Synthesis of Privacy-Preserving Systems

Orna Kupferman
The Hebrew University of Jerusalem, Israel

Ofer Leshkowitz
The Hebrew University of Jerusalem, Israel

Abstract

Synthesis is the automated construction of a system from its specification. In many cases, we want to maintain the *privacy* of the system and the environment, thus limit the information that they share with each other or with an observer of the interaction. We introduce a framework for synthesis that addresses privacy in a simple yet powerful way. Our method is based on specification formalisms that include an *unknown* truth value. When the system and the environment interact, they may keep the truth values of some input and output signals private, which may cause the satisfaction value of specifications to become unknown. The input to the synthesis problem contains, in addition to the specification φ, also *secrets* $\psi_1, \ldots, \psi_k$. During the interaction, the system directs the environment which input signals should stay private. The system then realizes the specification if in all interactions, the satisfaction value of the specification φ is true, whereas the satisfaction value of the secrets $\psi_1, \ldots, \psi_k$ is unknown. Thus, the specification is satisfied without the secrets being revealed. We describe our framework for specifications and secrets in LTL, and extend the framework also to the multi-valued specification formalism LTL[$\mathcal{F}$], which enables the specification of the *quality* of computations. When both the specification and secrets are in LTL[$\mathcal{F}$], one can trade-off the satisfaction value of the specification with the extent to which the satisfaction values of the secrets are revealed. We show that the complexity of the problem in all settings is 2EXPTIME-complete, thus it is not harder than synthesis with no privacy requirements.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Synthesis, Privacy, LTL, Games

Digital Object Identifier 10.4230/LIPIcs.FST& TCS 2022..

1 Introduction

Synthesis is the automated construction of a system from its specification: Given a linear temporal logic (LTL) formula φ over sets I and O of input and output signals, the goal is to return an I/O*-transducer* that *realizes* φ. At each moment in time, the transducer reads a truth assignment, generated by the environment, to the signals in I, and it generates a truth assignment to the signals in O. Thus, with every sequence of inputs, the transducer associates a sequence of outputs, and it realizes φ if all the computations that are generated by the interaction satisfy φ. Synthesis enables designers to focus on *what* the system should do rather than on *how* it should do it, and has attracted a lot of research and interest [41, 9].

While synthesized systems are correct, there is no guarantee about their quality. This is a real obstacle, as designers will give up manual design only after being convinced that the automatic process replacing it generates systems of comparable quality. An important quality measure is *privacy*: we seek systems that allow the underlying components not to reveal information they prefer to keep private. Unlike quality measures that are based on prioritizing different on-going behaviors, privacy is a global conceptual requirement, and it is not clear how to address the challenge of privacy in existing formulations and algorithms of synthesis. The Computer Science community has adopted the notion of *differential privacy* for formalizing when an algorithm maintains privacy. Essentially, an algorithm is differentially private if by observing its output, we cannot tell if a particular individual's information is

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

used in the computation [20, 22]. An orthogonal related challenge is that of *obfuscation* in system development, where we aim to develop systems whose internal operation is hidden [5, 27].

We introduce a framework for synthesis that addresses privacy in a simple yet powerful way. Our method is based on extending the semantics of the specification formalism to include an *unknown* truth value, denoted "?". Let us first explain the framework when applied to LTL. In our framework, the input and output signals take values from $\{\mathtt{F}, ?, \mathtt{T}\}$, and so the semantics is defined with respect to an infinite *noisy computation* $\kappa \in (\{\mathtt{F}, ?, \mathtt{T}\}^{I \cup O})^\omega$. The satisfaction value of a formula ψ in κ is T if all the computations obtained by "filling" the missing information in κ satisfy ψ, is F if all these computations do not satisfy ψ, and is ? otherwise. Allowing the system and the environment to hide the values of some signals by assigning them ? is a natural way to increase privacy. Indeed, the truth value of these signals remains private. Adjusting the definition of realizability to require the system to satisfy the specification in all possible "fillings" of the missing values is also natural. Indeed, as is the case in other settings with incomplete information, we want the specification to hold no matter what the hidden values are [30].

An important question in the three-valued approach is how to measure the privacy level of the system. Clearly, the more signals are hidden, the more privacy is maintained. Still, an approach that is based on the number or the density of unknown assignments is not satisfactory, as many values are often not interesting, and we need an approach that captures the fact that the system and the environment do not reveal information they care not to reveal. A three-valued semantics for LTL and other temporal logics has already been studied in formal methods, in settings in which information is lost due to abstraction and other methods for coping with the state-explosion problem [10, 39, 18]. In all these methods, an evaluation of a specification to ? is a problem, which one should address by adding information, for example by refinement or revealing of data. The novelty of our approach is that we let the system and environment specify in LTL the behaviors they want to keep secret, and we view an evaluation to ? as a desirable outcome – when it applies to secret behaviors.

More formally, the input to our synthesis problem includes a specification φ and a list of *secrets* $\psi_1, \ldots, \psi_k$, both over the set $I \cup O$ of input and output signals. The output of the synthesis problem is an *I/O-transducer with masking*. Such a transducer outputs, in each state, both a three-valued assignment to the signals in O (that is, some output signals may be assigned ?), and a subset of input signals – these whose truth value should not be revealed in the current state. If in state s, the transducer asks the environment not to reveal the truth value of the signals in $M \subseteq I$, then the transitions from s are independent of the values of the signals in M. Accordingly, the interaction of a transducer $\mathcal{T}$ with an environment produces an infinite noisy computation in $(\{\mathtt{F}, ?, \mathtt{T}\}^{I \cup O})^\omega$. The transducer is correct if for all input sequences $w_I \in (\{\mathtt{F}, \mathtt{T}\}^I)^\omega$, the interaction of $\mathcal{T}$ with an environment that generates w_I result in a noisy computation $\kappa \in (\{\mathtt{F}, ?, \mathtt{T}\}^{I \cup O})^\omega$ such that the satisfaction value of the specification φ in κ is T, and the satisfaction value of all secrets $\psi_1, \ldots, \psi_k$ in κ is ?. In other words, all the computations of $\mathcal{T}$ satisfy φ without revealing the secrets. Note that while in traditional synthesis, the system only decides what the assignment to the signals in O is, here the system decides which signals in $I \cup O$ it masks, and then decides what the assignment to the unmasked signals in O is. Clearly, the more signals the system masks, the easier it is for $\psi_1, \ldots, \psi_k$ to stay secret, and the harder it is for φ to be satisfied.

Recall that our goal of synthesizing systems that preserve privacy is a component in our overarching objective to synthesize systems of high quality. In recent years, researchers have started to address the challenge of synthesis of high-quality systems by extending the

Boolean setting to a multi-valued one, capturing different levels of satisfaction [8, 13, 1, 2]. We consider here the linear temporal logic LTL[$\mathcal{F}$], which extends LTL with an arbitrary set $\mathcal{F}$ of functions over $[0,1]$. The satisfaction value of an LTL[$\mathcal{F}$] formula φ is a value in $[0,1]$, where the higher the satisfaction value is, the higher is the quality in which φ is satisfied [1]. Using the functions in $\mathcal{F}$, a specifier can prioritize different ways of satisfaction. Classical decision problems in the Boolean setting become optimization problems in the quantitative setting. In particular, in the synthesis problem, we seek systems with the highest possible satisfaction value [1, 2].

Adding privacy to the setting, this highest possible satisfaction value for the specification φ should be achieved without revealing the satisfaction value of the secrets. We follow the worst-case approach, where the quality of the synthesized system is the minimal satisfaction value of φ in some interaction, and the satisfaction value of all the secrets should be kept unknown in all interactions. We focus on secrets in LTL, but study also secrets in LTL[$\mathcal{F}$], where we can also trade-off the satisfaction value of φ and the extent to which the satisfaction value of the secrets is revealed. We show that the complexity of the problem in all settings is 2EXPTIME-complete for specifications in LTL and LTL[$\mathcal{F}$], thus it is not harder than synthesis with no privacy requirements.

As an example, consider a system that directs a robot patrolling a warehouse storage. Typical specifications for the system requires it to direct the robot so that it eventually reaches the shelves of requested items, it never runs out of energy, etc. Our algorithm automatically synthesizes a system that not only satisfies the specification, but also decides which parts of the interaction to hide so that the specification is satisfied without revealing secrets that would have been revealed by an observer of the full interaction. Such secrets may be dependencies between customers and shelves visited, locations of battery docking stations, and other properties of the structure of the warehouse. As a more specific example, assume there is a set of shelves $S = \{s_1, s_2, \ldots, s_k\}$ such that we want to keep private the vicinity of shelves in S to docking stations. The input signals, namely these assigned by the robot, include the signals at_s_i, for $1 \leq i \leq k$, indicating the robot is at shelf s_i, and the signal *charging*, indicating the robot is at a docking station. Let $at\_S = \bigvee_{i \in [k]} at\_s_i$. Then, adding a secret $\mathsf{F}(\mathit{charging} \wedge at\_S)$ requires the system to direct the robot to hide the values of signals in a way that hides from an observer the truth value of "eventually, the robot is near both some shelf in S and a docking station". Similarly, the secret $\mathsf{F}(\mathit{charging} \wedge at\_s_i)$ hides whether shelf s_i is near a docking station (recall that our framework supports a set of secrets, in particular a secret for each shelf in S). If we need to keep the whole radius of the charging docks secret, we can strengthen the secrets to $\mathsf{F}(\mathsf{X}\mathit{charging} \wedge (at\_S \vee \mathsf{X}at\_S \vee \mathsf{XX}at\_S))$, and similarly for the individual shelves. In order to prevent this secret from being evaluated to $\mathtt{T}$, the system needs to direct the robot to assign ? to at_s_i not only when it assigns $\mathtt{T}$ to *charging*, but also in three time units around it, namely one time unit before and after making *charging* visible. Alternatively, the system can direct the robot to assign ? to *charging*. In addition, since the secret is evaluated to $\mathtt{F}$ in computations in which $at\_s_i \wedge \mathit{charging}$ is always evaluated to $\mathtt{F}$, the system needs to direct the robot to assign ? to at_s_i and *charging* in a way that prevents such an evaluation, for example by assigning them both ? in the initial state. In general, the choice of the system which signals to hide depends on other specifications it has to satisfy. If, for example, it is essential for the system to know about visits to all the shelves in S, then it may direct the robot not to charge near them or to hide the fact it does so. Otherwise, the system may leave the information about the visits unknown, and it may also combine the two solutions – this is exactly what our procedure does automatically.

One technical challenge of our algorithms is the need to combine automata for the specification with automata for the secrets. For the specification φ, the quantification of the hidden information is universal – we want all computations obtained by filling the hidden information to satisfy φ. For a secret ψ_i, the quantification of the hidden information is existential – we want witnesses that different fillings lead to different satisfaction values of ψ_i. The fact we need automata that handle both types of quantification makes it impossible to proceed with a *Safraless* synthesis algorithm, which requires universal automata [31]. We introduce and study a *syntax-based three-valued semantics* for LTL in noisy computations, which enables us to construct universal automata for secrets, leading to a Safraless synthesis algorithm that circumvents determinization and solution of parity games.

Related work A very basic model of privacy has been studied in the context of synthesis with *incomplete information* [35, 30, 14], where the value of a subset of the signals stays secret throughout the interaction. Synthesis with incomplete information can be viewed as a special case of our approach here. Indeed, hiding of a signal p can be achieved with the secrets $\mathsf{F}p$ and $\mathsf{F}\neg p$. Moreover, our framework supports hiding of designated signals in parts (rather than all) of the interaction.

Lifting differential privacy to formal methods, researchers have introduced the temporal logic *HyperLTL*, which extends LTL with explicit trace quantification [17]. In particular, such a quantification can relate computations that differ only in non-observable elements, and can be used for specifying that computations with the same observable input have the same observable output. The synthesis problem of HyperLTL is undecidable, yet is decidable for the fragment with a single existential quantifier, which can specify interesting properties [24]. Our approach here is different, as it enables the specification of arbitrary secrets, and can be implemented on top of LTL synthesis tools.

As for obfuscation, while it is mainly studied in the context of software, where it has exciting connections with cryptography [5, 27], researchers have also studied the synthesis of obfuscation policies for temporal specifications [21, 43], which is closer to our approach here. In [43], an obfuscation mechanism is based on edit functions that alter the output of the system, aiming to make it impossible for an observer to distinguish between secret and non-secret behaviors. In [21], the goal is to synthesize a control function that directs the user which actions to disable, so that the observed sequence of actions would not disclose a secret behavior. Our work, on the other hand, addresses the general synthesis problem and thus handles specifications and secrets that are on-going infinite behaviors given by LTL and LTL[$\mathcal{F}$] specifications. In particular, while our transducers can mask information, they do not have an option to edit the interaction or disable actions of the environment.

2 Preliminaries

2.1 Automata

For a finite nonempty alphabet Σ, an infinite *word* $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of letters from Σ. A *language* $L \subseteq \Sigma^\omega$ is a set of infinite words.

An *automaton* over infinite words is $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where Σ is an alphabet, 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 α is an *acceptance condition*, to be defined below. For states $q, s \in Q$ and a letter $\sigma \in \Sigma$, we say that s is a σ-successor of q if $s \in \delta(q, \sigma)$. We consider automata with a total transition function. That is, for every state $q \in Q$ and letter $\sigma \in \Sigma$, we have that $|\delta(q, \sigma)| \geq 1$. If $|\delta(q, \sigma)| = 1$ for every state $q \in Q$ and letter $\sigma \in \Sigma$, then $\mathcal{A}$ is *deterministic*.

A *run* of $\mathcal{A}$ on $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of states $r = r_0, r_1, r_2, \ldots \in Q^\omega$, such that $r_0 = q_0$, and for all $i \geq 0$, we have that $r_{i+1} \in \delta(r_i, \sigma_{i+1})$. The acceptance condition α determines which runs are "good". We consider here the *Büchi*, *co-Büchi*, *generalized Büchi*, *generalized co-Büchi*, and *parity* acceptance conditions. All conditions refer to the set $inf(r) \subseteq Q$ of states that r traverses infinitely often. Formally, $inf(r) = \{q \in Q : q = r_i$ for infinitely many i's$\}$. In generalized Büchi and co-Büchi automata, the acceptance condition is of the form $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, for sets $\alpha_i \subseteq Q$. In a generalized Büchi automaton, a run r is accepting if for all $1 \leq i \leq k$, we have that $inf(r) \cap \alpha_i \neq \emptyset$. Thus, r visits each of the sets in α infinitely often. Dually, in a generalized co-Büchi automaton, a run r is accepting if there exists $1 \leq i \leq k$ such that $inf(r) \cap \alpha_i = \emptyset$. Thus, r visits at least one of the sets in α only finitely often. Büchi and co-Büchi automata are special cases, with $k = 1$, of their generalized forms. Finally, in a parity automaton $\alpha : Q \to \{1, ..., k\}$ maps states to ranks, and a run r is accepting if the maximal rank of a state in $inf(r)$ is even. Formally, $\max_{q \in inf(r)}\{\alpha(q)\}$ is even. A run that is not accepting is *rejecting*. We refer to the number k in α (that is, the number of sets in the generalized conditions and the number of ranks in parity conditions) as the *index* of the automaton.

Note that as $\mathcal{A}$ may not be deterministic, it may have several runs on a word. We distinguish between two branching modes. If $\mathcal{A}$ is a *nondeterministic* automaton, then a word w is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on w. If $\mathcal{A}$ is a *universal* automaton, then a word w is accepted by $\mathcal{A}$ if all the runs of $\mathcal{A}$ on w are accepting. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. Two automata are *equivalent* if their languages are equivalent.

We denote the different classes of automata by three-letter acronyms in {D,N,U} × {B,C,GB,GC,P} × {W,T}. The first letter stands for the branching mode of the automaton (deterministic, nondeterministic, or universal); the second for the acceptance condition type (Büchi, co-Büchi, generalized Büchi, generalized co-Büchi, or parity); and the third indicates we consider automata on words or trees (in Appendix A, we define tree automata). For example, NBWs are nondeterministic Büchi word automata.

2.2 Parity games

A parity game is $\mathcal{G} = \langle V, E, v_0, \alpha \rangle$ is played between two players SYS and ENV. The set V of positions is partitioned into two disjoint sets $V = V_{\text{ENV}} \cup V_{\text{SYS}}$, controlled by SYS and ENV. Then, $E \subseteq (V_{\text{SYS}} \times V_{\text{ENV}}) \cup (V_{\text{ENV}} \times V_{\text{SYS}})$ is a transition relation, which we assume to alternate between $V_s Sys$ and $V_s env$, $v_0 \in V$ is an initial position, and $\alpha : V \to \{1, \ldots, k\}$ is a parity winning condition.

A strategy $f_{Sys} : V^* \cdot V_{\text{SYS}} \to V_{\text{ENV}}$ for Sys maps a finite path in $\mathcal{G}$ that ends in a position $u \in V_{\text{SYS}}$ to a next position $v \in V_{\text{ENV}}$ such that $(u, v) \in E$, and similarly for a strategy $f_{Env} : V^* \cdot V_{\text{ENV}} \to V_{\text{SYS}}$ for ENV. When the two players play according to their strategies f_{Sys} and f_{Env}, the *outcome* of the game, denoted outcome($f_{\text{SYS}}, f_{\text{ENV}}$), is the unique infinite path $\rho = v_0, u_0, v_1, u_1, \ldots \in (V_{\text{SYS}} \cdot V_{\text{ENV}})^\omega$, where $v_0 \in V_{\text{SYS}}$ is the initial position, and for all $j \geq 0$, we have that $u_j = f_{Sys}(v_0, u_0, \ldots, v_j)$ and $v_j = f_{Env}(v_0, u_0, \ldots, v_{j-1}, u_{j-1})$.

A strategy f_{Sys} of SYS is *winning* if for every strategy f_{ENV} for ENV from v_0, we have that outcome($f_{\text{SYS}}, f_{\text{ENV}}$) satisfies the winning condition α. We say that SYS wins $\mathcal{G}$, if it has a winning strategy.

2.3 The temporal logic LTL[$\mathcal{F}$]

The logic LTL[$\mathcal{F}$] is a multi-valued logic that extends the linear temporal logic LTL with an arbitrary set of functions $\mathcal{F} \subseteq \{f : [0,1]^k \to [0,1] : k \in \mathbb{N}\}$ called quality operators. For example, $\mathcal{F}$ may contain the maximum or minimum between the satisfaction values of subformulas, their product, and their average. This enables the specifier to refine the Boolean correctness notion and associate different possible ways of satisfaction with different truth values [1].

Let AP be a finite set of Boolean atomic propositions. The syntax of LTL[$\mathcal{F}$] is given by the following grammar, where the symbol $\texttt{T}$ stands for True, p ranges over a set AP of atomic propositions, $\varphi_1, \varphi_2, \ldots, \varphi_k$ are LTL[$\mathcal{F}$] formulas and $f : [0,1]^k \to [0,1] \in \mathcal{F}$.

$$\varphi := \texttt{T} \mid p \mid f(\varphi_1, \varphi_2, \ldots, \varphi_k) \mid \mathsf{X}\varphi_1 \mid \varphi_1 \mathsf{U} \varphi_2.$$

The *length* of φ, denoted $|\varphi|$, is the number of nodes in the generating tree of φ. Note that $|\varphi|$ bounds the number of sub-formulas of φ. The semantics of LTL[$\mathcal{F}$] is defined with respect to *computations* over AP. Let the calligraphic digit $\mathit{2}$ denote the set $\{\texttt{F}, \texttt{T}\}$, where $\texttt{F}$ stands for False and $\texttt{T}$ stands for True. Given a computation $\pi = \pi_0, \pi_1, \ldots \in (\mathit{2}^{AP})^\omega$ and a position $j \geq 0$, we use π^j to denote the suffix $\pi_j, \pi_{j+1}, \ldots \in (\mathit{2}^{AP})^\omega$ of π. The semantics maps a computation $\pi \in (\mathit{2}^{AP})^\omega$ and an LTL[$\mathcal{F}$] formula φ to the *satisfaction value of* φ *in* π, denoted $[\![\pi, \varphi]\!]$. The satisfaction value is in $[0, 1]$, and is defined inductively as follows.

- $[\![\pi, \texttt{T}]\!] = 1$.
- $[\![\pi, p]\!] = 1$ if $p \in \pi_0$ and $[\![\pi, p]\!] = 0$ if $p \notin \pi_0$.
- $[\![\pi, f(\varphi_1, \varphi_2, \ldots, \varphi_k)]\!] = f([\![\pi, \varphi_1]\!], [\![\pi, \varphi_2]\!], \ldots, [\![\pi, \varphi_k]\!])$.
- $[\![\pi, \mathsf{X}\varphi_1]\!] = [\![\pi^1, \varphi_1]\!]$.
- $[\![\pi, \varphi_1 \mathsf{U} \varphi_2]\!] = \max_{i \geq 0} \{\min([\![\pi^i, \varphi_2]\!], \min_{0 \leq j < i} [\![\pi^j, \varphi_1]\!])\}$.

The logic LTL coincides with the logic LTL[$\mathcal{F}$] for $\mathcal{F} = \{\neg, \vee, \wedge\}$, which corresponds to the usual Boolean operators. Formally, for $x, y \in [0, 1]$, we have $\neg x = 1 - x$, $x \vee y = \max\{x, y\}$, and $x \wedge y = \min\{x, y\}$. To see that LTL indeed coincides with LTL[$\mathcal{F}$], note that for this $\mathcal{F}$, all formulas are mapped to $\{0, 1\}$ in a way that agrees with the semantics of LTL. When φ is an LTL formula, we say that a computation π *satisfies* φ, denoted $\pi \models \varphi$, iff $[\![\pi, \varphi]\!] = 1$.

The novelty of LTL[$\mathcal{F}$] is the ability to manipulate values by arbitrary functions. For example, $\mathcal{F}$ may contain the quantitative operator $\triangledown_\lambda$, for $\lambda \in [0, 1]$, which tunes down the quality of a sub-specification. Formally, $[\![\pi, \triangledown_\lambda \varphi_1]\!] = \lambda \cdot [\![\pi, \varphi_1]\!]$. Another useful operator is the weighted-average function $\oplus_\lambda$ that is defined, for $\lambda \in [0, 1]$, by $[\![\pi, \varphi_1 \oplus_\lambda \varphi_2]\!] = \lambda \cdot [\![\pi, \varphi_1]\!] + (1 - \lambda) \cdot [\![\pi, \varphi_2]\!]$. Consider, for example, the robot at the warehouse example from Section 1. Suppose shelf s_1 is at the east of the warehouse and we prefer the robot to be as close to the center as possible. Accordingly, we want a specification that incentivize the system to direct the robot in s_1 to the west, possibly also to the north or south, but less to the east. This can be done with the LTL[$\mathcal{F}$] specification $\psi_1 = \mathsf{G}(at\_s_1 \to \mathsf{X}(W \vee \triangledown_{\frac{4}{5}}(N \vee S) \vee \triangledown_{\frac{3}{5}} E)$. Then, the satisfaction value of ψ_1 in computations in which the system directs the robot to go east from s_1 (for example, in order to satisfy other specifications), get satisfaction value $\frac{3}{5}$.

Suppose further that the robot sends a signal *low* whenever its battery falls below some threshold, in which case the system should direct the robot not to pick up new packages and to charge its battery in the first docking station it comes across. Ideally, the robot stays in this docking station for two consecutive time units. This can be stated with the LTL[$\mathcal{F}$] specification $\psi_2 = \mathsf{G}(low \to (\neg pickup \wedge \neg station) U (station \wedge (charging \oplus_{\frac{2}{3}} \mathsf{X} charging)))$. When the robot indeed stops at the first docking station and charges for two time units, the satisfaction value is $\frac{2}{3} + \frac{1}{3} = 1$. If it stays there for only one time unit, the satisfaction value is $\frac{2}{3}$, and if it starts the charging only at the second time unit in the station, the satisfaction value drops to $\frac{2}{3}$. Note that the satisfaction value of ψ_1 and ψ_2 may not be 1 not only as a result of a non-optimal behavior but also as a result of hiding of an optimal behavior. For example, aiming to hide the secrets discussed in Section 1, the system may direct the robot to assign ? to *charging*, reducing the satisfaction value of ψ_2.

▶ **Theorem 1.** *[1] Let φ be an LTL[$\mathcal{F}$] formula over AP and $P \subseteq [0,1]$ be a predicate. There exists an NGBW $\mathcal{A}_\varphi^P$ over the alphabet 2^{AP} such that for every computation $\pi \in (2^{AP})^\omega$, we have that $\mathcal{A}_\varphi^P$ accepts π iff $[\![\pi, \varphi]\!] \in P$. Furthermore, $\mathcal{A}_\varphi^P$ has at most $2^{O(|\varphi|)}$ states and index at most $|\varphi|$.*

2.4 LTL[$\mathcal{F}$] synthesis

Consider finite disjoint sets I and O of input and output signals, which takes values in $\mathit{2}$. For $i \in \mathit{2}^I$ and $o \in \mathit{2}^O$, let $i \cup o \in \mathit{2}^{I \cup O}$ be the assignment that agrees with i and o. An *I/O-transducer* models an interaction between an environment that generates in each moment in time an input in $\mathit{2}^I$ and a system that responds with an output in $\mathit{2}^O$. Formally, an I/O-transducer is a tuple $\mathcal{T} = \langle I, O, S, s_0, \eta, \tau \rangle$ where S is a finite set of states, $s_0 \in S$ is an initial state, $\eta : S \times \mathit{2}^I \to S$ is a deterministic transition function, and $\tau : S \to \mathit{2}^O$ is an output-labeling function. Given a sequence $w_I = i_0, i_1, i_2, \ldots \in (2^I)^\omega$ of input letters, the *run of $\mathcal{T}$ on w_I* is defined to be the sequence of states $r(w_I) = s_0, s_1, s_2, \ldots \in S^\omega$ that begins with the initial state s_0 and is such that for all $j \geq 0$, we have $s_{j+1} = \eta(s_j, i_j)$. We define the *computation of $\mathcal{T}$ on w_I* to be $\mathcal{T}(w_I) = (i_0 \cup o_0), (i_1 \cup o_1), (i_2 \cup o_2), \ldots \in (2^{I \cup O})^\omega$, where for all $j \geq 0$, we have $o_j = \tau(s_j)$. Note that the interaction is initiated by the system: the j-th output letter is determined by the j-th state, prior of reading the j-th input letter.

Defining the satisfaction value of φ in $\mathcal{T}$, denoted $[\![\mathcal{T}, \varphi]\!]$, the environment is assumed to be hostile and we care for the minimal satisfaction value of some computation of $\mathcal{T}$. Formally, $[\![\mathcal{T}, \varphi]\!] = \min\{[\![\mathcal{T}(w_I), \varphi]\!] : w_I \in (2^I)^\omega\}$. Note that no matter what the input sequence is, the specification φ is satisfied with value at least $[\![\mathcal{T}, \varphi]\!]$.

The *realizability* problem for LTL[$\mathcal{F}$] is to determine, given φ and a predicate $P \subseteq [0,1]$, whether there exists a transducer $\mathcal{T}$ such that $[\![\mathcal{T}, \varphi]\!] \in P$. We then say that $\mathcal{T}$ realizes $\langle \varphi, P \rangle$. The *synthesis* problem is then to generate such a transducer. Of special interest are predicates P that are upward closed. Thus, $P = [v, 1]$ for some $v \in [0,1]$.

2.5 Satisfaction value in noisy computations

Let the calligraphic digit $\mathit{3}$ denote the set $\{\mathtt{F}, \mathtt{T}, ?\}$. We think of $\mathit{3}^{AP}$ as the set of *noisy assignments* to AP, where the truth value of a proposition mapped to ? is “unknown”. For two noisy assignments $\sigma, \sigma' \in \mathit{3}^{AP}$, we say that σ' is *more informative* than σ, denoted $\sigma \leq_{\mathtt{info}} \sigma'$, if for all $p \in AP$, we have that $\sigma(p) \in \{\sigma'(p), ?\}$. Thus, some assignments of F and T in σ' may become ? in σ. A *noisy computation* over AP is an infinite word $\kappa = \kappa_0, \kappa_1, \ldots \in (\mathit{3}^{AP})^\omega$. We extend the $\leq_{\mathtt{info}}$ relation to noisy computations in the expected way, thus for $\kappa, \kappa' \in (\mathit{3}^{AP})^\omega$, we have that $\kappa \leq_{\mathtt{info}} \kappa'$ iff for all $i \geq 0$, we have that $\kappa_i \leq_{\mathtt{info}} \kappa'_i$.

A noisy assignment $\sigma \in \mathit{3}^{AP}$ induces a set $\text{fill}(\sigma) \subseteq \mathit{2}^{AP}$ of assignments, obtained by replacing assignments to ? by assignments to F or T. Formally, an assignment $\sigma' \in \mathit{2}^{AP}$ is in $\text{fill}(\sigma)$ if $\sigma \leq_{\mathtt{info}} \sigma'$. Each noisy computation κ induces a set $\text{fill}(\kappa)$ of computations in $(\mathit{2}^{AP})^\omega$, where $\pi = \pi_0, \pi_1, \ldots$ is in $\text{fill}(\kappa)$ if for all $i \geq 0$, it holds that $\pi_i \in \text{fill}(\kappa_i)$. Note that $\kappa \leq_{\mathtt{info}} \pi$ iff $\pi \in \text{fill}(\kappa)$.

For a noisy computation $\kappa \in (\mathit{3}^{AP})^\omega$ and an LTL[$\mathcal{F}$] formula φ over AP, we denote by $[\![\kappa, \varphi]\!]$ the set of satisfaction values of φ in computations in $\text{fill}(\kappa)$. Formally,

$$[\![\kappa, \varphi]\!] = \{[\![\pi, \varphi]\!] : \pi \in (\mathit{2}^{AP})^\omega \text{ is such that } \pi \in \text{fill}(\kappa)\}.$$

For an LTL formula ψ, we say that κ satisfies ψ, denoted $\kappa \models \psi$, if $\pi \models \psi$ for all computations π in $\mathrm{fill}(\kappa)$. Thus, ψ is satisfied in all the computations obtained by filling κ. Note that for an LTL formula ψ, we have that $[\![\kappa, \psi]\!]$ is $\{0\}$, $\{1\}$, or $\{0, 1\}$. For simplicity, we use T, F, and ? to refer to these cases. In particular, $[\![\kappa, \psi]\!] =?$ if κ can be filled both to a computation that satisfies ψ and to a computation that does not satisfy ψ, and in such case we say that κ *hides* ψ.

3 Problem Formulation

In this section we define the problem of synthesis with privacy. We first define *noisy I/O-transducers*, which are the output of the synthesis algorithm.

3.1 Noisy transducers

A *noisy I/O-transducer* is $\mathcal{T} = \langle I, O, S, s_0, \eta, \tau, \mathfrak{m}\rangle$, which augments an I/O-transducer by an *input-masking function* $\mathfrak{m} : S \to 2^I$. In addition, the transition function assumes a noisy assignment to the input signals, thus $\eta : S \times 3^I \to S$, and the labeling function generates a noisy assignment to the output signals, thus $\tau : S \to 3^O$. Intuitively, when the transducer is in state s, it generates the noisy assignment $\tau(s)$ to the output signals and declares that the values of input signals in $\mathfrak{m}(s)$ should stay private. Then, the environment generates an assignment $\sigma \in 2^I$ and reveals only the values of signals not in $\mathfrak{m}(s)$. Thus, the transducer moves to the successor state $s' = \eta(s, \sigma')$, where $\sigma' \in 3^I$ is obtained from σ by assigning ? to the signals in $\mathfrak{m}(s)$.

Formally, for an input assignment $i \in 2^I$ and a subset $M \in 2^I$ of I, let $hide(M, i) \in 3^I$ be the noisy input assignment such that for every $p \in I$, if $p \in M$, then $hide(M, i)(p) =?$, and if $p \notin M$, then $hide(M, i)(p) = i(p)$. Given an infinite sequence of assignments to the input signals $w_I = i_0, i_1, i_2, \ldots \in (2^I)^\omega$, we define the *run* of $\mathcal{T}$ on w_I and the *observable input sequence* induced by w_I, as the sequences $r(w_I) = s_0, s_1, s_2, \ldots \in S^\omega$ and $w'_I = i'_0, i'_1, i'_2, \ldots \in (3^I)^\omega$, respectively, where for all $j \geq 0$, we have that $i'_j = hide(\mathfrak{m}(s_j), i_j)$ and $s_{j+1} = \eta(s_j, i'_i)$.

For a noisy input assignment $i' \in 3^I$ and a noisy output assignment $o' \in 3^O$, we define $i' \cup o' \in 3^{I \cup O}$ as the noisy assignment that agrees with i' and o'. The *noisy computation* of $\mathcal{T}$ on w_I is then $\mathcal{T}_\mathfrak{m}(w_I) = (i'_0 \cup \tau(s_0)), (i'_1 \cup \tau(s_1)), (i'_2 \cup \tau(s_2)), \ldots \in (3^{I \cup O})^\omega$.

Note that while each input sequence $w_I \in (2^I)^\omega$ induces a single noisy computation in $(3^{I \cup O})^\omega$, it induces several computations in $(2^{I \cup O})^\omega$. Namely, the set $\mathrm{fill}(\mathcal{T}_\mathfrak{m}(w_I))$ of all computations that are obtained by filling the noisy assignments to the signals that are unknown in $\mathcal{T}_\mathfrak{m}(w_I)$.

3.2 Synthesis with privacy

In *synthesis with privacy*, we are given a specification φ in LTL[$\mathcal{F}$] and a set of secrets $\{\psi_1, \ldots, \psi_k\}$ in LTL, and we seek a noisy I/O-transducer that satisfies φ in the highest specification value while keeping the satisfaction value of $\psi_1, \ldots, \psi_k$ unknown. Formally, a noisy I/O-transducer $\mathcal{T}$ *realizes* $\langle\varphi, P\rangle$ *with privacy* $\psi_1, \ldots, \psi_k$, for a predicate $P \subseteq [0, 1]$, if for every input sequence $w_I \in (2^I)^\omega$, it holds that $[\![\mathcal{T}_\mathfrak{m}(w_I), \varphi]\!] \subseteq P$ and $[\![\mathcal{T}_\mathfrak{m}(w_I), \psi_i]\!] =?$, for all $1 \leq i \leq k$.

Note that we chose to focus on a setting where the secret ψ is an LTL (rather than LTL[$\mathcal{F}$]) formula. This is because the behaviors we want to keep private are typically qualitative. In Section 4.1 we describe how our framework can be extended to secrets in LTL[$\mathcal{F}$].

Note also that while the input to our problem contains a single specification, it contains several secrets. Indeed, while for a set $\{\varphi_1, \ldots, \varphi_k\}$ of specifications, a system realizes their conjunction $\varphi_1 \wedge \cdots \wedge \varphi_k$ iff it realizes all conjuncts φ_i, for a set of secrets $\{\psi_1, \ldots, \psi_k\}$, we cannot guarantee that the system hides all the secrets in the set by defining a single secret that is some Boolean combination of $\psi_1, \ldots, \psi_k$. In particular, an unknown truth value for the conjunction $\psi_1 \wedge \cdots \wedge \psi_k$ does not guarantee an unknown truth value for all conjuncts.

▶ Remark 2. Note that our framework hides the truth values of the secrets from an external observer: rather than observing computations in $(2^{I \cup O})^\omega$, it observes noisy computations in $(3^{I \cup O})^\omega$. If we want to assure the environment that the secrets are hidden also from the system, then we can change the framework so that the labeling function of the transducer generates non-noisy assignments to the output signals, thus $\tau : S \to 2^O$. Then, the result of the interaction is a computation in which only the input signals are noisy, and it should still keep the satisfaction values of the secrets unknown. Dually, if we only care about the privacy of the system, we can give up the noisy assignment to the input signals, which considerably simplifies the setting, as it makes the input-masking function unneccesary. ◀

4 Specifying Secrets

A key component of our algorithms is a construction of automata over an alphabet 3^{AP} that accept noisy computations that hide the satisfaction value of a secret. In this section we define such automata. We start with secrets in LTL. Recall that a noisy computation $\kappa \in (3^{AP})^\omega$ hides an LTL formula ψ if there are two computations $\pi_1, \pi_2 \in \mathrm{fill}(\kappa)$ such that $\pi_1 \models \psi$ and $\pi_2 \not\models \psi$. Note that this implies that an observer of κ indeed does not know whether the computation that induces κ satisfies ψ. We first define an automaton that follows the above definition. Essentially, the automaton is obtained by taking the intersection of two automata, one that accepts a noisy computation $\kappa \in (3^{AP})^\omega$ iff $1 \in [\![\kappa, \psi]\!]$, and one that accepts κ iff $0 \in [\![\kappa, \psi]\!]$.

▶ **Theorem 3.** *Let ψ be an LTL formula over AP. There exists an NGBW $\mathcal{N}_\psi^?$ over the alphabet 3^{AP} such that for every noisy computation $\kappa \in (3^{AP})^\omega$, we have that $\mathcal{N}_\psi^?$ accepts κ iff $[\![\kappa, \psi]\!] = ?$. Also, $\mathcal{N}_\psi^?$ has at most $2^{O(|\psi|)}$ states and index at most $|\psi|$.*

Proof. Let $\mathcal{A}_\psi^1 = \langle 2^{AP}, Q, Q_0, \delta, \alpha \rangle$ be an NGBW such that for every computation $\pi \in (2^{AP})^\omega$, it holds that $\mathcal{A}_\psi^1$ accepts π iff $\pi \models \psi$. Let $\mathcal{N}_\psi^{\mathtt{T}} = \langle 3^{AP}, Q, Q_0, \delta', \alpha \rangle$ be the NGBW obtained from $\mathcal{A}_\psi^1$ by letting it guess an assignment to atomic propositions whose value is unknown. Formally, for every state $q \in Q$ and letter $\sigma' \in 3^{AP}$, we have that $\delta'(q, \sigma') = \bigcup\{\delta(q, \sigma) : \sigma \in 2^{AP}$ is such that $\sigma' \leq_{\mathtt{info}} \sigma\}$. It is easy to see to see that $\mathcal{N}_\psi^{\mathtt{T}}$ accepts a noisy computation $\kappa \in (3^{AP})^\omega$ iff $1 \in [\![\kappa, \psi]\!]$. In a similar way, one can construct an NGBW $\mathcal{N}_\psi^{\mathtt{F}}$ that accepts a noisy computation $\kappa \in (3^{AP})^\omega$ iff $0 \in [\![\kappa, \psi]\!]$. We can now define the required NGBW $\mathcal{N}_\psi^?$ as the intersection of $\mathcal{N}_\psi^{\mathtt{T}}$ and $\mathcal{N}_\psi^{\mathtt{F}}$. By [42], both $\mathcal{N}_\psi^{\mathtt{T}}$ and $\mathcal{N}_\psi^{\mathtt{F}}$ have at most $2^{O(|\psi|)}$ states and index at most $|\psi|$, implying the same bound for $\mathcal{N}_\psi^?$. ◀

A drawback of the construction in Theorem 3 is that the constructed automaton is nondeterministic, which seems unavoidable. Indeed, it guesses values for the unknown signals that lead to satisfaction and violation of ψ. The use of a nondeterministic automaton makes it impossible to proceed with a Safraless synthesis algorithm, which requires universal automata [31]. In order to address this weakness, we define a syntax-based three-valued semantics for LTL formulas when interpreted with respect to noisy computations. As we elaborate in the sequel, the syntax-based semantics coincides with the semantics-based one only for *well-specified* secrets, and it enables us to define universal automata for such secrets.

We start by defining the syntax-based three-valued semantics. We consider LTL formulas with the following syntax.

$$\psi := \mathtt{T} \mid p \mid \neg\psi_1 \mid \psi_1 \vee \psi_2 \mid \mathsf{X}\psi_1 \mid \psi_1 \mathsf{U} \psi_2.$$

Given a noisy computation $\kappa = \kappa_0, \kappa_1, \ldots \in (3^{AP})^\omega$ and a position $j \geq 0$, we use κ^j to denote the suffix $\kappa_j, \kappa_{j+1}, \ldots \in (3^{AP})^\omega$ of κ. The three-valued semantics maps a noisy computation $\kappa \in (3^{AP})^\omega$ and an LTL formula ψ to the *three-valued satisfaction value of* ψ *in* κ, denoted $\langle\langle \kappa, \psi \rangle\rangle$, and defined inductively as follows.

- $\langle\langle \kappa, \mathtt{T} \rangle\rangle = \mathtt{T}$.
- $\langle\langle \kappa, p \rangle\rangle = \kappa_0(p)$.
- $\langle\langle \kappa, \neg\psi_1 \rangle\rangle = \neg\langle\langle \kappa, \psi_1 \rangle\rangle$, where $\neg\mathtt{T} = \mathtt{F}$, $\neg\mathtt{F} = \mathtt{T}$, and $\neg ? = ?$.
- $\langle\langle \kappa, \psi_1 \vee \psi_2 \rangle\rangle = \begin{bmatrix} \mathtt{T} & \text{if } \langle\langle \kappa, \psi_1 \rangle\rangle = \mathtt{T} \text{ or } \langle\langle \kappa, \psi_2 \rangle\rangle = \mathtt{T}, \\ \mathtt{F} & \text{if } \langle\langle \kappa, \psi_1 \rangle\rangle = \mathtt{F} \text{ and } \langle\langle \kappa, \psi_2 \rangle\rangle = \mathtt{F}, \\ ? & \text{otherwise.} \end{bmatrix}$
- $\langle\langle \kappa, \mathsf{X}\psi_1 \rangle\rangle = \langle\langle \kappa^1, \psi_1 \rangle\rangle$.
- $\langle\langle \kappa, \psi_1 \mathsf{U} \psi_2 \rangle\rangle = \begin{bmatrix} \mathtt{T} & \text{if } \exists i \geq 0. \langle\langle \kappa^i, \psi_2 \rangle\rangle = \mathtt{T} \text{ and } \forall 0 \leq j < i, \langle\langle \kappa^j, \psi_1 \rangle\rangle = \mathtt{T}, \\ \mathtt{F} & \text{if } \forall i \geq 0. \langle\langle \kappa^i, \psi_2 \rangle\rangle \neq \mathtt{F} \text{ implies } \exists 0 \leq j < i, \langle\langle \kappa^j, \psi_1 \rangle\rangle = \mathtt{F}. \\ ? & \text{otherwise.} \end{bmatrix}$

As we now show, the classical translation of LTL formulas to NGBWs [42] can be extended to noisy computations. For an LTL formula ψ, let $cl(\psi)$ denote the set of ψ's subformulas and their negation. The state space of our NGBW consists of functions $f \in 3^{cl(\psi)}$ that do not contain propositional inconsistencies. For example, $f(\psi_1 \vee \psi_2) = ?$ iff $f(\psi_1) = ?$ and $f(\psi_2) \in \{?, \mathtt{F}\}$, or $f(\psi_2) = ?$ and $f(\psi_1) \in \{?, \mathtt{F}\}$. Then, the transition function corresponds to temporal requirements, and the acceptance condition makes sure that eventualities are not propagated forever. As is the case with the construction in [42], each noisy computation κ has a single accepting run in the NGBW: the run starts from the state f_0 that describes the satisfaction value of all the formulas in $cl(\psi)$ in κ (according to the syntax-based semantics), continues to the state f_1 that describes the satisfaction in the suffix κ^1, and so on. Accordingly, the choice of initial states determines the language of the NGBW. For obtaining an NGBW for computations κ with $\langle\langle \kappa, \psi \rangle\rangle = ?$, we define the set of initial states to consists of functions f for which $f(\psi) = ?$. For obtaining an equivalent UGCW, we dualize the NGBW whose set of initial state consists of functions f for which $f(\psi) \neq ?$ (see proof in Appendix B.1).

► **Theorem 4.** *Let ψ be an LTL formula over AP. There exist an NGBW $\mathcal{S}^?_\psi$ and a UGCW $\mathcal{U}^?_\psi$ over the alphabet 3^{AP}, such that for every noisy computation $\kappa \in (3^{AP})^\omega$, we have that $\mathcal{S}^?_\psi$ accepts κ iff $\mathcal{U}^?_\psi$ accepts κ iff $\langle\langle \kappa, \psi \rangle\rangle = ?$. Also, $\mathcal{S}^?_\psi$ and $\mathcal{U}^?_\psi$ have at most $2^{O(|\psi|)}$ states and index at most $|\psi|$.*

The syntax-based semantics may not change the polarity of evaluations, yet it may lead to a loss of information. Formally, we have the following, which can be proved by an induction on the structure of the LTL formula.

► **Lemma 5.** *For every noisy computation κ and LTL formula ψ, if $\langle\langle \kappa, \psi \rangle\rangle \in \{\mathtt{F}, \mathtt{T}\}$, then $\langle\langle \kappa, \psi \rangle\rangle = [\![\kappa, \psi]\!]$. Possibly, however, $\langle\langle \kappa, \psi \rangle\rangle = ?$ and $[\![\kappa, \psi]\!] \in \{\mathtt{F}, \mathtt{T}\}$.*

We say that a secret ψ is *well-specified* if for all noisy computations κ, we have that $[\![\kappa, \psi]\!] = \langle\langle \kappa, \psi \rangle\rangle$. Thus, the two semantics coincide for ψ. Equivalently, ψ is well-specified if $L(\mathcal{N}^?_\psi) = L(\mathcal{S}^?_\psi)$. The rationale behind the term "well-specified" is that, intuitively, the three-valued semantics loses information due to local dependencies that can be simplified. To see this, let us consider a few examples.

Recall that $[\![\kappa, \psi]\!] = \texttt{T}$ if $\pi \models \psi$ for all computations $\pi \in \text{fill}(\kappa)$. Accordingly, for every noisy computation κ and tautology ψ and, we have that $[\![\kappa, \psi]\!] = \texttt{T}$. In particular, $[\![\kappa, p \vee \neg p]\!] = \texttt{T}$, even for a noisy computation κ with $\kappa_0(p) =?$. On the other hand, for such a noisy computation κ, we have that $\langle\!\langle \kappa, p \vee \neg p \rangle\!\rangle =?$. This loss of information occurs not only with tautologies. For example, consider a noisy computation κ with $\kappa_0(q) = \texttt{F}$ and $\kappa_0(p) =?$. It is easy to see that $[\![\kappa, p \vee \neg(q \vee p)]\!] = \texttt{T}$ whereas $\langle\!\langle \kappa, p \vee \neg(q \vee p) \rangle\!\rangle =?$. Moreover, the loss happens not only in the propositional level. Assume that κ above continues with $\kappa_1 = \kappa_0$ and consider the LTL formula $\psi = (p \wedge \mathsf{X}\neg p)\mathsf{U}q$. Note that $[\![\kappa, \psi]\!] = \texttt{F}$ whereas $\langle\!\langle \kappa, \psi \rangle\!\rangle =?$.

Now, for the three examples above, we have that $p \vee \neg p = \texttt{T}$, $p \vee \neg(q \vee p) = p \vee \neg q$, and $(p \wedge \mathsf{X}\neg p)\mathsf{U}q = q \vee (p \wedge \mathsf{X}(q \wedge \neg p))$, thus, all three formulas can be simplified to formulas that describe the intention of the designer in a clearer way. As is the case with other forms of *inherent vacuity* [26, 33], the fact that a secret is not well-specified is valuable information for the designer, as it points to redundant complications in the formulation of the secret. Theorems 3 and 4 are useful also for this task. To see this, consider an LTL formula ψ, and recall that, by definition, ψ is well-specified iff $L(\mathcal{N}_\psi^?) = L(\mathcal{S}_\psi^?)$. By Lemma 5, it is always the case that $L(\mathcal{N}_\psi^?) \subseteq L(\mathcal{S}_\psi^?)$. Thus, ψ is well-specified iff $L(\mathcal{S}_\psi^?) \subseteq L(\mathcal{N}_\psi^?)$. Note, however, that the above only gives us an EXPSPACE upper bound for the problem, and we leave the exact complexity open (see Section 7).

4.1 Extension to multiple and LTL[$\mathcal{F}$] secrets

The constructions above handle a single secret in LTL. In this section we show how to extend them to multiple and LTL[$\mathcal{F}$] secrets. We start with multiple secrets. Recall that a set $S = \{\psi_1, \ldots, \psi_k\}$ of secrets cannot be composed to a single secret. Still, it is easy to extend the constructions above to such a set. First, in the semantics-based approach, we can extend Theorem 3 to S by taking an NGBW for the intersection language of the NGBWs $\mathcal{N}_{\psi_1}^?, \ldots, \mathcal{N}_{\psi_k}^?$ described there, hence the following theorem.

▶ **Theorem 6.** *Let $S = \{\psi_1, \ldots, \psi_k\}$ be a set of LTL formulas over AP. There exists an NGBW $\mathcal{N}_S^?$ over the alphabet 3^{AP} such that for every noisy computation $\kappa \in (3^{AP})^\omega$, we have that $\mathcal{N}_S^?$ accepts κ iff $[\![\kappa, \psi_i]\!] =?$ for all $1 \le i \le k$. Also, $\mathcal{N}_S^?$ has at most $2^{O(m)}$ states and index at most m, where $m = \sum_{i=1}^k |\psi_i|$.*

Then, in the syntax-based approach, the situation is even simpler, as there we can actually compose S to a single secret. Indeed, under the syntax-based three valued semantics, for every noisy computation κ, we have that $\langle\!\langle \kappa, \psi_i \vee \neg\psi_i \rangle\!\rangle =?$ iff $\langle\!\langle \kappa, \psi_i \rangle\!\rangle =?$, and otherwise $\langle\!\langle \kappa, \psi_i \vee \neg\psi_i \rangle\!\rangle = \texttt{T}$. Accordingly, $\langle\!\langle \kappa, (\psi_1 \vee \neg\psi_1) \vee \cdots \vee (\psi_k \vee \neg\psi_k) \rangle\!\rangle =?$ iff $\langle\!\langle \kappa, \psi_i \rangle\!\rangle =?$ for all $1 \le i \le k$. Thus, here, the fact $\psi_i \vee \neg\psi_i$ is a tautology and is thus not well-specified is surprisingly helpful.

We continue to LTL[$\mathcal{F}$] secrets. For two disjoint predicates $P_1, P_2 \subseteq [0,1]$, we say that κ $\langle P_1, P_2 \rangle$*-hides* ψ if there are two computations $\pi_1, \pi_2 \in \text{fill}(\kappa)$ such that $[\![\pi_1, \psi]\!] \in P_1$ and $[\![\pi_2, \psi]\!] \in P_2$. Thus, by observing κ, one cannot tell whether the satisfaction value of a computation that induces κ is in P_1 or P_2. Note that the semantics-based definition for LTL is a special case of the above definition, with $P_1 = \{0\}$ and $P_2 = \{1\}$. It is not hard to see that the same construction described in the proof of Theorem 3 can be applied to LTL[$\mathcal{F}$] formulas, with the automata $\mathcal{A}_\psi^{P_1}$ and $\mathcal{A}_\psi^{P_2}$ (see Theorem 1) replacing the automata for ψ and $\neg\psi$ there. Formally, we have the following.

▶ **Theorem 7.** *Let φ be an LTL[$\mathcal{F}$] formula over AP, and let $P_1, P_2 \subseteq [0,1]$ be two predicates. There exists an NGBW $\mathcal{N}^?_\varphi$ over the alphabet 3^{AP} such that for every noisy computation $\kappa \in (3^{AP})^\omega$, we have that $\mathcal{N}^?_\varphi$ accepts κ iff κ $\langle P_1, P_2\rangle$-hides ψ. Also, $\mathcal{N}^?_\varphi$ has at most $2^{O(|\varphi|)}$ states and index at most $|\varphi|$.*

Finally, handling a set S of LTL[$\mathcal{F}$] secrets combines Theorems 6 and 7: each secret ψ_i is given with predicates $P_1^i, P_2^i \subseteq [0,1]$, and the NGBW $\mathcal{N}^?_S$ is obtained by intersecting these defined in Theorem 7.

5 Solving Synthesis with Privacy

In this section we describe a solution for the problem of synthesis with privacy. Let φ be an LTL[$\mathcal{F}$] formula (the specification), $P \subseteq [0,1]$ a predicate, and ψ an LTL formula (the secret). Note that, for simplicity, we assume a single LTL secret. As described in Section 4.1, the extension to multiple and LTL[$\mathcal{F}$] secrets is easy. Consider a noisy computation $\kappa \in (3^{I\cup O})^\omega$. We say that κ is *$\langle \psi, \varphi, P\rangle$-good* if $[\![\kappa, \varphi]\!] \subseteq P$ and $[\![\kappa, \psi]\!] = ?$. Recall that we seek a noisy transducer $\mathcal{T} = \langle I, O, S, s_0, \eta, \tau, \mathfrak{m}\rangle$ such that for every input sequence $w_I \in (2^I)^\omega$, the noisy computations $\mathcal{T}_\mathfrak{m}(w_I)$ is $\langle \psi, \varphi, P\rangle$-good.

The next Theorem states that is possible to construct a DPW (and, in the case of well-specified secrets, also a UGCW) that recognizes $\langle \psi, \varphi, P\rangle$-good noisy computations (see Appendix B.2). Once such a DPW or UGCW is defined, the problem is similar to usual synthesis, except that the transducer we construct is noisy and has to generate both noisy assignments to the output signals and input-masking instructions for the input signals.

▶ **Theorem 8.** *Let φ be an LTL[$\mathcal{F}$] formula over AP, $P \subseteq [0,1]$ a predicate, and ψ an LTL formula.*

1. *There exists a DPW $\mathcal{D}^P_{\varphi,\psi}$ over the alphabet 3^{AP} that recognizes $\langle \psi, \varphi, P\rangle$-good noisy computations. The DPW $\mathcal{D}^P_{\varphi,\psi}$ has $2^{2^{O(|\varphi|+|\psi|)}}$ states and index $2^{O(|\varphi|+|\psi|)}$.*
2. *If ψ is well-specified, then there exists a UGCW $\mathcal{U}^P_{\varphi,\psi}$ over the alphabet 3^{AP} that recognizes $\langle \psi, \varphi, P\rangle$-good noisy computations. The UGCW $\mathcal{U}^P_{\varphi,\psi}$ has $2^{O(|\varphi|+|\psi|)}$ states and index at most $|\varphi| + |\psi|$.*

We proceed to define the notion of *noisy synthesis*, which refers to languages of noisy computations.

▶ **Definition 9.** *Consider a language $L \subseteq (3^{I\cup O})^\omega$. We say that a noisy I/O-transducer $\mathcal{T}$* realizes *L if for all $w_I \in (2^I)^\omega$, the noisy computation $\mathcal{T}_\mathfrak{m}(w_I)$ is in L. The* noisy synthesis *problem gets as input an automaton $\mathcal{A}$ over the alphabet $3^{I\cup O}$ and returns a noisy I/O-transducer $\mathcal{T}$ that realizes $L(\mathcal{A})$, or decides that no such transducer exists.*

The next theorem follows immediately from the definition. Together with the constructions in Theorem 8, it enables us to reduce synthesis with privacy to noisy synthesis.

▶ **Theorem 10.** *Consider an LTL[$\mathcal{F}$] specification φ, a predicate $P \subseteq [0,1]$, and an LTL secret ψ. A noisy I/O-transducer $\mathcal{T}$ realizes $\langle \varphi, P\rangle$ with privacy ψ iff $\mathcal{T}$ realizes $L(\mathcal{D}^P_{\varphi,\psi})$. When ψ is well-specified, then $\mathcal{T}$ realizes $\langle \varphi, P\rangle$ with privacy ψ iff $\mathcal{T}$ realizes $L(\mathcal{U}^P_{\varphi,\psi})$.*

Following Theorem 10, it is left to solve noisy synthesis for specifications given by a DPW or a UGCW. The algorithms are variants of these for traditional synthesis: For DPWs, we describe a reduction to deciding a parity game. For UGCWs, we describe a Safraless solution that is based on tree automata. In both solutions, we have to extend the solutions with mechanisms that let the system choose the masked signals and direct the game or the tree automaton accordingly. Due to the lack of space, the definitions of tree automata can be found in Appendix A.

5.1 Solution for a DPW

In this section we describe a solution for the noisy-synthesis problem of a DPW $\mathcal{D} = \langle 3^{I \cup O}, Q, q_0, \delta, \alpha \rangle$.

We reduce noisy synthesis of $\mathcal{D}$ to the problem of finding a winning strategy in a parity game $\mathcal{G}_{\mathcal{D}}$ that models the interaction between the system (player SYS) and the environment (player ENV). At each round, SYS gives ENV masking instructions and a noisy output letter, and then ENV responds with a noisy input assignment according to the masking instructions of SYS. Formally, $\mathcal{G}_{\mathcal{D}} = \langle V, E, v_0, \alpha' \rangle$, where V is the set of positions and is partitioned into two disjoint sets $V = V_{\text{ENV}} \cup V_{\text{SYS}}$. The positions in $V_{\text{SYS}} = Q$ are controlled by SYS, and the positions in $V_{\text{ENV}} = Q \times 2^I \times 3^O$ are controlled by ENV. The game starts in position $v_0 = q_0 \in V_{\text{SYS}}$, and it alternates between positions of SYS and ENV, i.e., $E \subseteq (V_{\text{SYS}} \times V_{\text{ENV}}) \cup (V_{\text{ENV}} \times V_{\text{SYS}})$. The exact definition of E is given by the following description of the possible moves in the game. For every $k \geq 0$ the k-th round of the game begins in a position $q_k \in V_{\text{SYS}}$ and proceeds as follows:

1. SYS chooses a noisy output assignment $o_k \in 3^O$, and a set of input signals $M_k \in 2^I$, and the game moves to the position $\langle q_k, M_k, o_k \rangle \in V_{\text{ENV}}$.
2. ENV chooses an input assignment $i_k \in 2^I$, which is masked into $i'_k = \mathit{hide}(M_k, i_k)$, and the game moves to the position $q_{k+1} = \delta(q_k, i'_k \cup o_k) \in V_{\text{SYS}}$.

An outcome of the game then consists of the following components:

- a noisy input word $w'_I = i'_0, i'_1, i'_2, \ldots, \in (3^I)^\omega$,
- a noisy output word $w_O = o_0, o_1, o_2, \ldots \in (3^O)^\omega$,
- a run $r = q_0, q_1, q_2, \ldots \in Q^\omega$ of $\mathcal{D}$ on $w'_I \cup w_O$.

Finally, the winning condition α' is induced by the acceptance condition α of $\mathcal{D}$; thus a vertex v with Q-component q has $\alpha'(v) = \alpha(q)$.

We can now state the correctness of the reduction (see proof in Appendix B.3).

▶ **Proposition 11.** *The DPW $\mathcal{D}$ is realizable by a noisy I/O-transducer iff* SYS *wins $\mathcal{G}_{\mathcal{D}}$.*

By Proposition 11, noisy synthesis of a DPW $\mathcal{D}$ can be solved in the same complexity as the problem of deciding a parity game played on $\mathcal{D}$. Hence, by Theorem 8, we have the following (see proof in Appendix B.4). The lower bound follows from the fact we can reduce synthesis with privacy requirements to synthesis with no such requirements by adding a dummy atomic proposition $p \in I \cup O$ and a secret that refers to p.

▶ **Theorem 12.** *The problem of LTL[$\mathcal{F}$] synthesis with privacy is 2EXPTIME-complete.*

5.2 Solution for a UGCW

In this section we describe a Safraless solution for the noisy-synthesis problem of a UGCW $\mathcal{U} = \langle 3^{I \cup O}, Q, q_0, \delta, \alpha \rangle$.

We translate $\mathcal{U}$ into a UGCT $\mathcal{U}'$ on $2^I \times 3^O$-labeled 3^I-trees that accept trees induced by noisy I/O-transducers that realize $\mathcal{U}$. We define $\mathcal{U}' = \langle 3^I, \Sigma, Q, Q_0, \delta', \alpha \rangle$, where $\Sigma = 2^I \times 3^O$, and $\delta' : Q \times \Sigma \to \mathcal{B}^+(\mathit{3}^I \times Q)$ is such that for every state $q \in Q$ and letter $\langle M, o\rangle \in \Sigma$, we have

$$\delta'(q, \langle M, o\rangle) = \bigwedge_{i \in 2^I} \bigwedge_{q' \in \delta(q, hide(M,i) \cup o)} \langle hide(M, i), q'\rangle$$

Note that if $\mathcal{U}'$ is at node v labeled $\langle M, o\rangle$, and $i' \in \mathit{3}^I$ is a noisy assignment such that $i'^{-1}(\{\mathtt{T}, \mathtt{F}\}) \neq M$, then $\mathcal{U}'$ sends no requirements to the subtree that is the i'-successor of v. On the other hand, for a noisy assignment $i' \in (\mathit{3}^I)$ with $i'^{-1}(\{\mathtt{T}, \mathtt{F}\}) = M$, there is at least one copy that is sent to the i'-successor of v. This corresponds to the behavior of a noisy transducer: from a state s with $\mathfrak{m}(s) = M$, the transducer is expected to handle every possible assignment to M, and when constructing a run, the assignments to signals not in $\mathfrak{m}(s)$ are ignored.

Formally, we have the following (see proof in Appendix B.5).

▶ **Proposition 13.** *The UGCW $\mathcal{U}$ is realizable by a noisy I/O-transducer iff $L(\mathcal{U}') = \varnothing$.*

By Proposition 13, noisy synthesis of a UGCW $\mathcal{U}$ can be reduced to the nonemptiness of a UGCT with the same state space and index. Hence, by [31] and Theorem 8, we have the following.

▶ **Theorem 14.** *The problem of LTL[$\mathcal{F}$] synthesis can be solved Safralessly in 2EXPTIME for well-specified secrets.*

6 On the Trade-off Between Utility and Privacy

Privacy involves loss of information, which makes it more difficult to realize specifications. Technically, missing information is quantified universally, and the realizing transducer has to satisfy the specification for all possible ways to fill it. In this section we discuss ways to examine and play with the trade off between utility, namely the satisfaction value of the specification φ, and privacy, namely the extent to which the satisfaction value of the secret ψ is revealed.

For secrets in LTL, which are Boolean, possible compensations on privacy include weakening of the secrets. One way to do it is to replace a secret ψ by a pair $\langle \theta, \psi\rangle$, indicating we care to keep the satisfaction value of ψ unknown only in noisy computations that satisfy θ. Note that, unlike the case of assumptions on the environment in synthesis [15], this cannot be achieved by changing the secret to $\theta \to \psi$. Indeed, the latter only means that we require the satisfaction value of $\theta \to \psi$ to be unknown. Our algorithms can be changed to address a $\langle \theta, \psi\rangle$ secret by replacing the automata constructed in Section 4 by ones that take θ into account, thus accept a noisy computation κ iff $[\![\kappa, \theta]\!] \neq \mathtt{T}$ or $[\![\kappa, \psi]\!] = ?$.

For secrets in LTL[$\mathcal{F}$], taking the predicates described in Section 4 to be closed upward, we can say, given $h \in (0, 1]$, that a noisy computation κ *h-hides* a secret ψ in LTL[$\mathcal{F}$] if $\max[\![\kappa, \psi]\!] - \min[\![\kappa, \psi]\!] \geq h$. Thus, knowing κ, our uncertainty about the satisfaction value of ψ is at least h. Note that LTL secrets are a special case of the above definition, with $h = 1$. Now, in synthesis with LTL[$\mathcal{F}$] secrets, the input includes, in addition to the specification ψ, a threshold v for it, and a secret ψ, also a threshold h for the secret, and we require the generated computation to both satisfy φ with value at least v and to h-hide ψ. Our algorithm can be changed to address the variants of the problem in which either v or h are given, and the goal is to maximize the other parameter, having the first one as a hard constraint. In particular, when h is given, we must h-hide φ, and seek a transducer that, under this constraint, maximizes the satisfaction value of φ. Technically, this amounts to replacing the

DPW $\mathcal{D}^?_\psi$ by an automaton that accepts noisy computations that h-hides ψ. For the other case, where we fix v, the solution involves a search for h, which involves polynomially many executions of our algorithm.

7 Discussion

We introduced a simple yet powerful framework for synthesis of systems that preserve privacy. In our framework, the system and the environment may hide the values of signals they control, and they are guaranteed that "secrets" they care about are not going to be revealed. When one thinks about privacy, the first thing that comes to mind is privacy of *data* (age, salary, illnesses, gender, etc.). An underlying assumption of our work is that, in the context of reactive systems, privacy should concern *behaviours*. Thus, "secrets" are ω-regular languages, possibly weighted ones. A nice analogy is the way games are studied in the formal-methods community: classical game theory studies games with quantitative objectives, based on costs and rewards, whereas classical games in formal methods have ω-regular objectives, possibly weighted ones [9].

We introduced the key ideas behind the approach of "behavioral secrets", namely a use of a three-valued semantics for the specification formalism. We also described how existing algorithms for synthesis, in fact even high-quality synthesis, can be extended to handle privacy. The latter is simple for traditional synthesis algorithms and involved a study of a syntax-based three-valued semantics for Safraless algorithms.

Beyond the challenge of extending the framework to richer settings of the synthesis problem (e.g., rational, distributed, infinite-state, or probabilistic systems [4, 25, 32, 38]), we find the following research directions, which address the basic idea of behavioral secrets, very interesting.

A stochastic approach Recall that in the multi-valued setting, we followed the worst-case approach, thus the quality of the synthesized system is the minimal satisfaction value of the specification φ in some interaction. In the stochastic approach, we assume a given distribution on the input sequences, and the quality of the system is the expected satisfaction value of φ [2]. Extending synthesis with privacy to a stochastic approach, we seek noisy I/O-transducer that maximizes the expected satisfaction value of φ while hiding ψ with probability 1. Technically, as the valuation of φ refers to its expected satisfaction value, whereas hiding of the value of ψ is a hard constraint, the synthesis algorithm has to combine both types of objectives [3, 11, 6, 7]. The stochastic approach is of special interest when studying the trade-off between the expected uncertainty of ψ against the expected satisfaction of φ.

Specifying secrets In our framework, a behavior ψ in LTL is kept secret if its satisfaction is unknown. More sophisticated definitions can refer to the *probability* that ψ is satisfied, given the revealed information, or, even more sophisticated, to the extent in which the revealed information changes the probability of ψ to be satisfied. For example, if the secret is $\psi = p \wedge q$ and we revealed that q holds, we still do not know the satisfaction value of ψ, yet we did learn that the probability of its satisfaction has increased. A good treatment of definitions that take probability in mind should address the fact that computations are sampled from the set of computations that satisfy the specification, which poses interesting technical challenges.

Multiple view-points In our framework, revealed information is known to all parties:

the system, the environment, and an observer to the interaction. In some settings, the environment is composed of several components who are willing to share information with the system, but not with each other. Also, not all components care about the satisfaction of all specifications. Such settings can be addressed by extending the framework to handle *multiple-viewpoint assignments* to input and output signals. Thus, if the setting involves a set C of components, values are in $\{\mathtt{T}, \mathtt{F}\} \times 2^C$, specifying both the value and the subset of components that see it. Technically, the extension can be handled by using lattice automata and synthesis algorithms for them [28, 29].

Perturbation of signals Our framework handles Boolean signals and allows the system and environment to hide the values of signals they control. In some settings, the Boolean signals encode richer values, or the setting includes non-Boolean inputs in the first place (e.g., augmenting LTL with Presburger arithmetic [19] or register automata with linear arithmetic over the rationals [16]). In such settings, it makes sense to allow the components not to entirely hide the value of their variables, but rather to *perturb* it to an approximated value. A synthesizing transducer should then perturb the value of the (non-Boolean) secret while satisfying the specification, possibly up to some perturbation.

Syntax-based three-valued semantic As discussed in Section 4, our syntax-based three-valued semantic for LTL does not coincide with the semantics-based one. We described an EXPSPACE algorithm for deciding whether a given LTL formula is well-specified (that is, the two semantics coincide for it), and left the tight complexity of the problem open. Interestingly, the problem has similarities with both the satisfiability problem of $\forall$LTL, namely LTL augmented with universally-quantified propositions, which is EXPTIME-complete [40], and with *inherent vacuity*, namely deciding whether a given LTL formula ψ has a subformula θ such that ψ and $\forall x.\psi[\theta \leftarrow x]$ are equivalent, which is PSPACE-complete [26].

References

1. S. Almagor, U. Boker, and O. Kupferman. Formalizing and reasoning about quality. *Journal of the ACM*, 63(3):24:1–24:56, 2016.
2. S. Almagor and O. Kupferman. High-quality synthesis against stochastic environments. In *Proc. 25th Annual Conf. of the European Association for Computer Science Logic*, volume 62 of *LIPIcs*, pages 28:1–28:17, 2016.
3. S. Almagor, O. Kupferman, and Y. Velner. Minimizing expected cost under hard boolean constraints, with applications to quantitative synthesis. In *Proc. 27th Int. Conf. on Concurrency Theory*, volume 59 of *LIPIcs*, pages 9:1–9:15, 2016.
4. C. Baier, M. Größer, M. Leucker, B. Bollig, and F. Ciesinski. Controller synthesis for probabilistic systems. In *Exploring New Frontiers of Theoretical Informatics, IFIP 18th World Computer Congress, 3rd International Conference on Theoretical Computer Science*, volume 155 of *IFIP*, pages 493–506. Kluwer/Springer, 2004.
5. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S.P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012.
6. R. Berthon, S. Guha, and J-F Raskin. Mixing probabilistic and non-probabilistic objectives in markov decision processes. In *Proc. 35th IEEE Symp. on Logic in Computer Science*, pages 195–208. ACM, 2020.
7. R. Berthon, M. Randour, and J-F. Raskin. Threshold constraints with guarantees for parity objectives in markov decision processes. In *Proc. 44th Int. Colloq. on Automata, Languages, and Programming*, volume 80 of *LIPIcs*, pages 121:1–121:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

8 R. Bloem, K. Chatterjee, T. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *Proc. 21st Int. Conf. on Computer Aided Verification*, volume 5643 of *Lecture Notes in Computer Science*, pages 140–156. Springer, 2009.

9 R. Bloem, K. Chatterjee, and B. Jobstmann. Graph games and reactive synthesis. In *Handbook of Model Checking.*, pages 921–962. Springer, 2018.

10 G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *Proc. 11th Int. Conf. on Computer Aided Verification*, pages 274–287, 1999.

11 V. Bruyère, E. Filiot, M. Randour, and J-F. Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. In *Proc. 31th Symp. on Theoretical Aspects of Computer Science*, volume 25 of *LIPIcs*, pages 199–213, 2014.

12 C.S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan. Deciding parity games in quasipolynomial time. In *Proc. 49th ACM Symp. on Theory of Computing*, pages 252–263, 2017.

13 P. Cerný, K. Chatterjee, T.A. Henzinger, A. Radhakrishna, and R. Singh. Quantitative synthesis for concurrent programs. In *Proc. 23rd Int. Conf. on Computer Aided Verification*, pages 243–259, 2011.

14 K. Chatterjee, L. Doyen, T. A. Henzinger, and J-F. Raskin. Algorithms for ω-regular games with imperfect information. In *Proc. 15th Annual Conf. of the European Association for Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 287–302, 2006.

15 K. Chatterjee, T. Henzinger, and B. Jobstmann. Environment assumptions for synthesis. In *Proc. 19th Int. Conf. on Concurrency Theory*, volume 5201 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2008.

16 Y-F. Chen, O. Lengál, T. Tan, and Z. Wu. Register automata with linear arithmetic. In *Proc. 32nd IEEE Symp. on Logic in Computer Science*, pages 1–12, 2017.

17 M.R. Clarkson, B. Finkbeiner, M. Koleini, K.K. Micinski, M.N. Rabe, and C. Sánchez. Temporal logics for hyperproperties. In *3rd International Conference on Principles of Security and Trust*, volume 8414 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2014.

18 L. de Alfaro and P. Roy. Solving games via three-valued abstraction refinement. *Inf. Comput.*, 208(6):666–676, 2010.

19 S. Demri. Linear-time temporal logics with presburger constraints: an overview. *Journal of Applied Non-Classical Logics*, 16(3-4):311–348, 2006.

20 I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the 22nd ACM Symposium on Principles of Database Systems*, pages 202–210. ACM, 2003.

21 J. Dubreil, Ph. Darondeau, and H. Marchand. Supervisory control for opacity. *IEEE Transactions on Automatic Control*, 55(5):1089–1100, 2010.

22 C. Dwork, F. McSherry, K. Nissim, and A.D. Smith. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7(3):17–51, 2016.

23 E.A. Emerson and C.-L. Lei. Efficient model checking in fragments of the propositional μ-calculus. In *Proc. 1st IEEE Symp. on Logic in Computer Science*, pages 267–278, 1986.

24 B. Finkbeiner, C. Hahn, P. Lukert, M. Stenger, and L. Tentrup. Synthesis from hyperproperties. *Acta Informatica*, 57(1-2):137–163, 2020.

25 D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *Proc. 16th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 6015 of *Lecture Notes in Computer Science*, pages 190–204. Springer, 2010.

26 D. Fisman, O. Kupferman, S. Seinvald, and M.Y. Vardi. A framework for inherent vacuity. In *4th International Haifa Verification Conference*, volume 5394 of *Lecture Notes in Computer Science*, pages 7–22. Springer, 2008.

27 S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 45(3):882–929, 2016.

28 A. Gurfinkel and M. Chechik. Multi-valued model-checking via classical model-checking. In *Proc. 14th Int. Conf. on Concurrency Theory*, pages 263–277. Springer-Verlag, 2003.
29 O. Kupferman and Y. Lustig. Lattice automata. In *Proc. 8th Int. Conf. on Verification, Model Checking, and Abstract Interpretation*, volume 4349 of *Lecture Notes in Computer Science*, pages 199 – 213. Springer, 2007.
30 O. Kupferman and M.Y. Vardi. Synthesis with incomplete information. In *Advances in Temporal Logic*, pages 109–127. Kluwer Academic Publishers, 2000.
31 O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, 2005.
32 M. Z. Kwiatkowska. Model checking and strategy synthesis for stochastic games: From theory to practice. In *Proc. 43th Int. Colloq. on Automata, Languages, and Programming*, volume 55 of *LIPIcs*, pages 4:1–4:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
33 S. Maoz and R. Shalom. Inherent vacuity for GR(1) specifications. In *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 99–110. ACM, 2020.
34 N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *Proc. 21st IEEE Symp. on Logic in Computer Science*, pages 255–264. IEEE press, 2006.
35 J.H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and Systems Science*, 29:274–301, 1984.
36 S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.
37 S. Safra. *Complexity of automata on infinite objects*. PhD thesis, Weizmann Institute of Science, 1989.
38 S. Schewe. *Synthesis of distributed systems*. PhD thesis, Saarland University, Saarbrücken, Germany, 2008.
39 S. Shoham and O. Grumberg. A game-based framework for CTL counterexamples and 3-valued abstraction-refinement. In *Proc. 15th Int. Conf. on Computer Aided Verification*, volume 2725, pages 275–287, 2003.
40 A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.
41 M.Y. Vardi. From verification to synthesis. In *VSTTE*, page 2, 2008.
42 M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.
43 Y. Wu, V. Raman, B.C. Rawlings, S. Lafortune, and S.A. Seshia. Synthesis of obfuscation policies to ensure privacy and utility. *Journal of Automated Reasoning*, 60(1):107–131, 2018.

A Tree Automata

Given a set D of directions, a *D-tree* is a set $T \subseteq D^*$ such that if $x \cdot c \in T$, where $x \in D^*$ and $c \in D$, then also $x \in T$. The elements of T are called *nodes*, and the empty word ε is the *root* of T. For every $x \in T$, the nodes $x \cdot c$, for $c \in D$, are the *successors* of x. A *path* π of a tree T is a set $\pi \subseteq T$ such that $\varepsilon \in \pi$ and for every $x \in \pi$, either x is a leaf or there exists a unique $c \in D$ such that $x \cdot c \in \pi$. Given an alphabet Σ, a *Σ-labeled D-tree* is a pair $\langle T, \tau \rangle$ where T is a tree and $\tau : T \to \Sigma$ maps each node of T to a letter in Σ.

For a set X, let $\mathcal{B}^+(X)$ be the set of positive Boolean formulas over X (i.e., Boolean formulas built from elements in X using $\wedge$ and $\vee$), where we also allow the formulas `T` and `F`. For a set $Y \subseteq X$ and a formula $\theta \in \mathcal{B}^+(X)$, we say that Y *satisfies* θ iff assigning `T` to elements in Y and assigning `F` to elements in $X \setminus Y$ makes θ true. An *alternating tree automaton* is $\mathcal{A} = \langle \Sigma, D, Q, q_{in}, \delta, \alpha \rangle$, where Σ is the input alphabet, D is a set of directions, Q is a finite set of states, $\delta : Q \times \Sigma \to \mathcal{B}^+(D \times Q)$ is a transition function, $q_{in} \in Q$ is an initial state, and α is an acceptance condition. We consider here the Büchi, co-Büchi, and parity acceptance conditions. For a state $q \in Q$, we use $\mathcal{A}^q$ to denote the automaton obtained from $\mathcal{A}$ by setting the initial state to be q. The *size* of $\mathcal{A}$, denoted $|\mathcal{A}|$, is the sum of lengths of formulas that appear in δ.

The alternating automaton $\mathcal{A}$ runs on Σ-labeled D-trees. A *run* of $\mathcal{A}$ over a Σ-labeled D-tree $\langle T, \tau \rangle$ is a $(T \times Q)$-labeled $\mathbb{N}$-tree $\langle T_r, r \rangle$. Each node of T_r corresponds to a node of T. A node in T_r, labeled by (x, q), describes a copy of the automaton that reads the node x of T and visits the state q. Note that many nodes of T_r can correspond to the same node x of T. The labels of a node and its successors have to satisfy the transition function. Formally, $\langle T_r, r \rangle$ satisfies the following:

1. $\varepsilon \in T_r$ and $r(\varepsilon) = \langle \varepsilon, q_{in} \rangle$.
2. Let $y \in T_r$ with $r(y) = \langle x, q \rangle$ and $\delta(q, \tau(x)) = \theta$. Then there is a (possibly empty) set $S = \{(c_0, q_0), (c_1, q_1), \ldots, (c_{n-1}, q_{n-1})\} \subseteq D \times Q$, such that S satisfies θ, and for all $0 \leq i \leq n-1$, we have $y \cdot i \in T_r$ and $r(y \cdot i) = \langle x \cdot c_i, q_i \rangle$.

For example, if $\langle T, \tau \rangle$ is a $\{0,1\}$-tree with $\tau(\varepsilon) = a$ and $\delta(q_{in}, a) = ((0, q_1) \vee (0, q_2)) \wedge ((0, q_3) \vee (1, q_2))$, then, at level 1, the run $\langle T_r, r \rangle$ includes a node labeled $(0, q_1)$ or a node labeled $(0, q_2)$, and includes a node labeled $(0, q_3)$ or a node labeled $(1, q_2)$. Note that if, for some y, the transition function δ has the value $\texttt{T}$, then y need not have successors. Also, δ can never have the value $\texttt{F}$ in a run.

A run $\langle T_r, r \rangle$ is accepting if all its infinite paths satisfy the acceptance condition. Given a run $\langle T_r, r \rangle$ and an infinite path $\pi \subseteq T_r$, let $\mathit{inf}(\pi) \subseteq Q$ be such that $q \in \mathit{inf}(\pi)$ if and only if there are infinitely many $y \in \pi$ for which $r(y) \in T \times \{q\}$. That is, $\mathit{inf}(\pi)$ contains exactly all the states that appear infinitely often in π. The acceptance condition for alternating tree automata are similar to these defined for word automata, except that here, $\inf(\pi)$ has to satisfy the condition α for all paths π. We denote by $L(\mathcal{A})$ the set of all Σ-labeled trees that $\mathcal{A}$ accepts.

The alternating automaton $\mathcal{A}$ is *nondeterministic* if for all the formulas that appear in δ, if (c_1, q_1) and (c_2, q_2) are conjunctively related, then $c_1 \neq c_2$. (i.e., if the transition is rewritten in disjunctive normal form, there is at most one element of $\{c\} \times Q$, for each $c \in D$, in each disjunct). The automaton $\mathcal{A}$ is *universal* if all the formulas that appear in δ are conjunctions of atoms in $D \times Q$, and $\mathcal{A}$ is *deterministic* if it is both nondeterministic and universal. Note that word automata are a special case of tree automata, with $|D| = 1$.

B Missing Proofs

B.1 Proof of Theorem 4

For an LTL formula ψ, the *closure* of ψ, denoted $cl(\psi)$, is the set of ψ's subformulas and their negation ($\neg\neg\psi$ is identified with ψ). Formally, $cl(\psi)$ is the smallest set of formulas that satisfy the following.

- $\psi \in cl(\psi)$.
- If $\psi_1 \in cl(\psi)$ then $\neg\psi_1 \in cl(\psi)$.
- If $\neg\psi_1 \in cl(\psi)$ then $\psi_1 \in cl(\psi)$.
- If $\psi_1 \vee \psi_2 \in cl(\psi)$ then $\psi_1 \in cl(\psi)$ and $\psi_2 \in cl(\psi)$.
- If $\mathsf{X}\psi_1 \in cl(\psi)$ then $\psi_1 \in cl(\psi)$.
- If $\psi_1 \mathsf{U} \psi_2 \in cl(\psi)$ then $\psi_1 \in cl(\psi)$ and $\psi_2 \in cl(\psi)$.

Consider the set $cl(\psi)$. We say that a function $f \in 3^{cl(\psi)}$ is *consistent* if f does not have propositional inconsistency. Thus, f satisfies the following conditions.

1. For every formula $\psi_1 \in cl(\psi)$, one of the following holds:
   - $f(\psi_1) = \texttt{T}$ and $f(\neg\psi_1) = \texttt{F}$,
   - $f(\psi_1) = \texttt{F}$ and $f(\neg\psi_1) = \texttt{T}$, or
   - $f(\psi_1) =?$ and $f(\neg\psi_1) =?$.
2. For every formula of the form $\psi_1 \vee \psi_2 \in cl(\psi)$, the following holds.
   - $f(\psi_1 \vee \psi_2) = \texttt{T}$ iff $f(\psi_1) = \texttt{T}$ or $f(\psi_2) = \texttt{T}$.
   - $f(\psi_1 \vee \psi_2) = \texttt{F}$ iff $f(\psi_1) = \texttt{F}$ and $f(\psi_2) = \texttt{F}$.

   Note that it follows that $f(\psi_1 \vee \psi_2) =?$ iff $f(\psi_1) =?$ and $f(\psi_2) \in \{?, \texttt{F}\}$, or $f(\psi_2) =?$ and $f(\psi_1) \in \{?, \texttt{F}\}$.

Now, we define $\mathcal{S}^?_\psi = \langle 3^{AP}, Q, \delta, Q_0, \alpha\rangle$, where

- The state space $Q \subseteq 3^{cl(\psi)}$ is the set of all consistent functions.
- Let f and f' be two states in Q, and let $\sigma \in 3^{AP}$ be a letter. Then, $f' \in \delta(f, \sigma)$ if the following hold.
  1. For every $p \in AP$, we have that $\sigma(p) = f(p)$. Thus, σ agrees with f on the atomic propositions.
  2. For all $X\psi_1 \in cl(\psi)$, we have that $f(X\psi_1) = f'(\psi_1)$, and
  3. For all $\psi_1 U \psi_2 \in cl(\psi)$, we have
     - $f(\psi_1 U \psi_2) = \texttt{T}$ iff $f(\psi_2) = \texttt{T}$ or $(f(\psi_1) = \texttt{T}$ and $f'(\psi_1 U \psi_2) = \texttt{T})$.
     - $f(\psi_1 U \psi_2) = \texttt{F}$ iff $f(\psi_2) = \texttt{F}$ and $(f(\psi_1) = \texttt{F}$ or $f'(\psi_1 U \psi_2) = \texttt{F})$.

     Note that $f(\psi_1 U \psi_2) =?$ iff one of the following hold:
     - $f(\psi_2) =?$ and $(f(\psi_1) \neq \texttt{T}$ or $f'(\psi_1 U \psi_2) \neq \texttt{T})$.
     - $f(\psi_2) = \texttt{F}$, and $f(\psi_1) = \texttt{T}$ and $f'(\psi_1 U \psi_2) =?$.
     - $f(\psi_2) = \texttt{F}$, and $f(\psi_1) =?$ and $f'(\psi_1 U \psi_2) \neq \texttt{F}$.
- $Q_0 \subseteq Q$ is the set of all states $f \in Q$ for which $f(\psi) =?$.
- Every formula $\psi_1 U \psi_2$ contributes to α the two sets $\alpha^{\texttt{T}}_{\psi_1 U \psi_2} = \{f \in Q : f(\psi_2) = \texttt{T}$ or $f(\psi_1 U \psi_2) \neq \texttt{T}\}$. and $\alpha^?_{\psi_1 U \psi_2} = \{f \in Q : f(\psi_2) =?$ or $f(\psi_1 U \psi_2) \neq ?\}$.
  Thus, if a run eventually visits only states in which the satisfaction value of $\psi_1 U \psi_2$ is T, then it should visit infinitely many states in which the satisfaction value of ψ_2 is T, and if a run eventually visits only states in which the satisfaction value of $\psi_1 U \psi_2$ is ?, then it should visit infinitely many states in which the satisfaction value of ψ_2 is ?.

Finally, $\mathcal{U}^?_\psi$ is obtained by dualizing the NGBW $\mathcal{S}^{\neg ?}_\psi$, which is similar to $\mathcal{S}^?_\psi$, except that $Q_0 \subseteq Q$ is the set of all states $f \in Q$ for which $f(\psi) \neq ?$.

B.2 Proof of Theorem 8

Given φ and P, let $\bar{P}$ be the predicate that complements P, thus $\bar{P} = [0,1] \setminus P$. By Theorem 1, we can construct an NGBW $\mathcal{A}^{\bar{P}}_\varphi = \langle 2^{AP}, Q, Q_0, \delta, \alpha\rangle$ such that for every computation $\pi \in (2^{AP})^\omega$, it holds that $\mathcal{A}^{\bar{P}}_\varphi$ accepts π iff $[\![\pi, \varphi]\!] \notin P$. Also, $\mathcal{A}^{\bar{P}}_\varphi$ has at most $2^{O(|\varphi|)}$ states and index at most $|\varphi|$. Let $\mathcal{N}^{\bar{P}}_\varphi = \langle 3^{AP}, Q, Q_0, \delta', \alpha\rangle$ be the NGBW obtained from $\mathcal{A}^{\bar{P}}_\varphi$ by letting it guess an assignment to atomic propositions whose value is unknown. Formally, for every state $q \in Q$ and letter $\sigma' \in 3^{AP}$, we have that $\delta'(q, \sigma') = \bigcup\{\delta(q, \sigma) : \sigma \in 2^{AP}$ is such that $\sigma' \leq_{\texttt{info}} \sigma\}$. It is easy to see that $\mathcal{N}^{\bar{P}}_\varphi$ accepts a noisy computation

$\kappa \in (3^{AP})^\omega$ iff $[\![\kappa, \varphi]\!] \cap \bar{P} \neq \emptyset$. By dualizing $\mathcal{N}_\varphi^{\bar{P}}$, we get a UGCW $\mathcal{U}_\varphi^P$ that accepts a noisy computation $\kappa \in (3^{AP})^\omega$ iff $[\![\kappa, \varphi]\!] \subseteq P$.

By Theorem 3, given ψ, we can construct an NGBW $\mathcal{N}_\psi^?$ over the alphabet 3^{AP} such that for every noisy computation $\kappa \in (3^{AP})^\omega$, we have that $\mathcal{N}_\psi^?$ accepts κ iff $[\![\kappa, \psi]\!] =?$. The NGBW $\mathcal{N}_\psi^?$ has at most $2^{O(|\psi|)}$ states and index at most $|\varphi|$. Also, by Theorem 4, when ψ is well-specified, we can replace $\mathcal{N}_\psi^?$ by a UGCW $\mathcal{U}_\psi^?$.

Now, the desired UGCW $\mathcal{U}_{\varphi,\psi}^P$ can be obtained by taking the intersection of the UGCWs $\mathcal{U}_\varphi^P$ and $\mathcal{U}_\psi^?$. Such an intersection does not involve a blow up (intersection of universal automata is dual to union of nondeterministic automata), and we end up with a UGCW with $2^{O(|\varphi|+|\psi|)}$ states and index at most $|\varphi| + |\psi|$.

In order to obtain the desired DPW $\mathcal{D}_{\varphi,\psi}^P$, we first co-determinize $\mathcal{N}_\varphi^{\bar{P}}$, and get a DPW $\mathcal{D}_\varphi^P$ that accepts a noisy computation $\kappa \in (3^{AP})^\omega$ iff $[\![\kappa, \varphi]\!] \subseteq P$. By [36, 34], the DPW $\mathcal{D}_\varphi^P$ has $2^{2^{O(|\varphi|)}}$ states and index $2^{O(|\varphi|)}$. Then, we determinize $\mathcal{N}_\psi^?$ and get a DPW $\mathcal{D}_\psi^?$ with at most $2^{2^{O(|\psi|)}}$ states and index $2^{O(|\psi|)}$ such that $\mathcal{D}_\psi^?$ accepts a noisy computation $\kappa \in (3^{AP})^\omega$ iff $[\![\kappa, \psi]\!] = \{0, 1\}$. The DPW $\mathcal{D}_{\varphi,\psi}^P$ is then obtained by taking the intersection of $\mathcal{D}_\varphi^P$ and $\mathcal{D}_\psi^?$. Since intersection of DPWs involve an exponential blow up only in their indices, the required bounds on the state space and index follows.

In more detail, Parity automata can be translated into Street automata on top of the same structure and with index of the same order. Thus, we may treat both automata as Streett automata of size and index of the same order. Then, we take the intersection DSW which is of size $2^{k_\varphi + k_\psi}$ and index $k_\varphi + k_\psi$. By [37], a deterministic Streett automaton with m states and index k can be translated into a deterministic Rabin automaton with $\Theta(m 2^{k \log k})$ states and index $t = \Theta(k)$. The pairs in the acceptance condition in the Rabin automaton $(\langle B_i, G_i \rangle)_{i=1}^t$ are such that $B_i \subseteq B_j$ for all $i \leq j$ and all of the G_i are disjoint. Thus, it is not hard to see that the parity condition that gives G_i priority $2i$, and $B_i \setminus B_{i-1}$ priority $2i - 1$, and all other states priority $2t + 1$, defines an equivalent deterministic parity automaton, with states and index of the same order as the Rabin automaton. Hence, the DPW $\mathcal{A}$ for the intersection language has $2^{k_\varphi + k_\psi} 2^{(k_\varphi + k_\psi) \log(k_\varphi + k_\psi)} \leq 2^{2^{O(|\varphi|+|\psi|)}}$ states and index $O(k_\varphi + k_\psi) \leq 2^{O(|\varphi|+|\psi|)}$.

B.3 Proof of Proposition 11

We partition the proposition into two propositions.

▶ **Proposition 15.** *If $\mathcal{G}_\mathcal{D}$ is winning for* Sys*, then a noisy I/O-transducer $\mathcal{T}$ that realizes $\mathcal{D}$ can be constructed on top of $\mathcal{D}$ in time $O(n^k)$, where n is the number of positions in $\mathcal{G}_\mathcal{D}$ and k is the index of $\mathcal{D}$.*

Proof. Since parity games enjoy memoryless-determinacy, it follows that Sys wins iff it has a memoryless strategy. Thus assume that Sys wins $\mathcal{G}_\mathcal{D}$ and let $f_{\text{Sys}} : V_{\text{Sys}} \to V_{\text{Env}}$ be a winning memoryless strategy for Sys. Note that such a winning memoryless strategy f_{Sys} can be computed in time $O(n^k)$ [23] (in fact less, using improved algorithms for parity games [12]). We define a noisy I/O-transducer $\mathcal{T}$ as follows. The set of states of the transducer $\mathcal{T}$ is $S = V_{\text{Sys}} = Q$. For a state $q \in S$, let $f_{\text{Sys}}(q) = \langle q, M, o \rangle$, we set $\tau(q) = o$ and $\mathfrak{m}(q) = M$. Then, for $i' \in 3^I$ for which there exists $i \in 2^I$ with $i' = \mathit{hide}(M, i)$, we define the transition function by $\eta(q, i') = \delta(q, i' \cup o)$, and otherwise, if there is no such $i \in 2^I$, then we define $\eta(q, i')$ to be an arbitrary state (Recall that runs of $\mathcal{T}$ does not use such transitions of η). In other words, we let Env play with $i \in 2^I$ from $\langle q, M, o \rangle$, and move to the appropriate i-successor in the game. Notice that for all $w_I \in (2^I)^\omega$ the computation

$\mathcal{T}_{\mathfrak{m}}(w_I) = (i'_0 \cup o_0), (i'_1 \cup o_1), \ldots \in (3^{I \cup O})^\omega$ is obtained from the input and output components of the outcome of the game $\mathcal{G}_\mathcal{D}$ when ENV plays with $w_I = i_0, i_1, i_2, \ldots \in (2^I)^\omega$ and SYS plays according to the strategy f_{SYS}. Hence, since f_{SYS} is winning for the System, it follows that for all $w_I \in (2^I)^\omega$, the run of $\mathcal{D}$ over $\mathcal{T}_{\mathfrak{m}}(w_I)$ is accepting. That is, $\mathcal{T}$ is a noisy I/O-transducer that realizes $\mathcal{D}$. ◀

▶ **Proposition 16.** *If $\mathcal{D}$ is realizable with a noisy I/O-transducer, then* SYS *wins $\mathcal{G}_\mathcal{D}$.*

Proof. Assume that $\mathcal{T} = \langle I, O, \mathcal{L}, S, \eta, \tau, \mathfrak{m} \rangle$ is a noisy I/O-transducer that realizes $\mathcal{D}$, we will construct a winning strategy f_{SYS} that uses $\mathcal{T}$ as a memory structure. Let W be the set of all finite paths in $\mathcal{G}_\mathcal{D}$ that start in $v_0 = q_0 \in V_{\text{SYS}}$ and end in some position $v_k \in V_{\text{SYS}}$ that belongs to SYS. We define the strategy $f_{\text{SYS}} : W \nrightarrow V_{\text{ENV}}$ as a partial function, where f_{SYS} is defined on $\langle q_0 \rangle \in W$, and for all $\rho = \langle q_0, \langle q_0, M_0, o_0 \rangle, q_1, \ldots, \langle q_{k-1}, M_{k-1}, o_{k-1} \rangle, q_k \rangle \in W$, if f_{SYS} is defined on ρ, and $f_{\text{SYS}}(\rho) = \langle q_k, M_k, o_k \rangle$, then for all $i \in 2^I$, if $q_{k+1} = \delta(q_k, \mathit{hide}(M_k, i) \cup o_k)$, then f_{SYS} is also defined on $\rho' = \langle q_0, \langle q_0, M_0, o_0 \rangle, \ldots, \langle q_k, M_k, o_k \rangle, q_{k+1} \rangle \in W$. Namely, f_{SYS} is defined on ρ', which is the extension of ρ when *Sys* plays with f_{SYS}, hence moves to $f_{\text{SYS}}(\rho) = \langle q_k, M_k, o_k \rangle$, and then ENV proceeds to $q_{k+1} = \delta(q_k, \mathit{hide}(M_k, i) \cup o_k)$ for some $i \in 2^I$. In order to define f_{SYS} we also define two more partial functions $f_S : W \nrightarrow S$ and $f_I : W \nrightarrow 2^I$. Intuitively, f_I guesses the last input letter played by ENV, and f_S simulates the run of $\mathcal{T}$ on the word guessed by f_I. The functions f_S and f_I have the same domain as f_{SYS}, with the only exception that f_I is not defined on the path $\rho = \langle q_0 \rangle$, as ENV haven't yet played, and hence there's nothing for f_I to guess. We define f_S, f_I and f_{SYS} by induction. First, for $\rho = \langle q_0 \rangle$, let $f_S(\rho) = s_0$, where $s_0 \in S$ is the initial state of $\mathcal{T}$, and let $f_{\text{SYS}}(\rho) = \langle q_0, \mathfrak{m}(f_S(q_0)), \tau(f_S(q_0)) \rangle$. Then, assume that f_S and f_{SYS} have been defined on $\rho = \langle q_0, \langle q_0, M_0, o_0 \rangle, q_1, \ldots, \langle q_{k-1}, M_{k-1}, o_{k-1} \rangle, q_k \rangle \in W$, and let $f_{\text{SYS}}(\rho) = \langle q_k, M_k, o_k \rangle \in V_{\text{ENV}}$. Consider $q_{k+1} \in V_{\text{SYS}}$ such that $(f_{\text{SYS}}(\rho), q_{k+1}) \in E$. I.e., q_{k+1} is a possible move of ENV from $f_{\text{SYS}}(\rho) = \langle q_k, M_k, o_k \rangle$. Let $i_k \in 2^I$ be some input letter such that $q_{k+1} = \delta(q_k, \mathit{hide}(M_k, i_k) \cup o_k)$. Note that such an input letter $i_k \in 2^I$ exists since q_{k+1} is a successor of the ENV-position $f_{\text{SYS}}(\rho) = \langle q_k, M_k, o_k \rangle$. Thus for the extension $\rho' = \langle q_0, \langle q_0, M_0, o_0 \rangle, \ldots, \langle q_k, M_k, o_k \rangle, q_{k+1} \rangle \in W$ of ρ, we set $f_I(\rho') = i_k$, and $f_S(\rho') = \eta(f_S(\rho), \mathit{hide}(M_k, i_k))$ and $f_{\text{SYS}}(\rho') = \langle q_{k+1}, \mathfrak{m}(f_S(\rho')), \tau(f_S(\rho')) \rangle$. It is now not hard to see that any outcome of the game when SYS plays with f_{SYS}, is such that the run component $r_\mathcal{D}$ is a run of $\mathcal{D}$ over the noisy computation $\mathcal{T}_{\mathfrak{m}}(w_I)$, where $w_I = i_0, i_1, i_2, \ldots \in (2^I)$ is obtained by f_I. Hence, since $\mathcal{T}$ realizes $\mathcal{D}$, it follows that $r_\mathcal{D}$ is accepting. That is, any outcome of the game when SYS plays with f_{SYS} is winning for SYS, and f_{SYS} is a winning strategy for SYS. ◀

B.4 Proof of Theorem 12

We start with the upper bound. Given an LTL[$\mathcal{F}$] specification φ, a predicate $P \subseteq [0,1]$, and an LTL secret ψ, we construct the DPW $\mathcal{D} = \mathcal{D}^P_{\varphi,\psi}$ as in Theorem 8, and then solve the game $\mathcal{G}_\mathcal{D}$. By Theorem 10 and Proposition 11, it follows that $\langle \varphi, P \rangle$ is realizable with privacy ψ iff SYS wins $\mathcal{G}_\mathcal{D}$, and that solving $\mathcal{G}_\mathcal{D}$ is done in time $O(n^k)$ where n is the number of positions in $\mathcal{G}_\mathcal{D}$ and k is the index of $\mathcal{D}$. By Theorem 8, the number of states in $\mathcal{D}$ is $|Q| = 2^{2^{O(|\varphi|+|\psi|)}}$, and the index is of size $k = 2^{O(|\varphi|+|\psi|)}$, and in particular, the construction of $\mathcal{D}$ is done in 2EXPTIME in the size of the formulas φ and ψ. The number of positions in $\mathcal{G}_\mathcal{D}$ is $|V| \leq |Q| \cdot 3^{|I|+|O|} = 2^{2^{O(|\varphi|+|\psi|)}} \cdot 3^{|I|+|O|}$, and the number of priorities is the same as in $\mathcal{D}$. We may assume that $I \cup O \subseteq \mathit{cl}(\varphi) \cup \mathit{cl}(\psi)$, hence $3^{|I|+|O|} = 2^{O(|\varphi|+|\psi|)}$, and

$|V| = 2^{2^{O(|\varphi|+|\psi|)}}$. Thus, $\mathcal{G}_{\mathcal{D}}$ is solved in time,

$$n^k \leq (2^{2^{O(|\varphi|+|\psi|)}})^{2^{O(|\varphi|+|\psi|)}} = 2^{2^{O(|\varphi|+|\psi|)}}$$

That is, $\mathcal{G}_{\mathcal{D}}$ is solved in 2EXPTIME in the size of φ and ψ.

For the lower bound, it is easy to reduce LTL[$\mathcal{F}$] synthesis with no privacy requirements to LTL[$\mathcal{F}$] synthesis with such requirements, for example by adding a secret that refers to a dummy output signal $p \notin I \cup O$.

B.5 Proof of Proposition 13

We prove that if $L(\mathcal{U}') = \varnothing$ then $\mathcal{U}$ is not realizable by a noisy I/O-transducer, and that if $L(\mathcal{U}') \neq \varnothing$, then there is a finite witness for the nonemptiness of $\mathcal{U}'$ that encodes a noisy transducer that realizes $\mathcal{U}$.

Given a $(2^I \times 3^O)$-labeled 3^I-tree $\langle (3^I)^*, f \rangle$ and an input word $w_I = i_0, i_1, i_2, \ldots \in (2^I)^\omega$, we define the sequence of masking instructions $M_0, M_1, M_2, \ldots \in (2^I)^\omega$, the sequence of noisy output assignments $o_0, o_1, o_2, \ldots \in (3^O)^\omega$, and the masked input word $w'_I = i'_0, i'_1, i'_2, \ldots \in (3^I)^\omega$ that correspond to f and w_I as follows. First, $\langle M_0, o_0 \rangle = f(\varepsilon)$. Then, for all $k \geq 0$, we have that $i'_k = hide(M_k, i_k)$ and $\langle M_{k+1}, o_{k+1} \rangle = f(i'_0, i'_1, \ldots, i'_k)$. Then, let $\kappa = (i'_0 \cup o_0), (i'_1 \cup o_1), \ldots \in (3)^{I \cup O}$ be the noisy computation that correspond to f and w_I. Observe that f is accepted by $\mathcal{U}'$ iff for all $w_I \in (2^I)^\omega$, the noisy computation κ that corresponds to f and w_I is accepted by $\mathcal{U}$. Thus, f can be thought as a strategy for the noisy synthesis of $\mathcal{U}$, and f is accepted by $\mathcal{U}'$ iff it is a winning strategy.

Note that the language of $\mathcal{U}'$ is not empty iff there is a finite memory strategy $f : (3^I)^* \to 2^I \times 3^O$ that is accepted by $\mathcal{A}'$, and the memory structure of f is at most exponential in the size of $\mathcal{A}'$ [31]. Hence, the specification given by $\mathcal{A}$ is realizable by a noisy I/O-transducer iff the language of $\mathcal{A}'$ is not empty, and a finite memory witness for the non-emptiness of $\mathcal{A}'$ is a noisy I/O-transducer that realizes $\mathcal{A}$. Deciding whether the language of a UGCT is empty, and finding a finite memory witness in the case it is not empty is in EXPTIME. Hence, the synthesis of a noisy transducer that realizes $\mathcal{A}$ is reduced to the nonemptiness of UGCT problem, and we have an EXPTIME upper bound.