is of the form 2^P, for a finite set P of signals. There, the complexity of classification stems not only from the languages but also from the presence of uncertainty, namely when the valuation to some signals may not be known. We introduce and study *classification under uncertainty*, where the input words may be partially observed. We consider three sources for uncertainty: (1) *Given*: the input to the problem specifies which signals may be sensed after each behavior. (2) *Privacy*: the input includes a list of secret behaviors, and the classifier should restrict sensing so that secrets are not revealed. (3) *Budget*: Sensing of signals incurs a cost, which the classifier should minimize.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Formal methods, Automata, Incomplete Information

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2026.40

Funding Research supported by the Israel Science Foundation, Grant 3177/25, and European Research Council, Advanced Grant ADVANSYNT.

1 Introduction

Already in the 1970s, researchers have studied the problem of DFA*identification* from data: given a set D of words, labeled positively and negatively, we seek a deterministic automaton over finite words (DFA) $\mathcal{A}$ that agrees with D. That is, $\mathcal{A}$ accepts all the positively-labeled words, rejects all the negatively-labeled words, and may accept or reject all other words. A fundamental result by Gold shows that the freedom with respect to unlabeled words makes the problem of finding a minimal identifying DFA NP-complete [12]. The identification problem has numerous heuristics and applications [27, 15]. The massive use of data of rich types motivates two natural generalization of the problem. First, rather than a finite set D of labeled words, one may consider two DFAs that describe the (possibly infinite) sets of positive and negative words. This is called *separation*: Given two DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ with disjoint languages, we seek a DFA $\mathcal{A}$ such that $L(\mathcal{A}_1) \subseteq L(\mathcal{A})$ and $L(\mathcal{A}) \cap L(\mathcal{A}_2) = \emptyset$. NP-hardness of DFA separation can be easily obtained by a reduction from DFA identification, and researchers have studied variants like strict separation [11, 17], regular separators of general languages [5], as well as separation of regular languages by weaker classes of languages, e.g., FO-definable languages [23] or piecewise testable languages [6].

A second generalization allows the words in D to be labeled by more than two labels. Then, rather than a DFA that agrees with D, we seek a *transducer* that agrees with D. Essentially, a transducer is a DFA whose states are labeled by output letters, refining the accept/reject label in DFAs. Combining the two extensions, one gets the *classification*

problem: For $k \geq 1$ and k DFAs $\mathcal{A}_1, \ldots, \mathcal{A}_k$ with disjoint languages over some alphabet Σ, we seek a transducer that, at each moment t in time, reads an input letter $\sigma_t \in \Sigma$ and outputs an index in $\{1, \ldots, k\}$ such that whenever the word $\sigma_1 \cdots \sigma_t$ generated so far belongs to $L(\mathcal{A}_i)$, the output index is i. Again, it is easy to see that as long as k is constant, results on identification apply also to the classification problem. Indeed, a classifying transducer can be defined on top of the product of the underlying DFAs, a candidate minimal transducer can be checked in polynomial time, implying membership in NP, and NP-hardness follows from the NP-hardness of the separation problem, namely the case $k = 2$. When k is not constant, the product of the underlying DFAs is exponential in k, the problem is PSPACE-complete, and the analysis is less interesting, as the exponent in k dominates other factors. In order to cancel the dominance of k, we assume that the input to the classification problem is a transducer $\mathcal{T}$ that maps each word to an index in $\{0, 1, \ldots, k\}$, inducing k disjoint languages – for $i \in \{1, \ldots, k\}$, the language L_i consists of words mapped to index i, as well as *don't care* words – these that $\mathcal{T}$ map to 0.

Classifiers are used in runtime monitoring and control of reactive systems, where decisions must be taken online based on a growing execution prefix. Typical applications include fault detection, protocol identification, and intrusion detection, where the system must continuously determine which of several behaviors or specifications is currently being realized. This view is closely related to runtime verification [21, 13], where executions are monitored against formal specifications, and to stream or event processing, where patterns over data streams must be recognized incrementally [4]. Classifiers can be seen as a refinement of monitors that not only detect violations but also distinguish between multiple hypotheses or modes of operation.

In many of the above applications, the input alphabet is of the form 2^P, for a finite set P of signals. This corresponds to the standard representation of computations in formal methods, where behaviors correspond to a sequence of valuations [24, 28]. In these settings, the challenge of classifying stems not only from the languages but also from the need to make decisions in the presence of uncertainty, namely when the valuation to some signals may not be known.

▶ **Example 1.** Consider a setting with $P = \{p\}$, thus behaviors correspond to languages over the alphabet of two valuations, which we denote by $\{p, \bar{p}\}$. Then, `True` denotes an unknown valuation. Consider the transducer $\mathcal{T}$ on the left of Figure 1.

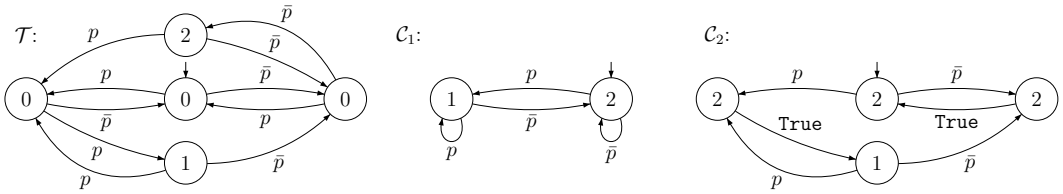

Figure 1 The specification transducer $\mathcal{T}$, and two classifiers that satisfy $\mathcal{T}$.

The transducer $\mathcal{T}$ specifies two disjoint languages $L_1, L_2 \subseteq \{p, \bar{p}\}^*$. It does so by leading each word in $\{p, \bar{p}\}^*$ to a state labeled 0 (the word is not in $L_1 \cup L_2$), 1 (the word is in L_1), or 2 (the word is in L_2). It is not hard to see (at least to readers who do not miss the arrow pointing to the initial state...) that $L_1 = (\texttt{True} \cdot \texttt{True})^* \cdot p \cdot p$ and $L_2 = (\texttt{True} \cdot \texttt{True})^* \cdot \bar{p} \cdot \bar{p}$.

The transducer $\mathcal{C}_1$ is a classifier: it maps each word in $\{p, \bar{p}\}^*$ to either 1 or 2. Specifically, it maps to 1 exactly all words in $\texttt{True}^* \cdot p$ and maps to 2 all words in $\texttt{True}^* \cdot \bar{p}$. The classifier $\mathcal{C}_1$ satisfies $\mathcal{T}$. Indeed, all words in L_1 are mapped to 1 and all words in L_2 are mapped to 2. It is not hard to prove that $\mathcal{C}_1$ is a minimal classifier for $\mathcal{T}$. The classifier $\mathcal{C}_1$, however, needs to know the valuation to the signal p in all its transitions.

The transducer $\mathcal{C}_2$ is also a classifier for $\mathcal{T}$. Indeed, $\mathcal{C}_2$ maps to 1 exactly all words in $(\texttt{True} \cdot \texttt{True})^* \cdot p \cdot \texttt{True}$, and maps to 2 exactly all words in $(\texttt{True} \cdot \texttt{True})^* \cdot \bar{p} \cdot \texttt{True}$. Thus, all words in L_1 are mapped to 1 and all words in L_2 are mapped to 2. Essentially, the transducer $\mathcal{C}_2$ uses the fact that the specification transducer $\mathcal{T}$ does not care for the classification of words whose last two letters are different, and uses this fact in order to give up the sensing of p in all even positions. ⌟

We introduce and study *classification under uncertainty*, where the input words are only partially observed. That is, rather than reading at each moment a valuation to all the signals in P, the classifier may sense a valuation to a subset of them – possibly a different subset at each moment – and its transitions depend only on the sensed signals. Our contribution is related to work on monitoring and runtime verification under partial observability, for example by allowing missing events, indistinguishability relations, or assumptions on the hidden part of the trace [20, 3, 9]. This line of work primarily focuses on extending the monitoring pipeline so as to produce sound verdicts despite incomplete observations. For example, [3] proposes to restrict the space of executions by adding assumptions on the missing information, while [9] reasons about sets of traces consistent with the observed one, maintaining a belief set over the hidden behaviors. In contrast, we study the theoretic foundation of the problem: we explicitly model the sources of uncertainty and study their effect on the classification problem, yielding a structural and complexity-theoretic understanding of classification under uncertainty.

We consider three sources for uncertainty. The first is when uncertainty is determined by the environment and is known in advance. More formally, uncertainty is specified by a *sensing constraint* – a transducer $\mathcal{U}$ that maps each word $w \in (2^P)^*$ to the set of signals whose valuation is observed (in particular, by $\mathcal{U}$) after w is read. The goal is to generate, given a language specification $\mathcal{T}$ and a sensing constraint $\mathcal{U}$, a classifier that satisfies $\mathcal{T}$ and respects $\mathcal{U}$. We show that, surprisingly, while the problem of deciding whether such a classifier exists is NLOGSPACE-complete, a minimal classifier that satisfies $\mathcal{U}$ may be exponential in $\mathcal{T}$. Intuitively, while a negative answer to the decision problem has a short witness, induced by following the runs of $\mathcal{T}$ on two partially-sensed short words, a witness for a positive answer involves all words, and thus has to remember arbitrary subsets of states of $\mathcal{T}$.

The two other sources of uncertainty concern settings in which it is of the classifier's interest to generate uncertainty, either for ensuring *privacy* or for minimizing *sensing cost*.

Privacy is an important quality measure for reactive systems: we seek systems that allow the underlying components not to reveal information they prefer to keep private. For reactive systems, researchers have suggested to reason about privacy either by hyper-properties [10], or, more directly, by letting the users specify *secrets* whose truth value should be kept unknown [18, 19]. For the problem of classification, we would like the classifier to observe the input word in a way that enables correct classification without revealing secret information.

Consider for example the language specification $\mathcal{T}$ from Example 1. Suppose we want to classify the words according to $\mathcal{T}$, yet we want to keep the value of p in all positions that are 1 mod 3 unknown. Clearly, the secret is not hidden from Classifier $\mathcal{C}_1$, as it fully observes all the valuations to p. The secret is not hidden from Classifier $\mathcal{C}_2$ either, as $\mathcal{C}_2$ observes the first letter. In Example 9, we describe a classifier $\mathcal{C}_3$ for $\mathcal{T}$ that does hide the secret.

Formally, the problem of *privacy-preserving classification* gets as input, in addition to the language specification $\mathcal{T}$, also a *secret* $\mathcal{H}$, which is also a language-specification transducer; thus $\mathcal{H}$ maps each word in $(2^P)^*$ to an index in $\{0, \ldots, m\}$, for some $m \geq 2$. The goal is to generate a classifier that satisfies $\mathcal{T}$ and respects a sensing restriction $\mathcal{U}$ that hides $\mathcal{H}$: for

every word $w \in (2^P)^*$, an observer restricted by $\mathcal{U}$ cannot classify w according to $\mathcal{H}$. We show that a privacy-preserving classification can be solved in EXPTIME, a classifier may be exponential in $\mathcal{T}$, and the problem is PSPACE-hard, already for secrets of size 2. Our results can be extended to multiple secrets and richer types of secrets.

In many applications, sensing the evaluation of signals involves the operation of sensors and thus has a cost. In the third type of uncertainty, we assume that whenever a classifier senses a signal, this operation has a cost, and we seek a classifier that minimizes the expected sensing cost. Expectation is measured with respect to words of increasing lengths under a given distribution of the valuations. Consider for example the classifiers $\mathcal{C}_1$ and $\mathcal{C}_2$ in Example 1, and assume that each sensing of p incurs a cost of 1, and that p is assigned `True` and `False` with probability $\frac{1}{2}$ each. Since $\mathcal{C}_1$ senses p in all states, its expected sensing cost is 1. Since $\mathcal{C}_2$ senses p in exactly all even positions, its expected sensing cost is only $\frac{1}{2}$.

As Example 1 demonstrates, minimizing the size of classifiers may conflict with minimizing their sensing cost. We show that the tradeoff is unbounded, in fact already for $P = \{p\}$ and $k = 2$. That is, we show that for every $n \geq 1$, there is a transducer $\mathcal{T}_n$ of size $O(n)$ such that the smallest classifier that satisfies $\mathcal{T}_n$ has 2 states, it needs to constantly sense p, and there is also a classifier of size $O(n)$ that satisfies $\mathcal{T}_n$, with sensing cost $\frac{1}{n}$ only. More interestingly, we show that, in general, optimal sensing may be obtained only in a classifier that is exponentially larger than the language specification $\mathcal{T}$. This complexity is carried over to the problem of finding a minimally-sensing transducer, which we solve in EXPTIME and show to be PSPACE-hard.

From a technical point of view, all three sources of uncertainty are handled via a game-theoretic approach to the corresponding problems. Given a language specification, we define a two-player safety game in which the actions of Classifier correspond to choosing a set of signals to be sensed and the actions of Environment correspond to choosing a valuation to the signals. The goal of Classifier is to stay in a safe region in which it can assign labels to all words generated during the interaction despite its partial sensing. Sensing constraints limit the actions of Classifier, privacy limits the safe region, and sensing cost makes the game a stochastic mean-payoff game. In Section 7, we discuss how the different variants influence the complexity of the corresponding problems, and explain the challenges that make some of them still open.

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

2 Preliminaries

2.1 Automata and Transducers

A *nondeterministic automaton on finite words* (NFA, for short) is $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where Q is a finite set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, and $\alpha \subseteq Q$ is a set of accepting states. A run of $\mathcal{A}$ on a word $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_n \in \Sigma^*$ is a sequence of states $q_0, q_1, \ldots, q_n$ such that $q_{i+1} \in \delta(q_i, \sigma_{i+1})$ for all $i \geq 0$. A run is accepting if $q_n \in \alpha$. A word $w \in \Sigma^*$ is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on w. 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 to δ as a function $\delta : 2^Q \times \Sigma^* \to 2^Q$, thus extend it to set of states and to words: For every set $S \subseteq Q$, we have that $\delta(S, \epsilon) = S$; for every letter $\sigma \in \Sigma$, we have that $\delta(S, \sigma) = \cup_{q \in S} \{\delta(q, \sigma)\}$; and for every word $w \in \Sigma^*$, we have that $\delta(S, w \cdot \sigma) = \delta(\delta(S, w), \sigma)$. When δ is such that $|\delta(q, \sigma)| = 1$ for all $q \in Q$ and $\sigma \in \Sigma$, then $\mathcal{A}$ is *deterministic* (DFA, for short). We then assume that δ is of the form $\delta : Q \times \Sigma \to Q$, thus its extension to words is of the form $\delta : Q \times \Sigma^* \to Q$.

Consider two alphabets Σ_I and Σ_O, referred to as the input and output alphabet, respectively. A (Σ_I/Σ_O)-transducer $\mathcal{T}$ is similar to a deterministic automaton over the alphabet Σ_I, except that instead of an acceptance condition, each state is labelled by a letter in Σ_O. Formally, $\mathcal{T} = \langle \Sigma_I, \Sigma_O, Q, q_0, \delta, \tau \rangle$ has $\delta : Q \times \Sigma_I \to Q$, and $\tau : Q \to \Sigma_O$ maps each state to a letter in Σ_O. We extend τ to sets of states, where for each $S \subseteq Q$, we have that $\tau(S) = \bigcup_{q \in S} \{\tau(q)\}$. As with automata, we talk about the run of $\mathcal{T}$ on words in Σ_I^*, and extend δ to words. For a word $w \in \Sigma_I^*$, we use $\mathcal{T}(w)$ to denote $\tau(\delta(q_0, w))$, namely the label of the state that $\mathcal{T}$ reaches when it reads w.

2.2 Classification

Consider an alphabet Σ. A *specification* is a $(\Sigma/\{0, 1, \ldots, k\})$-transducer $\mathcal{T}$, for some $k \geq 2$, which we refer to as the *width* of $\mathcal{T}$. Each specification defines k disjoint languages $L_1, \ldots, L_k \subseteq \Sigma^*$, where for all $i \in \{1, \ldots, k\}$, we have $L_i = \{w : \mathcal{T}(w) = i\}$. The words that are mapped by $\mathcal{T}$ to 0 may be viewed as "don't care" words. We say $\mathcal{T}$ *separates* two words $w, w' \in \Sigma^*$, denoted $w \not\approx_{\mathcal{T}} w'$, if $\mathcal{T}$ cares about both w and w' and map them to different languages. That is, $0 \neq \mathcal{T}(w) \neq \mathcal{T}(w') \neq 0$.

Describing k disjoint languages by a $(\Sigma/\{0, 1, \ldots, k\})$-transducer rather than by k DFAs leads to a tighter complexity analysis. We elaborate on this choice below. Given DFAs $\mathcal{A}_i = \langle \Sigma, Q_i, q_0^i, \delta_i, \alpha_i \rangle$ for the underlying languages L_i, a k-language specification $\mathcal{T}$ that induces $L_1, \ldots, L_k$ can be defined on top of the product, thus $\mathcal{T} = \langle \Sigma, Q_1 \times \cdots \times Q_k, \langle q_0^1, \ldots, q_0^k \rangle, \delta, \tau \rangle$, where for all $\langle q_1, \ldots, q_k \rangle \in Q_1 \times \cdots \times Q_k$ and $\sigma \in \Sigma$, we have that $\delta(\langle q_1, \ldots, q_k \rangle, \sigma) = \langle \delta_1(q_1, \sigma), \ldots, \delta_k(q_k, \sigma) \rangle$. The fact the languages are disjoint guarantees that for every state $\langle q_1, \ldots, q_k \rangle \in Q_1 \times \cdots \times Q_k$, there is at most one $i \in \{1, \ldots, k\}$ such that $q_i \in \alpha_i$. If such $i \in \{1, \ldots, k\}$ exists, then $\tau(\langle q_1, \ldots, q_k \rangle) = i$, and otherwise $\tau(\langle q_1, \ldots, q_k \rangle) = 0$. Since a classifier with full visibility can be defined on top of $\mathcal{T}$, we find an analysis that refers to $\mathcal{T}$ tighter in the study of uncertainty. In particular, when k is a constant, $\mathcal{T}$ is polynomial in the underlying $\mathcal{A}_i$'s. Beyond the complexity-theoretic motivation, the choice of modeling the specification by a transducer is natural from a practical perspective. In both the specification and the desired classification, we partition the execution space into distinct scenarios or modes. Transducers form the most basic and natural formalism for such a partition.

A *classifier* is a $(\Sigma/\{1, \ldots, k\})$-transducer $\mathcal{C}$, for some *width* $k \geq 2$. Note that while specifications have don't care words, the classifier $\mathcal{C}$ induces a partition of Σ^* to k disjoint languages $C_1, \ldots, C_k \subseteq \Sigma^*$, where for all $i \in \{1, \ldots, k\}$, we have $C_i = \{w : \mathcal{C}(w) = i\}$.

We say that a classifier $\mathcal{C}$ *satisfies* a specification $\mathcal{T}$ (or that $\mathcal{C}$ is a classifier for $\mathcal{T}$, for short) if $\mathcal{C}$ and $\mathcal{T}$ have the same width, and for every word $w \in \Sigma^*$ and $i \in \{1, \ldots, k\}$, if $\mathcal{T}(w) = i$, then $\mathcal{C}(w) = i$. That is, words that $\mathcal{T}$ don't care about may be mapped arbitrarily by $\mathcal{C}$, and other words should keep their mapping by $\mathcal{T}$.

It is not hard to see that a violation of classification can be witnessed by a short word, leading to easy model checking for classifiers (see proof in the full version).

▶ **Theorem 2.** *Given a specification $\mathcal{T}$ and a classifier $\mathcal{C}$, the problem of checking whether $\mathcal{C}$ satisfies $\mathcal{T}$ is NLOGSPACE-complete.*

2.3 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 $P = \{a, b\}$, then the expression $(\texttt{True})^* \cdot a \cdot b \cdot (\texttt{True})^*$ describes all words over 2^P that contain a subword

in $\{\{a\},\{a,b\}\} \cdot \{\{b\},\{a,b\}\}$. Let Ψ_P denote the set of predicates over P. Describing transducers over input alphabet 2^P, we use predicates in Ψ_P in order to describe the transitions symbolically. Thus, transitions are described by a partial function $\delta : Q \times \Psi_P \to Q$, where for every state $q \in Q$ and letter $\sigma \in 2^P$, there is exactly one predicate $\theta \in \Psi_P$ such that σ satisfies θ and $\delta(q,\theta)$ is defined. The run of $\mathcal{T}$ on a word $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_m \in \Sigma^*$ is then the sequence of states $q_0, q_1, \ldots, q_m$ such that $q_{i+1} = \delta(q_i, \theta_{i+1})$, where for all $i \geq 0$, $\theta_{i+1} \in \Psi_P$ is the unique predicate such that σ_{i+1} satisfies θ_{i+1} and $\delta(q_i, \theta_{i+1})$ is defined.

Consider a $(2^P/\Sigma_O)$-transducer $\mathcal{T} = \langle 2^P, \Sigma_O, Q, q_0, \delta, \tau\rangle$. For a state $q \in Q$ and a signal $p \in P$, we say that p is *sensed in* q if, intuitively, the value of p affects the destination of at least one transition from q. Formally, there exists an assignment $\sigma \in 2^P$ such that $\delta(q, \sigma \setminus \{p\}) \neq \delta(q, \sigma \cup \{p\})$. We use $\mathit{sensed}(q)$ to denote the set of signals sensed in q.

For a set $P' \subseteq P$, let $\sim_{P'} \subseteq 2^P \times 2^P$ be the equivalence relation on 2^P defined by $\sigma \sim_{P'} \sigma'$ iff $\sigma \cap P' = \sigma' \cap P'$. That is, σ and σ' are indistinguishable when one observes only the signals in P'. We say that two words $w \in (2^P)^*$ and $w' \in (2^P)^*$ are *indistinguishable* by $\mathcal{T}$, denoted $w \sim_{\mathcal{T}} w'$, if $w = \sigma_1 \cdots \sigma_m$ and $w' = \sigma'_1 \cdots \sigma'_m$ have the same length, and for all $i \geq 1$, we have that $\sigma_i \sim_{\mathit{sensed}(q_i)} \sigma'_i$, where $q_i = \delta(q_0, \sigma_1 \cdots \sigma_{i-1}) = \delta(q_0, \sigma'_1 \cdots \sigma'_{i-1})$. That is, w and w' are indistinguishable if for every position, the two letters in that position are indistinguishable with respect to the signals sensed in the state reached by $\mathcal{T}$ after reading their prefix up to that position. Note that indistinguishability implies that w and w' induce the same run in $\mathcal{T}$, and so $\mathcal{T}(w) = \mathcal{T}(w')$.

Consider a set P of signals and output alphabet Σ_O. We say that a $(2^P/\Sigma_O)$-transducer $\mathcal{T} = \langle 2^P, \Sigma_O, Q, q_0, \delta, \tau\rangle$ *respects* a $(2^P/2^P)$-transducer $\mathcal{U} = \langle 2^P, 2^P, R, r_0, \delta', \tau'\rangle$ if for every word $w \in (2^P)^*$, the set of signals sensed in the state $\delta(q_0, w)$ is contained in the set of signals $\mathcal{U}(w)$. A *sensing constraint* is a $(2^P/2^P)$-transducer $\mathcal{U}$ that respects itself. That is, for every word $w \in (2^P)^*$, we have that $\mathit{sensed}(\delta'(r_0, w)) \subseteq \mathcal{U}(w)$. Intuitively, a sensing constraint $\mathcal{U}$ specifies the set of signals that can be sensed after reading each word, and the fact that $\mathcal{U}$ respects itself means that a transducer that respects it can follow its constraints (indeed, such a transducer can execute $\mathcal{U}$ despite its partial sensing).

The *sensing cost* of a state $q \in Q$, denoted $\mathit{sense}(q)$, is the number of signals sensed in q, thus $\mathit{sense}(q) = |\mathit{sensed}(q)|$. For a finite run $r = q_0, \ldots, q_m$ of $\mathcal{T}$, with $m \geq 0$, the sensing cost of r, denoted $\mathit{sense}(r)$, is the average number of sensors that $\mathcal{T}$ uses during r, thus $\mathit{sense}(r) = \frac{1}{m+1} \sum_{i=0}^{m} \mathit{sense}(q_i)$. For a word $w \in (2^P)^*$, the sensing cost of w, denoted $\mathit{sense}(w)$, is the sensing cost of the run of $\mathcal{T}$ on w. Finally, the sensing cost of $\mathcal{T}$, denoted $\mathit{sense}(\mathcal{T})$, is the expected sensing cost of words of length that tends to infinity, where we assume that the letters in 2^P are uniformly distributed. That is, in every point in time, each signal in P is assigned `True` with probability $\frac{1}{2}$ and is assigned `False` with probability $\frac{1}{2}$ independently. As detailed below, the value $\mathit{sense}(\mathcal{T})$ can be found by analyzing the *limiting distribution* of a random walk on the Markov chain induced by $\mathcal{T}$.

We first need some definitions. A *Markov chain* $M = \langle S, s_0, \mathrm{P}\rangle$ consists of a finite state space S, an initial state s_0, and a stochastic transition matrix $\mathrm{P} : S \times S \to [0,1]$. That is, for all $s \in S$, we have $\sum_{s' \in S} \mathrm{P}(s,s') = 1$. An infinite random walk on M starts in s_0 and repeatedly move between states according to P. Recall that we assume that signals are assigned `True` and `False` with probability $\frac{1}{2}$. Accordingly, a transducer $\mathcal{T} = \langle 2^P, \Sigma_O, Q, q_0, \delta, \tau\rangle$ induces a Markov chain $M_{\mathcal{T}} = \langle Q, q_0, \mathrm{P}\rangle$, where for all states $q, q' \in Q$, we have $\mathrm{P}(q,q') = \frac{1}{2^{|P|}} |\{\sigma \in 2^P : \delta(q,\sigma) = q'\}|$.

In Appendix A, we detail how each Markov chain $M = \langle S, s_0, \mathrm{P}\rangle$ has a *limiting distribution* $\pi : S \to [0,1]$, mapping each state s to the expected proportion of time that a random walk on M that starts in s_0 and proceeds according to P spends in s. By [14], the limiting distribution

can be computed in polynomial time by solving a system of linear equations. Then, by [1], the sensing cost of a transducer $\mathcal{T}$ can be calculated by examining the sensing cost of its states and their limiting distributions in $M_{\mathcal{T}}$. Specifically, if the limiting distribution of $M_{\mathcal{T}}$ is $\pi : Q \to [0,1]$, then, $\mathit{sense}(\mathcal{T}) = \sum_{q \in Q} \pi(q) \cdot \mathit{sense}(q)$.

For example, the classifier $\mathcal{C}_2$ from Example 1 has $\pi(q) = \frac{1}{4}$ for all states q. Since two states sense p and two states do not sense p, we have that $\mathit{sense}(\mathcal{T}) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$.

▶ Remark 3. It is easy to extend the setting to one in which the assignments to the signals are not uniform or fixed. Rather, they are generated by a Markov chain M_{assign}, each of its states describe the probability of each signal to be assigned `True`. Indeed, in this case we define $M_{\mathcal{T}}$ by taking the product of $\mathcal{T}$ with M_{assign}, and all our results apply. Likely, it is easy to let different signals to have different costs, updating the definition of $\mathit{sense}(q)$ to $\sum_{p \in sensed(q)} \mathit{sense}(p)$. ⌟

3 The Classification Game

In this section we describe the game-theoretic approach to sensing under uncertainty. Given a specification $\mathcal{T}$, we describe a game between Classifier and Environment, such that winning strategies for Classifier correspond to different sensing constraints.

A 2-*player safety game* between Classifier and Environment is $\mathcal{G} = \langle V_C, V_E, v_0, v_\perp, E \rangle$, where V_C and V_E are disjoint sets of vertices, owned by Classifier and Environment, respectively, $v_0 \in V_C$ is an initial vertex, $v_\perp$ is a loosing vertex, and $E \subseteq (V_C \times V_E) \cup (V_E \times (V_C \cup \{v_\perp\}))$ is a total edge relation. In the beginning of a play in the game, a token is placed on v_0. Then, in each turn, the player that owns the vertex that hosts the token chooses a successor vertex and moves the token to it. Together, the players generate a *play* $\rho = v_0, v_1, \ldots$ in G, namely an infinite path that starts in v_0 and respects E: for all $i \geq 0$, we have that $(v_i, v_{i+1}) \in E$. If the play reaches $v_\perp$, it ends, and Environment wins. If the play continues forever, Classifier wins.

A *(positional) strategy* for Classifier is a function $f_C : V_C \to V_E$ that maps each vertex owned by Classifier to a successor to which Classifier moves the token, in a way that respects E. That is, for every $v \in V_C$, we have that $(v, f_C(v)) \in E$. A strategy for Environment is defined similarly, as $f_E : V_E \to V_C \cup \{v_\perp\}$. A *profile* is a tuple $\pi = \langle f_C, f_E \rangle$ of strategies, one for each player. The *outcome* of a profile $\pi = \langle f_C, f_E \rangle$ is the play obtained when the players follow their strategies in π. Formally, $\mathsf{Outcome}(\pi) = v_0, v_1, v_2 \ldots$ is such that for all $j \geq 0$, either $v_j = v_\perp$, or $v_j \in V_C$, in which case $v_{j+1} = f_C(v_0, v_1, \ldots, v_j)$, or $v_j \in V_E$, in which case $v_{j+1} = f_E(v_0, v_1, \ldots, v_j)$. A strategy f_C is *winning* for Classifier if for all strategies f_E of Environment, we have that $\mathsf{Outcome}(\langle f_C, f_E \rangle)$ is winning for Classifier.

Consider a specification $\mathcal{T} = \langle 2^P, \{0, \ldots, k\}, Q, q_0, \delta, \tau \rangle$. Let $\mathit{legit}(\mathcal{T}) \subseteq 2^Q$ be the set of subsets of Q that do not contain states labeled by different labels in $\{1, \ldots, k\}$. Formally, $S \in \mathit{legit}(\mathcal{T})$ iff there is $i \in \{1, \ldots, k\}$ such that for all $q \in S$, we have that $\tau(q) \in \{0, i\}$. Intuitively, $\mathit{legit}(\mathcal{T})$ includes subsets with respect to which uncertainty is legitimate, in the sense that a classifier that does not distinguish between the states in S can still classify all of them to the same language.

We define $\mathcal{G}_{\mathcal{T}}$ as follows. Intuitively, in each round in the game, Classifier decides which signals in P it senses, and Environment generates a letter in (2^P). Together, the players generate a word $w \in (2^P)^\omega$ – the one obtained by concatenating the letters generated by Environment, along with the partial view of Classifier on w. Formally, $\mathcal{G}_{\mathcal{T}} = \langle V_C, V_E, v_0, v_\perp, E \rangle$, where $V_C = \mathit{legit}(\mathcal{T})$, $V_E = \mathit{legit}(\mathcal{T}) \times 2^P$, $v_0 = \{q_0\}$, and E is defined as follows. For a state $s \in Q$, a set $P' \subseteq P$ of sensed signals, and an assignment $\sigma \in 2^P$, we define $\mathit{succ}(s, P', \sigma)$

as the set of possible successors of s when the classifier senses only signals in P' and the environment assigns σ. Formally, $succ(s, P', \sigma) = \{\delta(s, \sigma') : \sigma' \in 2^P, \sigma \sim_{P'} \sigma'\}$. Then, for a set $S \subseteq Q$ of states, we extend this to $succ(S, P', \sigma) = \bigcup_{s \in S} succ(s, P', \sigma)$. This represents the set of states reachable from S given that the classifier observes only signals in P' and the environment chooses assignment σ.

- For every $S \in legit(\mathcal{T})$ and $P' \subseteq P$, we include the edge $\langle S, \langle S, P' \rangle \rangle \in E$. Thus, from a vertex in V_C, Classifier chooses the set P' of signals to sense in the current legitimate set S.
- For a state $\langle S, P' \rangle \in V_E$ and a letter $\sigma \in 2^P$, Environment has a move from $\langle S, P' \rangle$ associated with σ, and the next vertex is determined as follows.
  - If $succ(S, P', \sigma) \in legit(\mathcal{T})$, then $\langle \langle S, P' \rangle, succ(S, P', \sigma) \rangle \in E$.
  - Otherwise, $\langle \langle S, P' \rangle, v_\perp \rangle \in E$.

  In other words, if the successors of the states in S under P' and σ form a legitimate set, the game continues there; otherwise, it moves to $v_\perp$.

Every winning positional strategy f for Classifier induces the sensing constraint $\mathcal{U}_f = \langle 2^P, 2^P, legit(\mathcal{T}), \{q_0\}, \delta', \tau' \rangle$, where for all $S \in legit(\mathcal{T})$ with $f(S) = \langle S, P' \rangle$, we have $\tau'(S) = P'$ and for all $\sigma \in 2^P$, we have $\delta'(S, \sigma) = succ(S, P', \sigma)$. By definition of δ', we have $sensed(S) \subseteq \tau'(S)$, so $\mathcal{U}_f$ respects itself. Moreover, since f is winning, it induces a classifier $\mathcal{C}_f$ that satisfies $\mathcal{T}$ and respects $\mathcal{U}_f$. This classifier is defined on the same structure as $\mathcal{U}_f$ with labeling function κ defined for all $S \in legit(\mathcal{T})$ by $\kappa(S) = \max \tau(S) \cup \{1\}$. Since $S \in legit(\mathcal{T})$, we have $\tau(S) \subseteq \{0, i\}$ for some $i \in \{1, \ldots, k\}$. Thus, $\kappa(S)$ is (arbitrarily) set to 1 if $\tau(S) = \{0\}$, and to i otherwise. By the definitions, we have the following.

▶ **Lemma 4.** *Consider a positional winning strategy f for Classifier in $\mathcal{G}_\mathcal{T}$. The classifier $\mathcal{C}_f$ satisfies $\mathcal{T}$ and respects the sensing constraint $\mathcal{U}_f$.*

4 Classification with Constrained Sensing

The problem of classification with constrained sensing gets as input a language specification $\mathcal{T}$ and a sensing constraint $\mathcal{U}$, both over the same alphabet 2^P, and has to return a classifier that satisfies $\mathcal{T}$ and respects $\mathcal{U}$, or determine that no such classifier exists.

In order to solve the problem, we first augment the states of $\mathcal{T}$ by states of $\mathcal{U}$, indicating the set of signals that may be sensed. Let $\mathcal{T} = \langle 2^P, \{0, \ldots, k\}, Q, q_0, \delta, \tau \rangle$, and consider a sensing constraint $\mathcal{U} = \langle 2^P, 2^P, R, r_0, \delta', \tau' \rangle$. We define the specification $\mathcal{T}^\mathcal{U} = \langle 2^P, \{0, \ldots, k\}, Q \times R, \langle q_0, r_0 \rangle, \delta^\mathcal{U}, \tau^\mathcal{U} \rangle$, where for every state $\langle q, r \rangle \in Q \times R$ and letter $\sigma \in 2^P$, we have that $\delta^\mathcal{U}(\langle q, r \rangle, \sigma) = \langle \delta(q, \sigma), \delta'(r, \sigma) \rangle$. Also, $\tau^\mathcal{U}(\langle q, r \rangle) = \tau(q)$. It is easy to see that since $\mathcal{U}$ is a deterministic transducer with transitions defined for all letters in 2^P, the specifications $\mathcal{T}$ and $\mathcal{T}^\mathcal{U}$ are equivalent, in the sense that for all $w \in (2^P)^*$, we have that $\mathcal{T}(w) = \mathcal{T}^\mathcal{U}(w)$.

Consider the game $\mathcal{G}_{\mathcal{T}^\mathcal{U}} = \langle V_C, V_E, v_0, v_\perp, E \rangle$, namely the game defined in Section 3 for the specification $\mathcal{T}^\mathcal{U}$. Recall that $V_C = legit(\mathcal{T}^\mathcal{U})$, and so each state in V_C is a subset $S \subseteq Q \times R$. We say that S is *good* if there is $r \in R$ such that $S \subseteq Q \times \{r\}$. That is, all the states in S agree on their state in $\mathcal{U}$. We call r the *witness* of S.

Let $f_\mathcal{U} : V_C \to V_E$ be a strategy for Classifier that follows $\mathcal{U}$ in all good positions. That is, if S is good with witness r, then $f_\mathcal{U}(S) = \langle S, \tau'(r) \rangle$. By definition, the vertex $v_0 = \{\langle q_0, r_0 \rangle\}$ is good, with witness r_0. Moreover, following $f_\mathcal{U}$ is the only way for Classifier to win with a strategy that respects $\mathcal{U}$, and it is guaranteed that all vertices visited along a play in $\mathcal{G}_{\mathcal{T}^\mathcal{U}}$ are good. As Classifier wins a safety game iff it has a positional winning strategy [8], we have the following:

▶ **Theorem 5.** *Consider a specification $\mathcal{T}$ and a sensing constraint $\mathcal{U}$. Let $\mathcal{G}_{\mathcal{T}^{\mathcal{U}}}$ be the game between Classifier and Environment that correspond to the specification $\mathcal{T}^{\mathcal{U}}$. There is a classifier that satisfies $\mathcal{T}$ and respects $\mathcal{U}$ iff the strategy $f_{\mathcal{U}}$ is winning in $\mathcal{G}_{\mathcal{T}^{\mathcal{U}}}$.*

Theorem 5 implies that the problem of deciding whether there is a classifier that satisfies $\mathcal{T}$ and respects $\mathcal{U}$ can be solved by reasoning about $\mathcal{T}^{\mathcal{U}}$. As good and surprising news, we show that even though the state space of $\mathcal{G}_{\mathcal{T}^{\mathcal{U}}}$ is exponential in $\mathcal{T}$, one need not traverse the whole state space of $\mathcal{G}_{\mathcal{T}^{\mathcal{U}}}$ and the problem is NLOGSPACE-complete.

▶ **Lemma 6.** *Consider a specification $\mathcal{T}$ and a sensing constraint $\mathcal{U}$ with state spaces Q and R, respectively. The following are equivalent.*

(1) *There is no classifier that satisfies $\mathcal{T}$ and respects $\mathcal{U}$.*
(2) *There are words $w, w' \in (2^P)^*$ such that $w \sim_{\mathcal{U}} w'$ and $\mathcal{T}(w) \not\sim_{\mathcal{T}} \mathcal{T}(w')$.*
(3) *There are words $w, w' \in (2^P)^*$ such that $|w| = |w'| \leq |Q|^2 \cdot |R|$, $w \sim_{\mathcal{U}} w'$, and $\mathcal{T}(w) \not\sim_{\mathcal{T}} \mathcal{T}(w')$.*

Proof. The implications (1) → (2) and (3) → (1) follow easily from the fact that (1) holds iff the strategy $f_{\mathcal{U}}$ is losing in $\mathcal{G}_{\mathcal{T}^{\mathcal{U}}}$ (see details in the full version).

We prove here that (2) → (3). Consider words $w, w' \in (2^P)^*$ such that $w \sim_{\mathcal{U}} w'$, and $\mathcal{T}(w) \not\sim_{\mathcal{T}} \mathcal{T}(w')$. Since $w \sim_{\mathcal{U}} w'$, the runs of $\mathcal{U}$ on w and w' coincides. Let $r_0, r_1, \ldots, r_m$ be this run and let $q_0, q_1, \ldots, q_m$ and $q'_0, q'_1, \ldots, q'_m$ be the runs of $\mathcal{T}$ on w and w', respectively. Assume that $m > |Q|^2 \cdot |R|$. Then, there are $0 \leq i_1 < i_2 \leq m$ such that $\langle r_{i_1}, q_{i_1}, q'_{i_1} \rangle = \langle r_{i_2}, q_{i_2}, q'_{i_2} \rangle$. Thus, we can shorten w and w' by removing the subword between positions $i_1 + 1$ and i_2, and obtain shorter words that still satisfy the same properties. Repeating this process, we can obtain words $x, x' \in (2^P)^*$ of length at most $|Q|^2 \cdot |R|$ such that $x \sim_{\mathcal{U}} x'$ and $\mathcal{T}(x) \not\sim_{\mathcal{T}} \mathcal{T}(x')$. ◀

▶ **Theorem 7.** *Consider a specification $\mathcal{T}$ and a sensing constraint $\mathcal{U}$. The problem of deciding whether there is a classifier that satisfies $\mathcal{T}$ and respects $\mathcal{U}$ is NLOGSPACE-complete.*

Proof. Let $\mathcal{T} = \langle 2^P, \{0, \ldots, k\}, Q, q_0, \delta, \tau \rangle$ and $\mathcal{U} = \langle 2^P, 2^P, R, r_0, \delta', \tau' \rangle$. We prove that the complementing problem is NLOGSPACE. By Lemma 6, there is no classifier that satisfies $\mathcal{T}$ and respects $\mathcal{U}$ iff there are words $w, w' \in (2^P)^*$ such that $w, w' \in (2^P)^*$ such that $|w| = |w'| \leq |Q|^2 \cdot |R|$ and $w \sim_{\mathcal{U}} w'$ and $\mathcal{T}(w) \not\sim_{\mathcal{T}} \mathcal{T}(w')$. A nondeterministic logarithmic-space Turing machine can guess such words w and w' letter by letter, maintaining in each iteration the states $r \in R$, $q \in Q$, and $q' \in Q$, of the runs of $\mathcal{U}$ and $\mathcal{T}$ on w and w', and a counter of the length of the generated words. Note that the constraint $w \sim_{\mathcal{U}} w'$ is fulfilled by guessing the next letters $\sigma_i \in 2^P$ and $\sigma'_i \in 2^P$, of w and w' respectively, so that $\sigma_i \sim_{\tau'(r_{i-1})} \sigma'_i$. The machine accepts if at some point the generated word satisfy the conditions $\tau(q) \neq \tau(q')$ and $\tau(q), \tau(q') \neq 0$, and rejects when the counter exceeds $|Q|^2 \cdot |R|$.

Hardness in NLOGSPACE is by an easy reduction from non-reachability in a directed graph: Given a directed graph $G = \langle V, E \rangle$ and two vertices $s, t \in V$, let 2^P encode the vertices in V, and for every vertex $v \in V$, let $\sigma_v \in 2^P$ be the letter that encodes v. Now, let $\mathcal{T}$ map to 1 all words in $\sigma_s \cdot \texttt{True}^* \cdot \sigma_t$ that correspond to paths in G and map to 2 all words in $\sigma_s \cdot (2^P \setminus \{\sigma_t\})^*$. Then, let $\mathcal{U}$ enable no sensing. It is easy to see that there is a classifier for $\mathcal{T}$ that respects $\mathcal{U}$ iff there is no path from s to t in G. Indeed, such a classifier may map all words to 2. Also, both $\mathcal{T}$ and $\mathcal{U}$ can be generated in logarithmic space from G. ◀

The low complexity of the problem is especially surprising given that a minimal classifier that satisfies $\mathcal{T}$ and respects $\mathcal{U}$ may be exponential in $\mathcal{T}$, as we state below.

▶ **Theorem 8.** *For every specification $\mathcal{T}$ and sensing constraint $\mathcal{U}$, if there is a classifier that satisfies $\mathcal{T}$ and respects $\mathcal{U}$, then there is also one of size exponential in $\mathcal{T}$ and polynomial in $\mathcal{U}$. This bound is tight: For every $n \geq 1$ there is a set of signals P_n of size $O(n)$, a $(2^{P_n}/\{0,1,2\})$-transducer $\mathcal{T}_n$ with $O(n)$ states, and a sensing constraint $\mathcal{U}_n$ with 2 states, such that every classifier that satisfies $\mathcal{T}_n$ and respects $\mathcal{U}_n$ has at least 2^n states.*

Proof. By Theorem 5, a classifier that satisfies $\mathcal{T}$ and respects $\mathcal{U}$ can be defined on top of the state space of $\mathcal{G}_{\mathcal{T}^{\mathcal{U}}}$, restricted to good sets in $\mathit{legit}(\mathcal{T}^{\mathcal{U}})$. Thus, the upper bound follows from the bound on the number of such sets.

We continue to the lower bound. For $n \geq 0$, let $[n] = \{0, 1, \ldots, n-1\}$. We define $\mathcal{T}_n = \langle 2^{P_n}, \{0,1,2\}, Q_n, q_0, \delta_n, \tau\rangle$ as follows.

- $P_n = \{s_0, \ldots, s_{n-1}, d_1, \ldots, d_{n-1}\}$ consists of n *s-signals* and $n-1$ *d-signals*. We interpret a valuation $v \in 2^{P_n}$ in two ways. The *odd interpretation* to v, denoted $\mathit{odd}(v)$, is the pair $\langle i, j\rangle \in [n] \times \{0,1\}$ defined as follows. Let $i = \sum_{\ell=1}^{n-1} v(d_\ell)$ be the number of d-signals assigned true in v. Then, $\mathit{odd}(v) = \langle i, v(s_i)\rangle$. The *even interpretation* of v, denoted $\mathit{even}(v)$, is the pair $\langle i, j\rangle \in [n]\times\{0,1\}$ defined as follows. Let $k = \sum_{\ell=0}^{n-1} v(s_\ell) + \sum_{\ell=1}^{n-1} v(d_\ell)$ be the total number of s- and d-signals assigned true in v. Let $i = \lfloor k/2 \rfloor$ and $j = k \pmod 2$. Then, $\mathit{even}(v) = \langle i, j\rangle$. Note that the even and odd interpretations both define surjective mappings $2^{P_n} \to [n] \times \{0,1\}$.
  Given two interpretations $\langle i, j\rangle, \langle i', j'\rangle \in [n] \times \{0,1\}$, we say that $\langle i, j\rangle$ and $\langle i', j'\rangle$ are a *couple*, denoted $\langle i, j\rangle \approx \langle i', j'\rangle$, iff $i = i'$ and $j \neq j'$. Note that $[n] \times \{0,1\}$ is partitioned into n couples. Also, for two valuations $v, u \in 2^{P_n}$, we have $\mathit{odd}(v) \approx \mathit{odd}(u)$ iff the number $i \in [n]$ of true d-signals is the same for both valuations, and yet $v(s_i) \neq u(s_i)$. In particular, for every $v \in 2^{P_n}$ there exists a valuation $u \in 2^{P_n}$ that is different from v only in the value of one s-signal, such that $\mathit{odd}(v) \approx \mathit{odd}(u)$. For the even interpretation, we have $\mathit{even}(v) \approx \mathit{even}(u)$ iff the numbers k_v and k_u of true s- and d-signals in v and u respectively, satisfy $\{k_v, k_u\} = \{2i, 2i+1\}$ for some $i \in [n]$. In particular, for every $v \in 2^{P_n}$, there exists a valuation $u \in 2^{P_n}$ that is different from v only in the value of one signal, such that $\mathit{even}(v) \approx \mathit{even}(u)$.
- The state space $Q_n = \{q_0, q_1, q_2\} \cup [n] \times \{0,1\}$ is partitioned into the set $\{q_0, q_1, q_2\}$ of *odd states*, and the set $\{\langle i, j\rangle : i \in [n], j \in \{0,1\}\}$ of *even states*.
- The transition function δ_n is defined as follows for every state $q \in Q_n$ and letter $v \in 2^{P_n}$:

$$\delta_n(q, v) = \begin{cases} \mathit{odd}(v) & \text{if } q \in \{q_0, q_1, q_2\} \\ q_1 & \text{if } q = \mathit{even}(v) \\ q_2 & \text{if } q \approx \mathit{even}(v) \\ q_0 & \text{otherwise} \end{cases}$$

  It is not hard to see that $\mathcal{T}_n$ alternates between even and odd states, and so when reading a word $w = v_1 \cdots v_m$, the transition on v_i is defined with respect to the odd interpretation of v_i if i is odd, and with respect to the even interpretation of v_i if i is even. Since the transition function is defined with respect to the even or odd interpretation of valuations, we sometimes write $\delta_n(q, \langle i, j\rangle)$ to denote the transition from a state q on a letter v such that $\mathit{even}(v) = \langle i, j\rangle$ or $\mathit{odd}(v) = \langle i, j\rangle$, depending on whether q is an even or odd state. Similarly, we refer to letters in 2^{P_n} as elements of $[n] \times \{0,1\}$ (where the interpretation is the even or odd interpretation, depending on the context).
- The labeling function τ is such that $\tau(q_1) = 1, \tau(q_2) = 2$, and for all other states $s \in Q_n$, we have $\tau(s) = 0$.

Consider a word $w \in (2^{P_n})^*$ of even length. The alternation between even and odd states implies that $\delta_n(q_0, w) = q \in \{q_0, q_1, q_2\}$. By the definition of δ_n, we have that for every $v_1, v_2 \in 2^{P_n}$

$$\delta_n(q_0, w \cdot v_1 \cdot v_2) = \delta_n(q, v_1 \cdot v_2) = \delta_n(\mathit{odd}(v_1), v_2) = \begin{cases} q_1 & \text{if } \mathit{odd}(v_1) = \mathit{even}(v_2) \\ q_2 & \text{if } \mathit{odd}(v_1) \approx \mathit{even}(v_2) \\ q_0 & \text{otherwise.} \end{cases}$$

Accordingly, the two languages L_1^n and L_2^n defined by $\mathcal{T}_n$ are

- $L_1^n = \{(\texttt{True} \cdot \texttt{True})^* \cdot v_1 \cdot v_2 : \mathit{odd}(v_1) = \mathit{even}(v_2)\}$.
- $L_2^n = \{(\texttt{True} \cdot \texttt{True})^* \cdot v_1 \cdot v_2 : \mathit{odd}(v_1) \approx \mathit{even}(v_2)\}$.

That is, L_1^n includes all words of the form $w \cdot v_1 \cdot v_2$ where w is a word of an even length, and the odd interpretation of v_1 is the same as the even interpretation of v_2. Likewise, L_2^n includes all words of the form $w \cdot v_1 \cdot v_2$ where w is a word of an even length, and the odd interpretation of v_1 is a couple of the even interpretation of v_2.

Consider the 2-state sensing constraint $\mathcal{U}_n$ that disables sensing all d-signals at odd positions, and allows sensing all other signals at all times. Formally, $\mathcal{U}_n = \langle 2^{P_n}, 2^{P_n}, \{r_0, r_1\}, r_0, \delta'_n, \tau'_n \rangle$, where $\delta'_n(r_0, \sigma) = r_1$ and $\delta'_n(r_1, \sigma) = r_0$ for all $\sigma \in 2^{P_n}$, $\tau'_n(r_0) = \{s_0, \ldots, s_{n-1}\}$, and $\tau'_n(r_1) = P_n$. In the full version, we prove that there exists a classifier $\mathcal{C}_n^*$ that satisfies $\mathcal{T}_n$ and respects $\mathcal{U}_n$, yet every such classifier has at least 2^n states. Intuitively, the sensing constraint $\mathcal{U}_n$ forces a correct classifier for $\mathcal{T}_n$ to maintain in its memory the last assignment to the sensed s-signals after every odd position. Indeed, for every two assignments S_1 and S_2 to the s-signals, there are two assignments $v_1 \in 2^{P_n}$ and $v_2 \in 2^{P_n}$ that extend S_1 and S_2, respectively, and there exists a letter $u \in 2^{P_n}$ such that $\mathit{odd}(v_1) = \mathit{even}(u)$ and $\mathit{odd}(v_2) \approx \mathit{even}(u)$. Therefore, a correct classifier must move to different states upon reading v_1 and v_2 from every odd position. Since there are 2^n different assignments to the s-signals, this implies that there are at least 2^n states in the classifier. ◀

5 Classification with Privacy

The input to the problem of *classification with privacy* includes, in addition to the language specification, also a *secret*, given by a $(2^P/\{0, 1, \ldots, m\})$-transducer $\mathcal{H}$, for some $m \geq 2$. We say that a sensing constraint $\mathcal{U}$ *hides* $\mathcal{H}$ if for every word $w \in (2^P)^*$ such that $\mathcal{H}(w) \neq 0$, there exists a word $w' \in (2^P)^*$ such that $w \sim_{\mathcal{U}} w'$ and $\mathcal{H}(w) \neq \mathcal{H}(w')$. That is, when a classifier respects $\mathcal{U}$, it cannot deduce the labeling that $\mathcal{H}$ gives to input words it cares about. We say that a classifier $\mathcal{C}$ hides $\mathcal{H}$ if it respects a sensing constraint that hides $\mathcal{H}$. Our goal is to find, given a language specification $\mathcal{T}$ and a secret $\mathcal{H}$, a classifier that satisfies $\mathcal{T}$ and hides $\mathcal{H}$.

▶ **Example 9.** Recall the specification $\mathcal{T}$ from Example 1, and consider the secret $\mathcal{H}$ in Figure 2. Note that for all words $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_n \in \{p, \bar{p}\}^*$, we have that $\mathcal{H}(w) = 1$ iff $n = 1 \pmod 3$ and $\sigma_n = p$. It is not hard to see that the classifiers $\mathcal{C}_1$ and $\mathcal{C}_2$ from Example 1 do not hide $\mathcal{H}$. Indeed, both reveal the value of the first assignment to p. Consider the classifier $\mathcal{C}_3$, also in Figure 2.

Note that $\mathcal{C}_3$ is a correct classifier for $\mathcal{T}$. Indeed, $\mathcal{C}_3$ makes use of the fact that it is possible to satisfy $\mathcal{T}$ while reading only one letter in each suffix $\sigma_{2i-1} \cdot \sigma_{2i}$, for all $i \geq 1$. Indeed, if one of these letters is p, then $\sigma_1 \cdots \sigma_{2i}$ can be mapped to 1, and if one of them is $\bar{p}$, then it can be mapped to 2. Since for all $i \geq 1$, either $2i - 1$ or $2i$ is not 1 mod 3, it is thus possible to satisfy $\mathcal{T}$ without revealing $\mathcal{H}$. Formally, $\mathcal{C}_3$ respects a sensing constraint $\mathcal{U}$ that senses σ_i iff $i \neq 1 \bmod 3$. Hence, every word w such that $|w| = 1 \pmod 3$ (that is, words that $\mathcal{H}$ cares about) has a word w' such that $w \sim_{\mathcal{U}} w'$ and $\mathcal{H}(w) \neq \mathcal{H}(w')$. ⌟

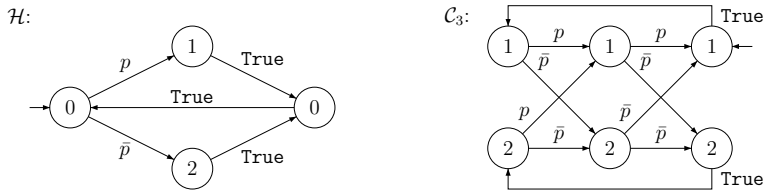

Figure 2 The secret $\mathcal{H}$ and the classifier $\mathcal{C}_3$ that satisfies $\mathcal{T}$ and hides $\mathcal{H}$.

▶ **Theorem 10.** *Classification with privacy can be solved in EXPTIME.*

Proof. Given a language specification $\mathcal{T}$ and a secret $\mathcal{H}$, we reduce classification with privacy to the game defined in Section 3, except that now the definition of legitimate sets reflects also the need to hide the secret.

For simplicity, we assume that $\mathcal{T}$ and $\mathcal{H}$ are defined over the same structure and differ only on their labeling. As explained below, this can be obtained by taking the product of $\mathcal{T}$ and $\mathcal{H}$. Formally, given $\mathcal{T} = \langle 2^P, \{0,1,\ldots,k\}, Q, q_0, \delta, \tau_\mathcal{T}\rangle$ and $\mathcal{H} = \langle 2^P, \{0,1,\ldots,m\}, Q', q'_0, \delta', \tau_\mathcal{H}\rangle$, we define the product $\mathcal{T} \times \mathcal{H} = \langle 2^P, \{0,\ldots,k\} \times \{0,\ldots,m\}, Q \times Q', \langle q_0, q'_0\rangle, \delta'', \tau\rangle$, which is a $(2^P/\{0,\ldots,k\} \times \{0,\ldots,m\})$-transducer defined as follows. The transition function δ'' is the product of δ and δ', i.e., for every $\langle q, q'\rangle \in Q \times Q'$ and $\sigma \in 2^P$, we have that $\delta''(\langle q, q'\rangle, \sigma) = \langle \delta(q,\sigma), \delta'(q',\sigma)\rangle$. The labeling function τ is defined by $\tau(\langle q, q'\rangle) = \langle \tau_\mathcal{T}(q), \tau_\mathcal{H}(q')\rangle$ for every $\langle q, q'\rangle \in Q \times Q'$. It is not hard to see that by projecting the labels of $\mathcal{T} \times \mathcal{H}$ to the first and second components, we can recover $\mathcal{T}$ and $\mathcal{H}$, respectively, yet with a shared structure.

Let Q be the shared state space of $\mathcal{T}$ and $\mathcal{H}$. That is, $\mathcal{T} = \langle 2^P, \{0,1,\ldots,k\}, Q, q_0, \delta, \tau_\mathcal{T}\rangle$ and $\mathcal{H} = \langle 2^P, \{0,1,\ldots,m\}, Q, q_0, \delta, \tau_\mathcal{H}\rangle$. A set $S \subseteq Q$ is in $\mathit{legit}(\mathcal{T}, \mathcal{H})$ if $S \in \mathit{legit}(\mathcal{T})$ and $S \notin \mathit{legit}(\mathcal{H})$. That is, there exists $i \in \{1,\ldots,k\}$ such that $\tau_\mathcal{T}(S) \subseteq \{0, i\}$, and there exists $1 \leq j < j' \leq m$ such that $\{j, j'\} \subseteq \tau_\mathcal{H}(S)$.

The problem of classification with privacy can be solved by solving the safety game $\mathcal{G}_{\mathcal{T},\mathcal{H}}$ defined similarly to $\mathcal{G}_\mathcal{T}$ in Section 3, except that now V_C is the set $\mathit{legit}(\mathcal{T}, \mathcal{H})$ rather than $\mathit{legit}(\mathcal{T})$. Since $\mathcal{G}_{\mathcal{T},\mathcal{H}}$ has $2^{O(|Q|)}$ vertices, and solving safety games can be done in linear time, the upper bound follows. ◀

▶ Remark 11 (Richer secrets). The results in this section can easily be extended to richer notions of privacy. Essentially, any definition that can be captured by an appropriate definition of legitimate subsets can be handled. First, *more secrets* may be added to the product, adding to the requirement of legitimate sets not to be able to classify words also according to the new secrets. Second, in our current definition, a classifier cannot classify an observed word, yet it may learn which classifications are impossible for it. We can change the definition of legitimate sets ensuring the classifier learns nothing about the secret: for every word w such that $\mathcal{H}(w) \neq 0$, there exist words $w_1, w_2, \ldots, w_m$ such that for every $i \in \{1,\ldots,m\}$, we have that $w \sim_\mathcal{U} w_i$ and $\mathcal{H}(w_i) = i$. Finally, we can allow *conditional secrets*: rather than hiding the classification to all words that the secret cares about, we may require the hiding of all words that satisfy some regular property, in particular these that the secret classify to some value. ⌟

Also in the case of privacy, the game induces a bound on the size of a correct classifier, which we show to be tight.

▶ **Theorem 12.** *For every specification $\mathcal{T}$ and secret $\mathcal{H}$, if there is a classifier that satisfies $\mathcal{T}$ and hides $\mathcal{H}$, then there is also one of size exponential in $\mathcal{T}$ and polynomial in $\mathcal{H}$. This bound is tight: For every $n \geq 1$ there is a set of signals P_n of size $O(n)$, a $(2^{P_n}/\{0,1,2\})$-transducer $\mathcal{T}_n$ with $O(n)$ states, and a secret $\mathcal{H}_n$ with 3 states, such that every classifier that satisfies $\mathcal{T}_n$ and hides $\mathcal{H}_n$ has at least 2^n states.*

Proof. The upper bound follows from the size of the safety game $\mathcal{G}_{\mathcal{T},\mathcal{H}}$. For the lower bound, consider the family of specifications $\mathcal{T}_n$ defined in Theorem 8. Recall that $\mathcal{T}_n$ is over the set of signals $P_n = \{s_0, \ldots, s_{n-1}, d_1, \ldots, d_{n-1}\}$. By Theorem 8, every classifier that satisfies $\mathcal{T}_n$ and respects the sensing constraint $\mathcal{U}_n$ that disables the sensing of all d-signals at odd positions and allows the sensing of all other signals at all positions, has at least 2^n states.

Consider the secret $\mathcal{H}_n$ that induces the languages

- $H_1^n = (\texttt{True} \cdot \texttt{True})^* \cdot (d_1 \vee d_2 \vee \cdots \vee d_{n-1})$, and
- $H_2^n = (\texttt{True} \cdot \texttt{True})^* \cdot (\bar{d_1} \wedge \bar{d_2} \wedge \cdots \wedge \bar{d_{n-1}})$.

It is not hard to see that $\mathcal{H}_n$ can be defined by a transducer with three states, and that a classifier hides $\mathcal{H}_n$ iff it disables sensing all d-signals at odd positions, and so it respects $\mathcal{U}_n$. Thus, by Theorem 8, every classifier that satisfies $\mathcal{T}_n$ and hides $\mathcal{H}_n$ has at least 2^n states. ◀

▶ **Theorem 13.** *The problem of classification with privacy is PSPACE-hard.*

Proof. We reduce the problem of NFA universality to classification with privacy. NFA universality is PSPACE-hard already for an NFA $\mathcal{N}$ over an alphabet of size 2, with a single accepting initial state, nondeterminism of branching degree 2, and in which every non-empty word has a rejecting run.

Let $\mathcal{N} = \langle \{p, \bar{p}\}, Q, q_0, \delta, \alpha \rangle$ be such an NFA. As δ has branching degree 2, we refer to it as a function $\delta : Q \times \{p, \bar{p}\} \times \{b, \bar{b}\} \to Q$, where b is a signal encoding the nondeterministic choices of $\mathcal{N}$. That is, for every $q \in Q$ and $\sigma \in \{p, \bar{p}\}$, we have that $\delta(q, \sigma) = \{\delta(q, \sigma, b), \delta(q, \sigma, \bar{b})\}$.

We construct a specification $\mathcal{T}$ and a secret $\mathcal{H}$ such that there is a classifier for $\mathcal{T}$ that does not reveal $\mathcal{H}$ iff $L(\mathcal{N}) = \{p, \bar{p}\}^*$. Essentially, $\mathcal{T}$ is over the set of signals $P = \{p, b\}$. Thus, a word $w \in (2^P)^*$ combines a word $word(w) \in \{p, \bar{p}\}^*$ along with information on how to resolve nondeterminism in a run of $\mathcal{N}$ on $word(w)$. Let $run(w)$ be the run of $\mathcal{N}$ on $word(w)$, namely the run obtained when nondeterminism is resolved by the assignments to the signal b. The specification $\mathcal{T}$ has width 2, and it induces the languages $L_1 = \texttt{True}^* \cdot p$ and $L_2 = \texttt{True}^* \cdot \bar{p}$. The secret $\mathcal{H}$ has width 2 too, and it induces the languages H_1 of words w such that $run(w)$ is accepting, and H_2 of words w such that $run(w)$ is rejecting.

In order to successfully classify words according to $\mathcal{T}$, a classifier has to constantly sense p. Recall that $\mathcal{N}$ has a rejecting run on every word in $\{p, \bar{p}\}^*$. It also has an accepting run on every word in $\{p, \bar{p}\}^*$ iff $\mathcal{N}$ is universal. Thus, a classifier that only senses p can satisfy $\mathcal{T}$ without revealing $\mathcal{H}$ iff $\mathcal{N}$ is universal. Indeed, $\mathcal{N}$ is universal iff every word $w \in (2^P)^*$ has a word $w' \in (2^P)^*$ such that $w \sim_{\{p\}} w'$ and $word(w)$ is accepting iff $word(w')$ is rejecting.

In the full version, we describe $\mathcal{T}$ and $\mathcal{H}$ in detail and prove the reduction. ◀

6 Classification with Minimal Sensing

In this section we study the problem of generating correct classifiers with minimal sensing cost. Note that two transducers may define the same mapping and have different sensing costs. For a given regular mapping $\tau : (2^P)^* \to \{1, \ldots, k\}$, the sensing cost of τ, denoted $sense(\tau)$ is the minimal sensing cost required for a transducer that defines τ by a transducer. Thus, $sense(\tau) = \inf\{sense(\mathcal{C}) : \mathcal{C} \text{ defines } \tau\}$. As proven in [1], $sense(\tau)$ is attained in the minimal (size-wise) transducer that defines τ. Thus, for a given mapping τ, there is no trade-off between minimizing size and minimizing sensing, and a minimally-sensing transducer that defines τ can be found in time polynomial in some transducer that defines τ. In the classification problem, however, the mapping τ is not given, as we have freedom with respect to words that the specification $\mathcal{T}$ maps to 0. In the case of size minimization, going over all correct mappings leads to an algorithm in NP, and the problem is NP-complete. As we show, the case of sensing minimization is more involved.

6.1 Trade-off Size and Sensing in Separation

As demonstrated in Example 1, minimal sensing may be attained by a classifier that is not of minimal size. Here, we strengthen the size-sensing trade-off in two ways. We show that it is unbounded and that a minimally-sensing classifier may be exponentially bigger than its specification.

▶ **Theorem 14.** *For every $n \geq 2$, there is a $(2^{\{p\}}/\{0,1,2\})$-transducer $\mathcal{T}_n$ with $O(n)$ states such that there is a two-state classifier for $\mathcal{T}_n$, every two-state classifier for $\mathcal{T}_n$ has sensing cost 1, and there is a $2n$-state classifier for $\mathcal{T}_n$ with sensing cost $\frac{1}{n}$.*

Proof. We generalize the transducer from Example 1 so that $\mathcal{T}_n$ defines the languages $L_1^n = (\texttt{True}^n)^* \cdot p^n$ and $L_2^n = (\texttt{True}^n)^* \cdot \bar{p}^n$. (Note that Example 1 corresponds to the case $n = 2$).

In the full version, we describe $\mathcal{T}_n$ in detail. It is not hard to see that the two-state classifier $\mathcal{C}_1$ from Example 1 can serve as a classifier for $\mathcal{T}_n$ for all $n \geq 2$. Indeed $\mathcal{C}_1$ maps to 1 exactly all words ending in p, and so it is still the case that all wards in L_1^n are mapped to 1 and all words in L_2^n are mapped to 2. In the full version, we prove that every two-state classifier $\mathcal{C}$ with $\mathit{sense}(\mathcal{C}) < 1$, namely a classifier that does not sense p in at least one of its states, cannot satisfy $\mathcal{T}_n$. Essentially, such a classifier must contain a problematic cycle.

Finally, a classifier with $2n$-states can map to 1 exactly all words in $(\texttt{True}^n)^*.p.\texttt{True}^{n-1}$, sensing p only once every n transitions, thus having sensing cost $\frac{1}{n}$. In the full version, we describe such a classifier in detail. ◀

We continue and prove that a minimally-sensing classifier for a given language classification $\mathcal{T}$ may be exponentially bigger than $\mathcal{T}$. Specifically, we prove the following theorem.

▶ **Theorem 15.** *For every $n \geq 1$, there is a specification $\mathcal{T}_n$ with $O(n)$ states such that $\mathcal{T}_n$ can be satisfied by a classifier with sensing cost $\frac{3n-1}{2}$, yet every classifier $\mathcal{C}_n$ that satisfies $\mathcal{T}_n$ and has $\mathit{sense}(\mathcal{C}_n) \leq \frac{3n-1}{2}$ needs at least 2^n states.*

Proof. Recall the family of transducers $\mathcal{T}_n$ defined in the proof of Theorem 8. In the full version, we prove that the classifier $\mathcal{C}_n^*$ described there has sensing cost $\frac{3n-1}{2}$, and that every classifier $\mathcal{C}_n$ for $\mathcal{T}_n$ with sensing cost $\frac{3n-1}{2}$ has at least 2^n states. ◀

6.2 Finding A Classifier with Minimal Expected Sensing

In this section we study the problem of generating a classifier with minimal sensing cost. Recall the 2-player game $\mathcal{G}_\mathcal{T}$ that embodies possible classifiers for a given specification $\mathcal{T}$. A 1.5-*player game* is similar to a 2-player game, except that now Environment is probabilistic, and we refer to it as *Nature*. In addition, in order to reason about the expected sensing, we augment the states of the game by a cost function, describing the number of signals sensed in each chose of Classifier, and study the *mean-payoff* of plays in the game.

Formally, given $\mathcal{T} = \langle 2^P, \{0,\ldots,k\}, Q, q_0, \delta, \tau\rangle$, we define the 1.5-player safety mean-payoff game $\mathcal{M}_\mathcal{T} = \langle V_C, V_E, v_0, v_\perp, E, \mathrm{P}, \mathit{cost}\rangle$ as follows. As in $\mathcal{G}_\mathcal{T}$, we have that $V_C = \mathit{legit}(\mathcal{T})$, $V_E = \mathit{legit}(\mathcal{T}) \times 2^P$, $v_0 = \{q_0\}$, and for each $S \in \mathit{legit}(\mathcal{T})$ and $P' \subseteq P$, we have that $\langle S, \langle S, P'\rangle\rangle \in E$. In addition $\mathrm{P} : V_E \times (V_C \cup \{v_\perp\}) \to [0,1]$ is a probabilistic transition function, defining for every two states $s \in V_E$ and $s' \in V_C \cup \{v_\perp\}$, the probability of Nature moving the token from s to s'. Since we assume a uniform distribution on valuations to the input signals, the function P is defined as follows. Consider a state $\langle S, P'\rangle \in V_E$.

- For every set $S' \in \mathit{legit}(\mathcal{T})$, we have that $\mathrm{P}(\langle S, P'\rangle, S') = \frac{|\{\sigma \in 2^P : \mathit{succ}(S,P',\sigma) = S'\}|}{2^{|P|}}$. That is, the probability of transitioning from $\langle S, P'\rangle$ to S' is the fraction of assignments $\sigma \in 2^P$ such that $\mathit{succ}(S, P', \sigma) = S'$.
- In addition, $\mathrm{P}(\langle S, P'\rangle, v_\bot) = \frac{|\{\sigma \in 2^P : \mathit{succ}(S,P',\sigma) \notin \mathit{legit}(\mathcal{T})\}|}{2^{|P|}}$. That is, the probability of transitioning from $\langle S, P'\rangle$ to $v_\bot$ is the fraction of assignments $\sigma \in 2^P$ such that $\mathit{succ}(S, P', \sigma)$ is not in $\mathit{legit}(\mathcal{T})$.

Note that in both cases, the probability may be 0. Finally, the cost function $\mathit{cost} : V_E \to \{0, \ldots, |P|\}$ is defined for every state in V_E by $\mathit{cost}(\langle S, P'\rangle) = |P'|$.

A positional strategy $f : V_C \to V_E$ for Classifier induces a Markov Chain $\mathcal{M}^f_\mathcal{T} = \langle V_C \cup \{v_\bot\}, \{q_0\}, \mathrm{P}_f\rangle$, where for every two states $S, S' \in V_C$, we have that $\mathrm{P}_f(S, S') = \mathrm{P}(\langle S, f(S)\rangle, S')$. In addition, $\mathrm{P}_f(S, v_\bot) = \mathrm{P}(\langle S, f(S)\rangle, v_\bot)$. That is, $\mathcal{M}^f_\mathcal{T}$ describes the expected behavior of $\mathcal{M}_\mathcal{T}$ when Classifier follows f. The *cost* of f is then the expected average cost of a random walk in $\mathcal{M}^f_\mathcal{T}$. Again, the cost of f can be calculated from the limiting distribution of $\mathcal{M}^f_\mathcal{T}$ (see e.g., [25]). Let $\mathit{cost}(\mathcal{M}^f_\mathcal{T}) = \inf\{\mathit{cost}(f) : f \text{ is a positional strategy for } \mathcal{M}_\mathcal{T} \text{ that avoids } v_\bot\}$. That is, $\mathit{cost}(\mathcal{M}^f_\mathcal{T})$ is the expected cost of a game played on $\mathcal{M}^f_\mathcal{T}$ in which Classifier uses an optimal policy that avoids $v_\bot$. By [7], $\mathit{cost}(\mathcal{M})$ the restriction to positional strategies does not increase the cost, and $\mathit{cost}(\mathcal{M}^f_\mathcal{T})$ can be computed in polynomial time.

Since $\mathit{legit}(\mathcal{T})$ and hence $\mathcal{M}_\mathcal{T}$ is of size exponential in $\mathcal{T}$, upper bounds to the problem and to the size of the classifier follow:

▶ **Theorem 16.** *The problem of finding a classifier with minimal sensing can be solved in EXPTIME. Moreover, for a specification with n states, such a classifier needs at most 2^n states.*

Theorem 15 provides a matching lower bound for the size of the classifier. For the decision problem, we are able to show a PSPACE lower bound.

▶ **Theorem 17.** *Given a specification $\mathcal{T}$ and a number $\gamma > 0$, it is PSPACE-hard to decide whether there is a classifier for $\mathcal{T}$ with sensing cost strictly smaller than γ.*

Proof. We describe a reduction from the universality problem for NFAs. That is, given an NFA $\mathcal{N}$ over an alphabet Σ, we describe $\gamma \geq 1$ and a specification $\mathcal{T}$ of two languages L_1 and L_2, such that $\mathcal{T}$ can be constructed from $\mathcal{N}$ in polynomial time, and $L(\mathcal{N}) \neq \Sigma^*$ iff there is a classifier for $\mathcal{T}$ with sensing strictly smaller than γ.

Since the universality problem is PSPACE-hard already for NFAs with a single initial state, no empty states, and a nondeterminism branching degree exactly 2 [22], we assume that $\mathcal{N} = \langle Q, \Sigma, \delta, q_0, \alpha\rangle$ is such that no state in $\mathcal{N}$ is empty and the transition function δ gets as a parameter also an index that resolves the nondeterminism. Thus, as in the proof of Theorem 13, we assume that $\delta : Q \times \Sigma \times \{b, \bar{b}\} \to Q$, where b is a signal encoding the nondeterministic choices of $\mathcal{N}$.

Before describing the languages L_1 and L_2, let us describe the set P of signals that defines their alphabet. Let P_Σ be such that $\Sigma = 2^{P_\Sigma}$. We define $P = P_\Sigma \cup \{b, r_1, r_2, s\}$, with the following intuition. Words in $(2^P)^*$ correspond to words over Σ, extended with information about resolving of nondeterminism (the signal b), reseting of the run (the signals r_1 and r_2), and choosing between L_1 and L_2 (the signal s).

Let $P_{base} = P_\Sigma \cup \{r_1, r_2\}$. Also, we say that an assignment a is a *reset assignment* if at least one of the reset signals (r_1 or r_2) are assigned `True`. Otherwise, a is a *simulation assignment*.

For a word $w \in (2^P)^*$, let $v = a_1 \ldots a_k$ be the maximal suffix of w consisting of simulation assignments only. For $1 \leq j \leq k$, let $(\sigma^j, b^j, r^j_1, r^j_2, s^j)$ be the projection of a_j on P_Σ, b, r_1, r_2, and s, respectively. Let $\mathsf{word}(w) = \sigma^1 \cdots \sigma^k \in \Sigma^*$, and let $\mathsf{run}(w) \in Q^*$ be the run of

$\mathcal{N}$ on $\mathsf{word}(w)$ when it resolves its nondeterministic choices according to the assignments to the signal b. Formally, $\mathsf{run}(w) = q_0, q_1, \ldots, q_k$, where for all $1 \leq j \leq k$, we have that $q_j = \delta(q_{j-1}, \sigma^j, b^j)$. Finally, let $\mathsf{last}(w) = q_k$ be the last state in $\mathsf{run}(w)$, namely the state that $\mathcal{N}$ reaches on its run on $\mathsf{word}(w)$ when it resolves its nondeterministic choices according to the assignments to the signal b.

The languages $L_1, L_2 \subseteq (2^P)^+$ are languages of non-empty words defined as follows. Consider a word of the form $w \cdot a$, for $w \in (2^P)^*$ and $a \in 2^P$.

- If a is a reset assignment, then $w \cdot a \in L_1 \cup L_2$, and classification depends on the parity of the signals in P_{base} to which a assigns `True`: If $|a \cap P_{base}|$ is odd, then $w \cdot a \in L_1$, and if $|a \cap P_{base}|$ is even, then $w \cdot a \in L_2$.
- If a is a simulation assignment, then $w \cdot a \in L_1 \cup L_2$ iff $\mathsf{last}(w) \in \alpha$, and classification depends on the assignment to b in a : If $b \in a$, then $w \cdot a \in L_1$, and if $b \notin a$, then $w \cdot a \in L_2$.

Note that in the first case, classification is independent of w, whereas in the second case, the condition about membership in α is independent of a.

We first describe a $(2^P/\{0,1,2\})$-transducer $\mathcal{T}$ that defines the languages L_1 and L_2 and is polynomial in $\mathcal{N}$. We define $\mathcal{T} = \langle 2^P, \{0,1,2\}, Q', q_0', \delta', \tau \rangle$ as follows. The state space $Q' = Q \times \{0,1,2\}$ consists of three copies of Q, and $\mathcal{T}$ starts in the state $q_0' = \langle q_0, 0 \rangle$. Intuitively, $\mathcal{T}$ simulates runs of $\mathcal{N}$ on words over 2^{P_Σ}, using the information from the other signals in order to reset the runs, resolve nondeterminism, and choose among Copies 1 and 2, which is needed whenever the next assignment is a reset assignment, or when the next assignment is a simulation assignment and the current state is accepting. Formally, for a state $\langle q, c \rangle \in Q'$ and letter $a \in 2^P$, we define

$$\delta'(\langle q, c \rangle, a) = \begin{cases} \langle q_0, 1 \rangle & \text{if } a \text{ is a reset assignment and } |a \cap P_{base}| \text{ is odd} \\ \langle q_0, 2 \rangle & \text{if } a \text{ is a reset assignment and } |a \cap P_{base}| \text{ is even} \\ \langle \delta(q, \sigma, i), 0 \rangle & \text{if } a \text{ is a simulation assignment and } q \notin \alpha \\ \langle \delta(q, \sigma, i), 1 \rangle & \text{if } a \text{ is a simulation assignment } q \in \alpha, \text{ and } b \in a \\ \langle \delta(q, \sigma, i), 2 \rangle & \text{if } a \text{ is a simulation assignment } q \in \alpha, \text{ and } b \notin a \end{cases}$$

Finally, $\tau(\langle q, c \rangle) = c$.

It is not hard to see that $\mathcal{T}$ indeed defines L_1 and L_2. Let $\mathcal{M}_\mathcal{T}$ be the 1.5-player game constructed from $\mathcal{T}$. In the full version, we prove that $\mathcal{N}$ is universal iff $\mathit{sense}(\mathcal{M}_\mathcal{T}) \geq |P_{base}| + 1$. The idea of the proof is as follows. First, we consider a classifier $\mathcal{C}$ that senses all signals in P_{base} at all times, never senses b, and senses s only when word(w) is in $L(\mathcal{N})$. The classifier $\mathcal{C}$ is correct, as if the value of s is relevant after reading a word w, then word$(w) \in L(\mathcal{N})$. Note that $\mathcal{C}$ has sensing cost of exactly $|P_{base}| + 1$ when $\mathcal{N}$ is universal, and strictly smaller than $|P_{base}| + 1$ when $\mathcal{N}$ is not universal. We then prove that $\mathcal{C}$ is optimal. This part involves a careful analysis of $\mathit{sense}(\mathcal{M}_\mathcal{T})$, and essentially follows from the fact that sensing b (aiming to reduce the uncertainty due to nondeterminism) is never beneficial: the savings from knowing which nondeterministic choice is made are outweighed by the cost of sensing b itself. Thus, $\mathcal{N}$ is non-universal iff a classifier can achieve sensing cost strictly below $|P_{base}| + 1$. ◀

7 Discussion

We introduced and studied *classification under uncertainty*, where we seek classifiers that satisfy a given language specification over an alphabet 2^P despite uncertainty about the values of some signals in P.

We studied three sources of uncertainty and showed that they all may require a classifier for a specification $\mathcal{T}$ to be of size exponential in $|\mathcal{T}|$. This exponential blow-up is similar to the one exhibited in other games with uncertainty [26]. In particular, in *synthesis with privacy* [18], one seeks transducers that satisfy a given specification while keeping the truth value of given secrets unknown, and in *synthesis with minimal sensing* [1], one seeks transducers that satisfy a given specification with minimal sensing of input signals. Both problems are EXPTIME-complete already for specifications given by deterministic automata. Since correct classification can be specified by a deterministic automaton, the problems are strongly related, and the game $\mathcal{G}_{\mathcal{T}}$ described in Section 3 is indeed similar to the game-theoretic approach used in [1, 18].

Classification and synthesis, however, have a conceptual difference, and the tight complexity of classification with privacy and with minimal sensing is unfortunately left open. Moreover, we found the lower bounds in Theorems 8, 15, and 17, much more complicated than these used in the synthesis case. In order to see the challenge in classification, recall (Theorem 7) that deciding whether $\mathcal{T}$ can be satisfied by a classifier that respects a given sensing constraint $\mathcal{U}$ can be done in NLOGSPACE. This is not the case for synthesis: deciding whether a specification given by a deterministic automaton $\mathcal{A}$ can be realized by a transducer that respect a given sensing constraint $\mathcal{U}$ is EXPTIME-hard [2]. The challenge in lower bounds for classification problems has to do with the fact that in classification, bad strategies of Classifier in the game $\mathcal{G}_{\mathcal{T}}$ have a relatively immediate effect, which can be detected by following two (rather than all) futures of the game. Thus, we still have an element of alternation, but it is weaker than the one that takes us from NLOGSPACE to PTIME in reachability [16]. When $\mathcal{U}$ is not given, we cannot avoid exploring all the subsets of the state space of $\mathcal{T}$, hence the PSPACE lower bound, but the acrobatics required in order to generate instances that generate all these subsets is much higher, and the amount of alternation that is needed in for their exploration is still open, inducing the gap between PSPACE and EXPTIME.

References

1 S. Almagor, D. Kuperberg, and O. Kupferman. Sensing as a complexity measure. *Int. J. Found. Comput. Sci.*, 30(6-7):831–873, 2019. doi:10.1142/S0129054119400203.

2 K. Chatterjee and L. Doyen. The complexity of partial-observation parity games. In *Proc. 16th Int. Conf. on Logic for Programming Artificial Intelligence and Reasoning*, pages 1–14. Springer, 2010. doi:10.1007/978-3-642-16242-8_1.

3 A. Cimatti, C. Tian, and S. Tonetta. Assumption-based runtime verification with partial observability and resets. In *19th International Conference on Runtime Verification*, Lecture Notes in Computer Science, pages 165–184. Springer, 2019. doi:10.1007/978-3-030-32079-9_10.

4 G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3), 2012. doi:10.1145/2187671.2187677.

5 W. Czerwinski, S. Lasota, R. Meyer, S. Muskalla, K.N. Kumar, and P. Saivasan. Regular separability of well-structured transition systems. In *Proc. 29th Int. Conf. on Concurrency Theory*, volume 118 of *LIPIcs*, pages 35:1–35:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CONCUR.2018.35.

6 W. Czerwinski, W. Martens, and T. Masopust. Efficient separability of regular languages by subsequences and suffixes. In *Proc. 40th Int. Colloq. on Automata, Languages, and Programming*, volume 7966 of *Lecture Notes in Computer Science*, pages 150–161. Springer, 2013. doi:10.1007/978-3-642-39212-2_16.

7 L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

8 S. Dziembowski, M. Jurdzinski, and I. Walukiewicz. How much memory is needed to win infinite games. In *Proc. 12th ACM/IEEE Symp. on Logic in Computer Science*, pages 99–110, 1997.

9 A. Ferrando and V. Malvone. Runtime verification via rational monitor with imperfect information. *ACM Trans. on Software Engineering and Methodology*, 35(3):74:1–74:29, 2026. doi:10.1145/3735130.

10 B. Finkbeiner. Logics and algorithms for hyperproperties. *SIGLOG News*, 5(1):31–45, 2018.

11 G. Gange, P. Ganty, and P.J. Stuckey. Fixing the state budget: Approximation of regular languages with small dfas. In *15th Int. Symp. on Automated Technology for Verification and Analysis*, volume 10482 of *Lecture Notes in Computer Science*, pages 67–83. Springer, 2017. doi:10.1007/978-3-319-68167-2_5.

12 E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978. doi:10.1016/S0019-9958(78)90562-4.

13 A. E. Goodloe and K. Havelund. High-integrity runtime verification. *Computer*, 57(4):37–45, 2024. doi:10.1109/MC.2023.3322902.

14 C. Grinstead and J.L. Snell. Chapter 11:Markov chains. In *Introduction to Probability*. American Mathematical Society, 1997.

15 M. Heule and S. Verwer. Exact DFA identification using SAT solvers. In *Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium*, volume 6339 of *Lecture Notes in Computer Science*, pages 66–79. Springer, 2010. doi:10.1007/978-3-642-15488-1_7.

16 N. Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and Systems Science*, 22(3):384–406, 1981. doi:10.1016/0022-0000(81)90039-8.

17 O. Kupferman, N. Lavee, and S. Sickert. Certifying DFA bounds for recognition and separation. *Innov. Syst. Softw. Eng.*, 18(3):405–416, 2022. doi:10.1007/S11334-022-00446-6.

18 O. Kupferman and O. Leshkowitz. Synthesis of privacy-preserving systems. In *Proc. 42nd Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 250 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:23, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.42.

19 O. Kupferman, O. Leshkowitz, and N. Shamash Halevy. Synthesis with privacy against an observer. *Logical Methods in Computer Science*, 21(3), 2025. doi:10.46298/LMCS-21(3:12)2025.

20 M. Leucker, C. Sánchez, T. Scheffel, M. Schmitz, and D. Thoma. Runtime verification for timed event streams with partial information. In *19th International Conference on Runtime Verification*, Lecture Notes in Computer Science, pages 273–291. Springer, 2019. doi:10.1007/978-3-030-32079-9_16.

21 M. Leucker and C. Schallhart. A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009. doi:10.1016/J.JLAP.2008.08.004.

22 A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. 13th IEEE Symp. on Switching and Automata Theory*, pages 125–129, 1972.

23 T. Place and M. Zeitoun. Separating regular languages with first-order logic. *Log. Methods Comput. Sci.*, 12(1), 2016. doi:10.2168/LMCS-12(1:5)2016.

24 A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981. doi:10.1016/0304-3975(81)90110-9.

25 M.L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

26 J.H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and Systems Science*, 29:274–301, 1984. doi:10.1016/0022-0000(84)90034-5.

27 B.A. Trakhtenbrot and Y.M. Barzdin. *Finite Automata*. North Holland, 1973.

28 M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994. doi:10.1006/INCO.1994.1092.

A Calculating the Sensing Cost of a Transducer

Consider a $(2^P/\Sigma_O)$-transducer $\mathcal{T} = \langle 2^P, \Sigma_O, Q, q_0, \delta, \tau\rangle$. Recall that for a finite word $w \in (2^P)^*$, we have that $\mathit{sense}(w)$ is the sensing cost of the run of $\mathcal{T}$ on w, and $\mathit{sense}(\mathcal{T})$ is the expected sensing cost of words of length that tends to infinity.

Thus,

$$\mathit{sense}(\mathcal{T}) = \lim_{m\to\infty} |\Sigma|^{-m} \sum_{w:|w|=m} \mathit{sense}(w).$$

A-priori, it is not clear that $\mathit{sense}(\mathcal{T})$ is well defined, i.e., that the limit above always exists. As detailed below, the limit does always exist, and can be found by analyzing the limiting distribution of a random walk on $\mathcal{T}$.

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.

A transducer $\mathcal{T} = \langle 2^P, \Sigma_O, Q, q_0, \delta, \tau\rangle$ induces a directed graph $\mathcal{G}_\mathcal{T} = \langle Q, E\rangle$ in which $\langle q, q'\rangle \in E$ iff there is a letter $\sigma \in 2^P$ such that $q' = \delta(q, \sigma)$. When we talk about the SCCs of $\mathcal{T}$, we refer to those of $\mathcal{G}_\mathcal{T}$.

Let $\mathcal{E}$ be the set of ergodic SCCs of $\mathcal{T}$. Consider an ergodic SCC $C \in \mathcal{E}$. 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 in C, and the value in coordinate q, q' is $\mathrm{P}_{q,q'}$. By [14], 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 $\pi_C(q) = \lim_{m\to\infty} \frac{O^C_m(q)}{m}$ almost surely, where $O^C_m(q)$ is the number of occurrences of q in a run of $M_\mathcal{T}$ of length m that starts from some arbitrary state in C [14]. Thus, intuitively, $\pi_C(q)$ is the proportion of time that a long run that begins in C spends in q. Note that π_C is the unique stationary distribution of C. In order to extend the distribution to the entire Markov chain of $\mathcal{T}$, we have to take into account the probability of reaching each of the ergodic components. The *SCC-reachability distribution* of $\mathcal{T}$ is the function $\rho : \mathcal{E} \to [0,1]$ that maps each ergodic SCC C of $\mathcal{T}$ to the probability that $M_\mathcal{T}$ eventually reaches C, starting from the initial state.

The *limiting distribution* $\pi : Q \to [0,1]$ is now defined by $\pi(s) = 0$, if s is transient, and $\pi(s) = \pi_C(s) \cdot \rho(C)$, if s is in some $C \in \mathcal{E}$. By [14], the limiting distribution can be computed in polynomial time by solving a system of linear equations. Intuitively, the limiting distribution at state q describes the expected proportion of time that a long run of $M_\mathcal{T}$ that starts in q_0 spends in q. Formally, by the Strong Law of Large Numbers for Markov chains, we have that $\pi(s) = \lim_{m\to\infty} \frac{\mathbb{E}[O_m(s)]}{m}$, where $O_m(s)$ be the number of occurrences of s in a run of length m of $M_\mathcal{T}$ that starts in s_0. Note that when the Markov chain has a single ergodic component, then the limiting distribution coincides with the stationary distribution of that component. That is, for all $q \in Q$, we have $\pi(q) = \pi_C(q)$ where C is the unique ergodic component of $M_\mathcal{T}$. Moreover, in that case, by the Ergodic Theorem for Markov chains it holds that $\pi(q) = \lim_{m\to\infty} \frac{O_m(q)}{m}$ almost surely, and not only $\pi(q) = \mathbb{E}[\lim_{m\to\infty} \frac{O_m(q)}{m}] = \lim_{m\to\infty} \frac{\mathbb{E}[O_m(q)]}{m}$ in expectation as in the general case.

As proven in [1], the sensing cost of a transducer $\mathcal{T}$ can be expressed also with respect to its limiting distribution, which enables a calculation of $\mathit{sense}(\mathcal{T})$ in polynomial time. Formally, if the limiting distribution of $M_\mathcal{T}$ is $\pi : Q \to [0,1]$, then, $\mathit{sense}(\mathcal{T}) = \sum_{q\in Q} \pi(q) \cdot \mathit{sense}(q)$.