Latticed-LTL Synthesis in the Presence of Noisy Inputs

Shaull Almagor and Orna Kupferman

The Hebrew University, Jerusalem, Israel

Abstract. In the classical synthesis problem, we are given a linear temporal logic (LTL) formula ψ over sets of input and output signals, and we synthesize a finite-state transducer that realizes ψ: with every sequence of input signals, the transducer associates a sequence of output signals so that the generated computation satisfies ψ. In recent years, researchers consider extensions of the classical Boolean setting to a multi-valued one. We study a setting in which the truth values of the input and output signals are taken from a finite lattice, and the specification formalism is Latticed-LTL (LLTL), where conjunctions and disjunctions correspond to the meet and join operators of the lattice, respectively. The lattice setting arises in practice, for example in specifications involving priorities or in systems with inconsistent viewpoints.

We solve the LLTL synthesis problem, where the goal is to synthesize a transducer that realizes ψ in desired truth values.

For the classical synthesis problem, researchers have studied a setting with incomplete information, where the truth values of some of the input signals are hidden and the transducer should nevertheless realize ψ. For the multi-valued setting, we introduce and study a new type of incomplete information, where the truth values of some of the input signals may be noisy, and the transducer should still realize ψ in a desired value. We study the problem of noisy LLTL synthesis, as well as the theoretical aspects of the setting, like the amount of noise a transducer may tolerate, or the effect of perturbing input signals on the satisfaction value of a specification.

1 Introduction

Synthesis is the automated construction of a system from its specification. The basic idea is simple and appealing: instead of developing a system and verifying that it adheres to its specification, we would like to have an automated procedure that, given a specification, constructs a system that is correct by construction. The first formulation of synthesis goes back to Church [10]. The modern approach to synthesis was initiated by Pnueli and Rosner, who introduced LTL (linear temporal logic) synthesis [24]: We are given an LTL formula ψ over sets I and O of input and output signals, and we synthesize a finite-state system that *realizes* ψ. At each moment in time, the system 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 ψ if all the computations that are generated by the interaction satisfy ψ. Synthesis has attracted a lot of research and interest [28].

In recent years, researchers have considered extensions of the classical Boolean setting to a multi-valued one, where the atomic propositions are multi-valued, and so is

A. Muscholl (Ed.): FOSSACS 2014, LNCS 8412, pp. 226–241, 2014.

the satisfaction value of specifications. The multi-valued setting arises directly in systems in which the designer can give to the atomic propositions rich values, expressing, for example, energy consumption, waiting time, or different levels of confidence [5,1], and arises indirectly in probabilistic settings, systems with multiple and inconsistent view-points, specifications with priorities, and more [20,14,2]. Adjusting the synthesis problem to this setting, one works with multi-valued specification formalisms. In such formalisms, a specification ψ maps computations in which the atomic propositions take values from a domain D to a satisfaction value in D. For example, ψ may map a computation in $(\{0,1,2,3\}^{\{p\}})^\omega$ to the maximal value assigned to the (multi-valued) atomic proposition p during the computation. Accordingly, the synthesis problem in the multi-valued setting gets as input a specification ψ and a predicate $P \subseteq D$ of desired values, and seeks a system that reads assignments in D^I, responds with assignments in D^O, and generates only computations whose satisfaction value is in P. The synthesis problem has been solved for several multi-valued settings [4,1].

A different extension of the classical setting of synthesis considers settings in which the system has *incomplete information* about its environment. Early work on incomplete information considers settings in which the system can read only a subset of the signals in I and should still generate only computations that satisfy the specification, which refers to all the signals in $I \cup O$ [18,6,7]. The setting is equivalent to a game with incomplete information, extensively studied in [25]. As shown there, the common practice in handling incomplete information is to move to an exponentially-larger game of complete information, where each state corresponds to a set of states that are indistinguishable by a player with incomplete information in the original game.

More recent work on synthesis with incomplete information studies richer types of incomplete information. In [8], the authors study a setting in which the transducer can read some of the input signals some of the time. In more detail, *sensing* the truth value of an input signal has a cost, the system has a budget for sensing, and it tries to realize the specification while minimizing the required sensing budget. In [30], the authors study games with *errors*. Such games correspond to a synthesis setting in which there are positions during the interaction in which input signals are read by the system with an error. The games are characterized by the number or rate of errors that the system has to cope with, and by the ability of the system to detect whether a current input is erred.

In this work we introduce and study a different model of incomplete information in the multi-valued setting. In our model, the system always reads all input signals, but their value may be perturbed according to a known noise function. This setting naturally models incomplete information in real-life multi-valued settings. For example, when the input is read by sensors that are not accurate (e.g., due to bounded precision, or to probabilistic measuring) or when the input is received over a noisy channel and may come with some distortion. The multi-valued setting we consider is that of *finite lattices*. A lattice is a partially-ordered set $\mathcal{L} = \langle A, \leq \rangle$ in which every two elements ℓ and ℓ' have a least upper bound (ℓ *join* ℓ', denoted $\ell \vee \ell'$) and a greatest lower bound (ℓ *meet* ℓ', denoted $\ell \wedge \ell'$). Of special interest are two classes of lattices: (1) Fully ordered, where $\mathcal{L} = \langle \{1, \ldots, n\}, \leq \rangle$, for an integer $n \geq 1$ and the usual "less than or equal" order. In this lattice, the operators $\vee$ and $\wedge$ correspond to $\max$ and $\min$, respectively.

(2) Power-set lattices, where $\mathcal{L} = \langle 2^X, \subseteq \rangle$, for a finite set X, and the containment (partial) order. In this lattice, the operators $\vee$ and $\wedge$ correspond to $\cup$ and $\cap$, respectively.

The lattice setting is a good starting point to the multi-valued setting. While their finiteness circumvents the infinite-state space of dense multi-values, lattices are sufficiently rich to capture many quantitative settings. Fully-ordered lattices are sometimes useful as is (for example, when modeling priorities [2]), and sometimes thanks to the fact that real values can often be abstracted to finitely many linearly ordered classes. The power-set lattice models a wide range of partially-ordered values. For example, in a setting with inconsistent viewpoints, we have a set X of agents, each with a different viewpoint of the system, and the truth value of a signal or a formula indicates the set of agents according to whose viewpoint the signal or the formula are true. As another example, in a peer-to-peer network, one can refer to the different attributes of the communication channels by assigning with them subsets of attributes. From a technical point of view, the fact that lattices are partially ordered poses challenges that do not exist in (finite and infinite) full orders. For example, as we are going to see, the fact that a specification is realizable with value ℓ and with value ℓ' does not imply it is realizable with value $\ell \vee \ell'$, which trivially holds for full orders.

We start by defining lattices and the logic *Latticed LTL* (LLTL, for short). We then study theoretical properties of LLTL: We study cases where the set of attainable truth values of an LLTL formula are closed under $\vee$, thus a maximal attainable value exists, even when the lattice elements are partially ordered. We also study stability properties, namely the affect of perturbing the values of the atomic propositions on the satisfaction value of formulas. We continue to the synthesis and the noisy-synthesis problems for LLTL, which we solve via a translation of LLTL formulas to Boolean automata. We show that by working with universal automata, we can handle the exponential blow-up that incomplete information involves together with the exponential blow-up that determination (or alternation removal, if we take a Safraless approach) involves, thus the noisy-synthesis problem stays 2EXPTIME-complete, as it is for LTL.

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

2 Preliminaries

2.1 Lattices

Consider a set A, a partial order $\leq$ on A, and a subset P of A. An element $\ell \in A$ is an *upper bound* on P if $\ell \geq \ell'$ for all $\ell' \in P$. Dually, ℓ is a *lower bound* on P if $\ell \leq \ell'$ for all $\ell' \in P$. The pair $\langle A, \leq \rangle$ is a *lattice* if for every two elements $\ell, \ell' \in A$, both the least upper bound and the greatest lower bound of $\{\ell, \ell'\}$ exist, in which case they are denoted $\ell \vee \ell'$ (ℓ *join* ℓ') and $\ell \wedge \ell'$ (ℓ *meet* ℓ'), respectively. We use $\ell < \ell'$ to indicate that $\ell \leq \ell'$ and $\ell \neq \ell'$. We say that ℓ is a *child* of ℓ', denoted $\ell \prec \ell'$, if $\ell < \ell'$ and there is no ℓ'' such that $\ell < \ell'' < \ell'$.

A lattice $\mathcal{L} = \langle A, \leq \rangle$ is *finite* if A is finite. Note that finite lattices are *complete*: every subset of A has a least-upper and a greatest-lower bound. We use $\top$ (*top*) and $\bot$ (*bottom*) to denote the least-upper and greatest-lower bounds of A, respectively. A lattice is *distributive* if for every $\ell_1, \ell_2, \ell_3 \in A$, we have $\ell_1 \wedge (\ell_2 \vee \ell_3) = (\ell_1 \wedge \ell_2) \vee (\ell_1 \wedge$

$\ell_3)$ and $\ell_1 \vee (\ell_2 \wedge \ell_3) = (\ell_1 \vee \ell_2) \wedge (\ell_1 \vee \ell_3)$. The traditional disjunction and conjunction logic operators correspond to the join and meet lattice operators. In a general lattice, however, there is no natural counterpart to negation. A *De-Morgan* (or *quasi-Boolean*) lattice is a lattice in which every element a has a unique complement element $\neg\ell$ such that $\neg\neg\ell = \ell$, De-Morgan rules hold, and $\ell \leq \ell'$ implies $\neg\ell' \leq \neg\ell$. In the rest of this paper we consider only finite distributive De-Morgan lattices. We focus on two classes of such lattices: (1) Fully ordered, where $\mathcal{L} = \langle\{1, \ldots, n\}, \leq\rangle$, for an integer $n \geq 1$ and the usual "less than or equal" order. Note that in this lattice, the operators $\vee$ and $\wedge$ correspond to $\max$ and $\min$, respectively, and $\neg i = n - i + 1$. (2) Power-set lattices, where $\mathcal{L} = \langle 2^X, \subseteq\rangle$, for a finite set X, and the containment (partial) order. Note that in this lattice, the operators $\vee$ and $\wedge$ correspond to $\cup$ and $\cap$, respectively, and negation corresponds to complementation.

Consider a lattice $\mathcal{L} = \langle A, \leq\rangle$. A *join irreducible* element in $\mathcal{L}$ is $l \in A$ such that $l \neq \bot$ and for all elements $l_1, l_2 \in A$, if $l_1 \vee l_2 \geq l$, then $l_1 \geq l$ or $l_2 \geq l$. For example, the join irreducible elements in $\langle 2^X, \subseteq\rangle$ are all singletons $\{x\}$, for $x \in X$. By *Birkhoff's representation theorem* for finite distributive lattices, in order to prove that $l_1 = l_2$, it is sufficient to prove that for every join irreducible element l it holds that $l_1 \geq l$ iff $l_2 \geq l$. We denote the set of join irreducible elements of $\mathcal{L}$ by $\mathrm{JI}(\mathcal{L})$. For convenience, we often talk about a lattice $\mathcal{L}$ without specifying A and $\leq$. We then abuse notations and refer to $\mathcal{L}$ as a set of elements and talk about $l \in \mathcal{L}$ or about assignments in $\mathcal{L}^{AP}$ (rather than $l \in A$ or assignments in A^{AP}).

2.2 The Logic LLTL

The logic LLTL is a natural generalization of LTL to a multi-valued setting, where the atomic propositions take values from a lattice $\mathcal{L}$ [9,16]. Given a (finite distributive De-Morgan) lattice $\mathcal{L}$, the syntax of LLTL is given by the following grammar, where p ranges over a set AP of atomic propositions, and ℓ ranges over $\mathcal{L}$.

$$\varphi := \ell \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi.$$

The semantics of LLTL is defined with respect to a *computation* $\pi = \pi_0, \pi_1, \ldots \in (\mathcal{L}^{AP})^\omega$. Thus, in each moment in time the atomic propositions get values from $\mathcal{L}$. Note that classical LTL coincides with LLTL defined with respect to the two-element fully-ordered lattice. For a position $i \geq 0$, we use π^i to denote the suffix $\pi_i, \pi_{i+1}, \ldots$ of π. Given a computation π and an LLTL formula φ, the *satisfaction value* of φ in π, denoted $[\![\pi, \varphi]\!]$, is defined by induction on the structure of φ as follows (the operators on the right-hand side are the join, meet, and complementation operators of $\mathcal{L}$). [1]

- $[\![\pi, \ell]\!] = \ell$.
- $[\![\pi, p]\!] = \pi_0(p)$.
- $[\![\pi, \neg\varphi]\!] = \neg[\![\pi, \varphi]\!]$.
- $[\![\pi, \varphi \vee \psi]\!] = [\![\pi, \varphi]\!] \vee [\![\pi, \psi]\!]$.
- $[\![\pi, \mathsf{X}\varphi]\!] = [\![\pi^1, \varphi]\!]$.
- $[\![\pi, \varphi \mathsf{U} \psi]\!] = \bigvee_{i \geq 0} ([\![\pi^i, \psi]\!] \wedge \bigwedge_{0 \leq j < i} [\![\pi^j, \varphi]\!])$.

Example 1. Consider a setting in which three agents a, b, and c have different viewpoints on a system $\mathcal{S}$. A truth assignment for the atomic propositions is then a function

[1] Unlike LTL, where the constants `True` and `False` do not increase the expressive power, in LLTL the constants $\ell \in \mathcal{L}$ do increase the expressive power.

in $(2^{\{a,b,c\}})^{AP}$ assigning to each $p \in AP$ the set of agents according to whose viewpoint p is true. We reason about $\mathcal{S}$ using the lattice $\mathcal{L} = \langle 2^{\{a,b,c\}}, \subseteq \rangle$. For example, the truth value of the formula $\psi = G(req \rightarrow Fgrant)$ in a computation is the set of agents according to whose view-point, whenever a request is sent, it is eventually granted.

2.3 LLTL Synthesis

Consider a lattice $\mathcal{L}$ and finite disjoint sets I and O of input and output signals that take values in $\mathcal{L}$. An (I/O)-*transducer over* $\mathcal{L}$ models an interaction between an environment that generates in each moment in time an input in $\mathcal{L}^I$ and a system that responds with outputs in L^O. Formally, an (I/O)-transducer over $\mathcal{L}$ (*transducer*, when I, O, and $\mathcal{L}$ are clear from the context) is a tuple $\mathcal{T} = \langle \mathcal{L}, I, O, S, s_0, \eta, \tau \rangle$ where S is a finite set of states, $s_0 \in S$ is an initial state, $\eta : S \times \mathcal{L}^I \rightarrow S$ is a deterministic transition function, and $\tau : S \rightarrow \mathcal{L}^O$ is a labeling function. We extend η to words in $(\mathcal{L}^I)^*$ in the straightforward way. Thus, $\eta : (\mathcal{L}^I)^* \rightarrow S$ is such that $\eta(\epsilon) = s_0$, and for $x \in (\mathcal{L}^I)^*$ and $i \in \mathcal{L}^I$, we have $\eta(x \cdot i) = \eta(\eta(x), i)$. Each transducer $\mathcal{T}$ induces a *strategy* $f_{\mathcal{T}} : (\mathcal{L}^I)^* \rightarrow \mathcal{L}^O$ where for all $w \in (\mathcal{L}^I)^*$, we have $f_{\mathcal{T}}(w) = \tau(\eta(w))$. Thus, $f_{\mathcal{T}}(w)$ is the letter that $\mathcal{T}$ outputs after reading the sequence w of input letters. Given a sequence $i_0, i_1, i_2, \ldots \in (\mathcal{L}^I)^\omega$ of input assignments, the transducer generates the computation $\rho = (i_0 \cup o_0), (i_1 \cup o_1), (i_2 \cup o_2), \ldots \in (\mathcal{L}^{I \cup O})^\omega$, where for all $j \geq 1$, we have $o_j = f_{\mathcal{T}}(i_0 \cdots i_{j-1})$.

Consider a lattice $\mathcal{L}$, an LLTL formula φ over the atomic propositions $I \cup O$, and a predicate $P \subseteq \mathcal{L}$. We say that a transducer $\mathcal{T}$ *realizes* $\langle \varphi, P \rangle$ if for every computation ρ of $\mathcal{T}$, it holds that $[\![\rho, \varphi]\!] \in P$. The *realizability* problem for LLTL is to determine, given φ and P, whether there exists a transducer that realizes $\langle \varphi, P \rangle$. We then say that φ is (I/O)-realizable with values in P. The *synthesis* problem is then to generate such a transducer. Of special interest are predicates P that are upward closed. Thus, P is such that for all $\ell \in \mathcal{L}$, if $\ell \in P$ then $\ell' \in P$ for all $\ell' \geq \ell$.

2.4 Noisy Synthesis

Consider an LLTL formula φ over atomic proposition $I \cup O$ and a predicate P. In *noisy synthesis*, we consider the synthesis problem in a setting in which the inputs are read with some perturbation and the goal is to synthesize a transducer that nevertheless realizes $\langle \varphi, P \rangle$.

In order to formalize the above intuition, we first formalize the notion of noise. Consider a lattice $\mathcal{L} = \langle A, \leq \rangle$ and two elements $\ell_1, \ell_2 \in \mathcal{L}$. We define the *distance* between ℓ_1 and ℓ_2, denoted $d(\ell_1, \ell_2)$, as the shortest path from ℓ_1 to ℓ_2 in the undirected graph $\langle A, E_\prec \rangle$ in which $E_\prec(v, v')$ iff $v \prec v'$ or $v' \prec v$. For example, in the fully-ordered lattice $\mathcal{L}$, we have $d(i, j) = |i - j|$, and in the power-set lattice, the distance coincides with the Hamming distance, thus $d(X_1, X_2) = |(X_1 \setminus X_2) \cup (X_2 \setminus X_1)|$. For two assignments $f, f' \in \mathcal{L}^{AP}$, we define $d(f, f') = \max_{p \in AP} d(f(p), f'(p))$.

We assume we are given a *noise function* $\nu : \mathcal{L}^I \rightarrow 2^{\mathcal{L}^I}$, describing the possible perturbations of each input. That is, for every $i \in \mathcal{L}^I$ the set $\nu(i)$ consists of the inputs that may have been actually generated by the environment, when the system reads i. A natural noise function is $\nu(i) = \{j : d(i, j) \leq \gamma\}$, for some constant γ, which is the

γ-units ball around i. Given a noise function ν and two computations $\pi, \pi' \in (\mathcal{L}^{I \cup O})^\omega$, we say that π' is *ν-indistinguishable* from π if for every $i \geq 0$, we have that $\pi'_i|_I \in \nu(\pi_i|_I)$ and $\pi'_i|_O = \pi_i|_O$, where $\sigma|_I$ is the restriction of $\sigma \in \mathcal{L}^{I \cup O}$ to inputs in I, and similarly for $\sigma|_O$ and O. Thus, π' is obtained from π by changing only the assignment to input signals, within ν. Note that ν need not be a symmetric function, nor is the definition of ν-indistinguishablity. We say that a transducer $\mathcal{T}$ *realizes* $\langle \varphi, P \rangle$ *with noise* ν if for every computation π of $\mathcal{T}$, we have that $[\![\pi', \varphi]\!] \in P$ for all computations π' that are ν-indistinguishable from π. Thus, the reaction of $\mathcal{T}$ on every input sequence satisfies φ in a desired satisfaction value even if the input sequence is read with noise ν.

2.5 Automata and Games

An *automaton over infinite words* is $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, where Σ is the input alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, and α is an acceptance condition. When $\mathcal{A}$ is a *generalized Büchi* or a *generalized co-Büchi* automaton, then $\alpha \subseteq 2^Q$ is a set of sets of accepting states. When $\mathcal{A}$ is a *parity* automaton, then $\alpha = \langle F_1, \ldots, F_d \rangle$, where the sets in α form a partition of Q. The number of sets in α is the *index* of $\mathcal{A}$. An automaton is *deterministic* if $|Q_0| = 1$ and for every $q \in Q$ and $\sigma \in \Sigma$, we have that $|\delta(q, \sigma)| = 1$. A run $r = r_0, r_1, \ldots$ of $\mathcal{A}$ on a word $w = w_1 \cdot w_2 \cdots \in \Sigma^\omega$ is an infinite sequence of states such that $r_0 \in Q_0$, and for every $i \geq 0$, we have that $r_{i+1} \in \delta(r_i, w_{i+1})$. We denote by $\inf(r)$ the set of states that r visits infinitely often, that is $\inf(r) = \{q : r_i = q \text{ for infinitely many } i \in \mathbb{N}\}$. The run r is *accepting* if it satisfies α. For generalized Büchi automata, a run is accepting if it visits all the sets in α infinitely often. Formally, for every set $F \in \alpha$, we have that $\inf(r) \cap F \neq \emptyset$. Dually, in generalized co-Büchi automata, there should exist a set $F \in \alpha$ for which $\inf(r) \cap F = \emptyset$. For parity automata, a run r is accepting if the minimal index i for which $\inf(r) \cap F_i \neq \emptyset$ is even.

When $\mathcal{A}$ is a *nondeterministic* automaton, it accepts a word w if it has an accepting run of on w. When $\mathcal{A}$ is a *universal* automaton, it accepts a word w if all its runs on w are accepting. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts.

A *parity game* is $\mathcal{G} = \langle \Sigma_1, \Sigma_2, S, s_0, \delta, \alpha \rangle$, where Σ_1 and Σ_2 are alphabets for Players 1 and 2, respectively, S is a finite set of states, $s_0 \in S$ is an initial state, $\delta : S \times \Sigma_1 \times \Sigma_2 \to S$ is a transition function, and $\alpha = \langle F_1, \ldots, F_d \rangle$ is a parity acceptance condition, as described above. A play of the game starts in s_0. In each turn Player 1 chooses a letter $\sigma \in \Sigma_1$ and Player 2 chooses a letter $\tau \in \Sigma_2$. The play then moves from the current state s to the state $\delta(s, \sigma, \tau)$. Formally, a *play* of $\mathcal{G}$ is an infinite sequence $\rho = \langle s_0, \sigma_0, \tau_0 \rangle, \langle s_1, \sigma_1, \tau_1 \rangle, \ldots$ such that for every $i \geq 0$, we have that $s_{i+1} = \delta(s_i, \sigma_i, \tau_i)$. We define $\inf(\rho) = \{s \in S : s = s_i \text{ for infinitely many } i \in \mathbb{N}\}$. A play ρ is *winning for Player 1* if the minimal index i for which $\inf(\rho) \cap F_i \neq \emptyset$ is even. A *strategy* for Player 1 is a function $f : (S \times \Sigma_1 \times \Sigma_2)^* \times S \to \Sigma_1$ that assigns, for every finite prefix of a play, the next move for Player 1. Similarly, a strategy for Player 2 is a function $g : (S \times \Sigma_1 \times \Sigma_2)^* \times S \times \Sigma_1 \to \Sigma_2$. A strategy is *memoryless* if it does not depend on the history of the play. Thus, a memoryless strategy for Player 1 is a function $f : S \to \Sigma_1$ and for Player 2 it is a function $g : S \times \Sigma_1 \to \Sigma_2$.

A pair of strategies f, g for Players 1 and 2, respectively, induces a single play that conforms with the strategies. We say that Player 1 wins $\mathcal{G}$ if there exists a strategy f

for Player 1 such that for every strategy g for Player 2, the play induced by f and g is winning for Player 1. Otherwise, Player 2 wins. By determinancy of Parity games [21], Player 2 wins $\mathcal{G}$ if there exists a strategy g for Player 2 such that for every strategy f of Player 1, the play induced by f and g is not winning for Player 1.

2.6 Solving the Boolean Synthesis Problem

The classical solution for the synthesis problem for LTL goes via games [24].[2] It involves a translation of the specification into a deterministic parity word automaton (DPW) over the alphabet $2^{I \cup O}$, which is then transformed into a game in which the players alphabets are 2^I and 2^O. More recent solutions avoids the determination and the solution of parity games and use instead alternating tree automata [19,13]. The complexity of both approaches coincide. Below we describe the classical solution for the synthesis problem, along with its complexity, when the starting point is a specification given by a DPW.[3] In Remark 1, we describe an alternative, Safraless, approach, where the starting point is a universal co-Büchi automaton.

Theorem 1. *Consider a specification φ over I and O given by means of a DPW $\mathcal{D}_\varphi$ of size t over the alphabet $2^{I \cup O}$, with index k. The synthesis problem for φ can be solved in time $O(t^k)$.*

Proof. Let $\mathcal{D}_\varphi = \langle 2^{I \cup O}, Q, q_0, \delta, \alpha \rangle$. We define a game $\mathcal{G}_\varphi$ that models an interaction that simulates $\mathcal{D}_\varphi$ between a system (Player 1) that generates assignments in 2^O and an environment (Player 2) that generates assignments in 2^I. Formally, $\mathcal{G}_\varphi = \langle 2^O, 2^I, Q, q_0, \eta, \alpha \rangle$, where $\eta : Q \times 2^O \times 2^I \to Q$ is such that for every $q \in Q$, $i \in 2^I$, and $o \in 2^O$, we have that $\eta(q, i, o) = \delta(q, i \cup o)$. By [11], the game is determined and one of the players has a memoryless winning strategy. Such a strategy for Player 1 in $\mathcal{G}_\varphi$ is then a transducer that realizes φ. The game $\mathcal{G}_\varphi$ is of size $O(t)$ and index k. Hence, by [15,27], we can find a memoryless strategy for the winner in time $O(t^k)$. □

3 Properties of LLTL

In this section we study properties of the logic LLTL. We focus on the set of attainable satisfaction values of an LLTL formula and on stability properties, namely the affect of perturbing the values of the atomic propositions on the satisfaction value of formulas.

3.1 Attainable Values

Consider a lattice $\mathcal{L}$. We say that $\mathcal{L}$ is *pointed* if for all LLTL formulas φ, partitions $I \cup O$ of AP, and values $\ell_1, \ell_2 \in \mathcal{L}$, if φ is (I/O)-realizable with value ℓ_1 and with value ℓ_2, then φ is also (I/O)-realizable with value $\ell_1 \vee \ell_2$. Observe that if $\mathcal{L}$ is pointed, then every LLTL formula over $\mathcal{L}$ has a transducer that realizes it with a maximal value.

[2] In [24] and other early works the games are formulated by means of tree automata.

[3] State-of-the-art algorithms for solving parity games achieve a better complexity [15,27]. The bound, however, remains polynomial in the size of the game and exponential in its index. Since the challenge of solving parity games is orthogonal to our contribution here, we keep this component of our contribution simple.

We start by showing that in general, not all lattices are pointed. In fact, our example has $O = \emptyset$, where (I/O)-realizability coincides with satisfiability. The lattices we focus on, are, however, pointed.

Theorem 2. *Not all distributive De-Morgan lattices are pointed. Fully-ordered lattices and power-set lattices are pointed.*

Proof. The proof of the positive result is in the full version. For the negative one, consider the lattice $\mathcal{L} = \langle 2^{\{a,b\}} \times \{0,1\}, \leq \rangle$ where $\langle S_1, v_1 \rangle \leq \langle S_2, v_2 \rangle$ iff $v_1 \leq v_2$ or ($v_1 = v_2$ and $S_1 \subseteq S_2$). We define $\neg\langle S, v\rangle = \langle \{a,b\} \setminus S, 1 - v\rangle$. It is easy to verify that $\mathcal{L}$ is a distributive De-Morgan lattice.

Let $I = \{p\}$ and consider the formula $\varphi = (p \wedge \langle\{a\}, 1\rangle) \vee (\neg p \wedge \langle\{b\}, 1\rangle)$. Both $\langle\{a\}, 1\rangle$ and $\langle\{b\}, 1\rangle$ are attainable satisfaction values of φ. For example, by setting p to $\langle\{a\}, 1\rangle$ or to $\langle\{a\}, 0\rangle$. On the other hand, for every assignment ℓ to p, the second component of either ℓ or $\neg\ell$ is 0. Consequently, $\langle\{a,b\}, 1\rangle$ is not attainable, thus $\mathcal{L}$ is not pointed. □

3.2 Stability

For two computations $\pi = \pi_0, \pi_1, \ldots$ and $\pi' = \pi'_0, \pi'_1, \ldots$, both in $(\mathcal{L}^{AP})^\omega$, we define the *global distance* between π and π', denoted $gd(\pi, \pi')$, as $\sum_{i \geq 0} d(\pi_i, \pi'_i)$. Note that $gd(\pi, \pi')$ may be infinite. We define the *local distance* between π and π', denoted $ld(\pi, \pi')$, as $\max_{i \geq 0} d(\pi_i, \pi'_i)$. Note that $ld(\pi, \pi') \leq |\mathcal{L}|$.

Consider an LLTL formula φ over AP and $\mathcal{L}$. We say that φ is *globally stable* if for every pair π and π' of computations, we have $d([\![\pi, \varphi]\!], [\![\pi', \varphi]\!]) \leq gd(\pi, \pi')$. Thus, the difference between the satisfaction value of φ in π and π' is bounded by the sum of differences between matching locations in π and π'. Also, φ is *locally stable* if for every pair π and π' of computations, we have $d([\![\pi, \varphi]\!], [\![\pi', \varphi]\!]) \leq ld(\pi, \pi')$. Thus, the difference between the satisfaction value of φ in π and π' is bounded by the maximal difference between matching locations in π and π'. Here, we study stability of all LLTL formulas. In Section 5.3, we study the problem of deciding whether a given LLTL formula is stable, and discuss the relevancy of stability to synthesis with noise.

Consider an LLTL formula φ over the atomic propositions AP, and consider computations $\pi, \pi' \in (\mathcal{L}^{AP})^\omega$. Assume that $gd(\pi, \pi') \leq 1$. That is, π and π' differ only in one location, where they differ in the value of a single atomic proposition, whose value in π is a child of its value in π' or vice versa. It is tempting to think that then, $d([\![\pi, \varphi]\!], [\![\pi', \varphi]\!]) \leq 1$, which would imply that φ should be globally stable.

We start by breaking this intuition, showing that for non-distributive lattices, this is false. The proof makes use of an *N5 structure*. Formally, an N5 structure in a lattice $\mathcal{L}$ is a tuple $\langle x, y, z, w, s\rangle$ such that the following relations hold: $s < x < y < w$, $s < z < w$, $y \not\leq z$, $z \not\leq y$, $x \not\leq z$, and $z \not\leq x$. Note that $x \vee (z \wedge y) = x \vee s = x$, whereas $(x \vee z) \wedge (x \vee y) = w \wedge y = y$. Hence, the structure of N5 is never a sub-lattice in a distributive lattice.

Theorem 3. LLTL *formulas may not be globally stable with respect to non-distributive lattices.*

Proof. Consider the lattice N5, the formula $\varphi = p \vee q$, and a computation π such that $\pi_0(p) = s$ and $\pi_0(q) = x$. Clearly $[\![\pi, \varphi]\!] = x$. Now, let π' be the computation obtained from π by setting $\pi'_0(p) = z$. It holds that $gd(\pi, \pi') = 1$. However, $[\![\pi', \varphi]\!] = z \vee x = w$, and $d(x, w) = 2$. Thus, φ is not globally stable over the lattice N5. □

We now proceed to show that when defined with respect to a distributive lattice, all LLTL formulas are globally stable.

Theorem 4. LLTL *formulas over De-Morgan distributive lattices are globally stable.*

Proof. We prove that for every LLTL formula φ and computations $\pi, \pi' \in (\mathcal{L}^{AP})^\omega$, if $gd(\pi, \pi') = 1$, then $d([\![\pi, \varphi]\!], [\![\pi', \varphi]\!]) \leq 1$. We then proceed by induction on $gd(\pi, \pi')$.

Consider an LLTL formula φ and computations π, π' such that $gd(\pi, \pi') = 1$. That is, there exists a single index $i \geq 0$ such that $d(\pi_i, \pi'_i) = 1$ and $\pi_j = \pi'_j$ for all $j \neq i$. W.l.o.g, there is $p \in AP$ such that $\pi_i(p) \preceq \pi'_i(p)$. By Birkhoff's representation theorem, there exists a unique element $u \in \mathrm{JI}(\mathcal{L})$ such that $\pi'_i(p) = \pi_i(p) \vee u$. We prove, by induction over the structure of φ, that $[\![\pi', \varphi]\!] \in \{[\![\pi, \varphi]\!] \wedge \neg u, [\![\pi, \varphi]\!], [\![\pi, \varphi]\!] \vee u\}$ and that $d([\![\pi', \varphi]\!], [\![\pi, \varphi]\!]) \leq 1$.

The proof appears in the full version. As detailed there, the interesting case is when $\varphi = \psi \vee \theta$, where we use the fact that a distributed lattice cannot have an N5 structure. □

We now turn to study local stability. Since local stability refers to the maximal change along a computation, it is a very permissive notion. In particular, it is not hard to see that in a fully-ordered lattice, a local change of 1 entails a change of at most 1 in the satisfaction value. Thus, we have the following.

Theorem 5. LLTL *formulas are locally stable with respect to fully-ordered lattices.*

In partially-ordered lattices, however, things are more involved, as local changes may be in different "directions". Formally, we have the following.

Theorem 6. LLTL *formulas may not be locally stable.*

Proof. Consider the power-set lattice $\langle 2^{a,b}, \subseteq \rangle$ and the LLTL formula $\varphi = p \vee \mathsf{X}p$. Consider computations π and π' with $\pi_0(p) = \pi_1(p) = \emptyset$, $\pi'_0(p) = \{a\}$, and $\pi'_1(p) = \{b\}$. It holds that $ld(\pi, \pi') = 1$, whereas $d([\![\pi, \varphi]\!], [\![\pi', \varphi]\!]) = d(\emptyset, \{a, b\}) = 2$. We conclude that φ is not locally stable. □

4 Translating LLTL to Automata

In this section we describe an automata-theoretic approach for reasoning about LLTL specifications. One approach is to develop a framework that is based on *lattice automata* [16]. Like LLTL formulas, lattice automata map words to values in a lattice. Lattice automata have proven to be useful in solving the satisfiability and the model-checking problems for LLTL [16]. However, the solution of the synthesis problem involves automata-theoretic constructions for which the latticed counterpart is either not known or is very complicated. In particular, Safra's determinization construction has not yet been studied for lattice automata, and a latticed counterpart of it is not going

to be of much fun. Likewise, the solution of two-player games (even reachability, and moreover parity) in the latticed setting is much more complicated than in the Boolean setting. In particular, obtaining a value $\ell_1 \vee \ell_2$ in a latticed game may require one strategy for obtaining ℓ_1 and a different strategy for obtaining ℓ_2 [17]. When the game is induced by a realizability problem, it is not clear how to combine such strategies into a single transducer that realizes the underlying specification with value $\ell_1 \vee \ell_2$.

Accordingly, a second approach, which is the one we follow, is to use Boolean automata. The fact LLTL formulas have finitely many possible satisfaction values suggests that this is possible. For fully-ordered lattices, a similar approach has been taken in [12,1]. Beyond the challenge in these works of maintaining the simplicity of the automata-theoretic framework of LTL, an extra challenge in the latticed setting is caused by the fact values may be only partially ordered. We will elaborate on this point below.

In order to explain our framework, let us recall first the translation of LTL formulas to nondeterministic generalized Büchi automata (NGBW), as introduced in [29]. There, each state of the automaton is associated with a set of formulas, and the NGBW accepts a computation from a state q iff the computation satisfies exactly all the formulas associated with q. The state space of the NGBW contains only states associated with maximal and consistent sets of formulas, the transitions are defined so that requirements imposed by temporal formulas are satisfied, and the acceptance condition is used in order to guarantee that requirements that involve the satisfaction of eventualities are not delayed forever.

In the construction here, each state of the NGBW assigns a satisfaction value to every subformula. While it is not difficult to extend the local consistency rules to the latticed settings, handling of eventualities is more complicated. To see why, consider for example the formula Fp, for $p \in AP$, and the computation π in which the satisfaction value of p is $(\{a\},\{b\},\{c\})^\omega$. While $[\![\pi, Fp]\!] = \{a, b, c\}$, the computation never reaches a position in which the satisfaction value of the eventuality p is $\{a, b, c\}$. This poses a problem on translations of LTL formulas to automata, where eventualities are handed by making sure that each state in which the satisfaction of $\psi_1 \mathsf{U} \psi_2$ is guaranteed, is followed by a state in which the satisfaction of ψ_2 is guaranteed. For a multi-valued setting with fully-ordered values, as is the case in [12,1], the latter can be replaced by a requirement to visit a state in which the guaranteed satisfaction value of ψ exceeds that of $\psi_1 \mathsf{U} \psi_2$. As the example above demonstrates, such a position need not exist when the values are partially ordered. In order to address the above problem, every state in the NGBW associates with every subformula of the form $\psi_1 \mathsf{U} \psi_2$ a value in $\mathcal{L}$ that ψ_2 still needs "accumulate" in order for $\psi_1 \mathsf{U} \psi_2$ to have its assigned satisfaction value. Thus, as in other break-point constructions [29,22], we decompose the requirement to obtain a value ℓ to requirements to obtain join-irreducible values whose join is ℓ, and we check these requirements together.

Theorem 7. *Let φ be an* LLTL *formula over $\mathcal{L}$ and $P \subseteq \mathcal{L}$ be a predicate. There exists an NGBW $\mathcal{A}_{\varphi,P}$ such that for every computation $\pi \in (2^{AP})^\omega$, it holds that $[\![\pi, \varphi]\!] \in P$ iff $\mathcal{A}_{\varphi,P}$ accepts π. The state space and transitions of $\mathcal{A}_{\varphi,P}$ are independent of P, which only influences the set of initial states. The NGBW $\mathcal{A}_{\varphi,P}$ has at most $|\mathcal{L}|^{O(|\varphi|)}$ states and index at most $|\varphi|$.*

Proof. We define $\mathcal{A}_{\varphi,P} = \langle \mathcal{L}^{AP}, Q, \delta, Q_0, \alpha \rangle$ as follows. Let $cl(\varphi)$ be the set of φ's subformulas, and let $ucl(\varphi)$ be the set of φ's subformulas of the form $\psi_1 \mathsf{U} \psi_2$. Let G_φ and F_φ be the collection of functions $g : cl(\varphi) \to \mathcal{L}$ and $f : ucl(\varphi) \to \mathcal{L}$, respectively. For an element $v \in \mathcal{L}$, let $\mathrm{JI}(v)$ be the minimal set $S \subseteq \mathrm{JI}(\mathcal{L})$ such that $v = \bigvee_{s \in S} s$. By Birkhoff's theorem, this set is well defined, and the JI mapping is a bijection.

For a pair of functions $\langle g, f \rangle \in G_\varphi \times F_\varphi$, we say that $\langle g, f \rangle$ is *consistent* if for every $\psi \in cl(\varphi)$, the following holds.

- If $\psi = v \in \mathcal{L}$, then $g(\psi) = v$.
- If $\psi = \neg\psi_1$, then $g(\psi) = \neg g(\psi_1)$.
- If $\psi = \psi_1 \vee \psi_2$, then $g(\psi) = g(\psi_1) \vee g(\psi_2)$.
- If $\psi = \psi_1 \mathsf{U} \psi_2$, then $\mathrm{JI}(f(\psi)) \cap \mathrm{JI}(g(\psi_2)) = \emptyset$.

The state space Q of $\mathcal{A}_{\varphi,\ell}$ is the set of all consistent pairs of functions in $G_\varphi \times F_\varphi$. Intuitively, while the function g describes the satisfaction value of the formulas in the closure, the function f describes, for each subformula of the form $\psi_1 \mathsf{U} \psi_2$, the values in which ψ_2 still has to be satisfied in order for the satisfaction value $g(\psi_1 \mathsf{U} \psi_2)$ to be fulfilled. Accordingly, if a value is in $\mathrm{JI}(g(\psi_2))$, it can be removed from $f(\psi_1 \mathsf{U} \psi_2)$, explaining why $\mathrm{JI}(f(\psi_1 \mathsf{U} \psi_2)) \cap \mathrm{JI}(g(\psi_2)) = \emptyset$.

Then, $Q_0 = \{g \in Q : \ g(\varphi) \in P\}$ contains all states in which the value assigned to φ is in P.

We now define the transition function δ. For two states $\langle g, f \rangle$ and $\langle g', f' \rangle$ in Q and a letter $\sigma \in \mathcal{L}^{AP}$, we have that $\langle g', f' \rangle \in \delta(\langle g, f \rangle, \sigma)$ iff the following hold.

- For all $p \in AP$, we have that $\sigma(p) = g(p)$.
- For all $\mathsf{X}\psi_1 \in cl(\varphi)$, we have $g(\mathsf{X}\psi_1) = g'(\psi_1)$.
- For all $\psi_1 \mathsf{U} \psi_2 \in cl(\varphi)$, we have $g(\psi_1 \mathsf{U} \psi_2) = g(\psi_2) \vee (g(\psi_1) \wedge g'(\psi_1 \mathsf{U} \psi_2))$ and
$$f'(\psi_1 \mathsf{U} \psi_2) = \begin{cases} \mathrm{JI}(f(\psi_1 \mathsf{U} \psi_2)) \setminus \mathrm{JI}(g'(\psi_2)) & \text{If } \mathrm{JI}(f(\psi_1 \mathsf{U} \psi_2)) \neq \emptyset, \\ \mathrm{JI}(g'(\psi_1 \mathsf{U} \psi_2)) \setminus \mathrm{JI}(g'(\psi_2)) & \text{Otherwise.} \end{cases}$$

Finally, every formula of the form $\psi_1 \mathsf{U} \psi_2$ contributes to the acceptance condition α the set $F_{\psi_1 \mathsf{U} \psi_2} = \{\langle g, f \rangle : \ \mathrm{JI}(f(\psi_1 \mathsf{U} \psi_2)) = \emptyset\}$.

Observe that while δ is nondeterministic, it is only nondeterministic in the first component. That is, once the function g' is chosen, there is a single function f' that can match the transition. The correctness proof can be found in the full version. □

5 LLTL Synthesis

Recall that in the synthesis problem we are given an LLTL formula φ over sets I and O of input and output variables, taking truth values from a lattice $\mathcal{L}$, and we want to generate an (I/O)-transducer over $\mathcal{L}$ all whose computations satisfy φ in a value from some desired set P of satisfaction values. In the noisy setting, the transducer may read a perturbed value of the input signals, and still all its computations need to satisfy φ as required. In this section we use the construction in Theorem 7 in order to solve both variants of the synthesis problem.

5.1 Solving the LLTL Synthesis Problem

Theorem 8. *The synthesis problem for* LLTL *is 2EXPTIME-complete. Given an* LLTL *formula φ over a lattice $\mathcal{L}$ and a predicate $P \subseteq \mathcal{L}$, we can solve the synthesis problem for $\langle \varphi, P \rangle$ in time $2^{|\mathcal{L}|^{O(|\varphi|)}}$.*

Proof. Let m denote the size of $\mathcal{L}$, and let n denote the length of φ. The construction in Theorem 7 yields an NGBW with $m^{O(n)}$ states and index n. By determinizing the NGBW we obtain an equivalent DPW $\mathcal{D}_{\varphi,P}$ of size $2^{m^{O(n)} \log m^{O(n)}} = 2^{O(n)m^{O(n)}} = 2^{m^{O(n)}}$ and index $m^{O(n)}$ [26,23]. Following the same lines as the proof of Theorem 1, we see that in order to solve the LLTL synthesis problem, it suffices to solve the parity game that is obtained from $\mathcal{D}_{\varphi}$, except that here the alphabets of Players 1 and 2 are $\mathcal{L}^O$ and $\mathcal{L}^I$, respectively. Accordingly, a winning memoryless strategy for Player 1 is an (I/O)-transducer over $\mathcal{L}$ that realizes $\langle \varphi, P \rangle$.

As stated in Theorem 1, the parity game that is obtained from $\mathcal{D}_{\varphi,P}$ can be solved in time $(2^{m^{O(n)}})^{m^{O(n)}} = 2^{m^{O(n)}}$. We conclude that the LLTL-synthesis problem is in 2EXPTIME. Hardness in 2EXPTIME follow from the hardness of the synthesis problem in the Boolean setting, which corresponds to a fully-ordered lattice with two values. □

5.2 Solving the Noisy LLTL Synthesis Problem

Consider an LLTL formula φ over the atomic propositions $I \cup O$, a predicate $P \subseteq \mathcal{L}$, and a noise function $\nu : \mathcal{L}^I \to 2^{\mathcal{L}^I}$. Recall that the goal in noisy synthesis is to find a transducer $\mathcal{T}$ that realizes $\langle \varphi, P \rangle$ with noise ν. Our goal is to construct a DPW on which we can apply the algorithm described in Theorem 1. For this, we proceed in three steps. First, we translate φ to a universal generalized co-Büchi word automaton (UGCW). Then, we incorporate the noise in the constructed UGCW. Finally, we determinize the UGCW to obtain a DPW, from which we proceed as described in Theorem 1. We start by showing how to incorporate noise in universal automata.

Lemma 1. *Consider a UGCW $\mathcal{D}$ and a noise function ν. There exists a UGCW $\mathcal{D}'$ such that $\mathcal{D}'$ accepts a computation ρ iff $\mathcal{D}$ accepts every computation ρ' that is ν-indistinguishable from ρ. Moreover, $\mathcal{D}'$ has the same state space and acceptance condition as $\mathcal{D}$.*

Proof. Let $\mathcal{D} = \langle I \cup O, Q, Q_0, \delta, \alpha \rangle$. We obtain $D' = \langle I \cup O, Q, Q_0, \delta', \alpha \rangle$ from $\mathcal{D}$ by modifying δ as follows. For every $\sigma \in I \cup O$, let $\Gamma_\sigma = \{\gamma : \gamma|_O = \sigma|_O \text{ and } \gamma|_I \in \nu(\sigma|_I)\}$. Thus, Γ_σ contains all letters that are ν-indistinguishable from σ. Then, for every state $q \in Q$, we have that $\delta'(q, \sigma) = \bigcup_{\gamma \in \Gamma_\sigma} \delta(q, \gamma)$. Thus, reading the letter σ, the UGCW $\mathcal{D}'$ simulates all the runs of $\mathcal{D}$ on all the letters that $\mathcal{D}$ may read when the actual letter in the input is σ.

It is not hard to show that the set of runs of $\mathcal{D}'$ on a computation ρ is exactly the set of all the runs of $\mathcal{D}$ on all the computations that are ν-indistinguishable from ρ. From this, the correctness of the construction follows. □

Theorem 9. *The noisy synthesis problem for* LLTL *is 2EXPTIME-complete. Given an* LLTL *formula φ over a lattice $\mathcal{L}$, a predicate $P \subseteq \mathcal{L}$, and a noise function ν, we can solve the synthesis problem for $\langle \varphi, P \rangle$ with noise ν in time $2^{m^{O(n)}}$.*

Proof. Let $\overline{P} = \mathcal{L} \setminus P$, and let $\mathcal{A}_{\varphi,\overline{P}}$ be the NGBW constructed for φ and $\overline{P}$ in Theorem 7. Observe that $\mathcal{A}_{\varphi,\overline{P}}$ accepts a computation ρ iff $[\![\rho, \varphi]\!] \notin P$. Next, we dualize $\mathcal{A}_{\varphi,\overline{P}}$ and obtain a UGCW $\mathcal{D}_{\varphi,P}$ for the complement language, namely all computations ρ such that $[\![\rho, \varphi]\!] \in P$. We now apply the procedure in Lemma 1 to $\mathcal{D}_{\varphi,P}$ and obtain a UGCW $\mathcal{D}'_{\varphi,P}$ that accepts ρ iff $\mathcal{D}_{\varphi,P}$ accepts every computation ρ' that is ν-indistinguishable from ρ. Next, we determinize $\mathcal{D}'_{\varphi,P}$ to an equivalent DPW $\mathcal{D}''_{\varphi,P}$.

We claim that the algorithm described in the proof of Theorem 1 can be applied to $\mathcal{D}''_{\varphi,P}$. To see this, let $\mathcal{D}''_{\varphi,P} = \langle I \cup O, S, s_0, \eta, \beta \rangle$ and consider the game $\mathcal{G}$ that is obtained from $\mathcal{D}''_{\varphi,P}$. That is, $\mathcal{G} = \langle \mathcal{L}^O, \mathcal{L}^I, S, s_0, \eta, \beta \rangle$, where for every $q \in S$, $i \in \mathcal{L}^I$, and $o \in \mathcal{L}^O$, we have that $\eta(q, i, o) = \mu(q, i \cup o)$.

A (memoryless) winning strategy f for Player 1 in $\mathcal{G}$ is then an (I/O)-transducer over $\mathcal{L}$ with the following property: for every strategy g of the environment, consider the play ρ that is induced by f and g. The play ρ induces a computation $w \in \mathcal{L}^{I \cup O}$ that is accepted by $\mathcal{D}''_{\varphi,P}$. By the construction of $\mathcal{D}''_{\varphi,P}$, this means that for every computation w' that is ν-indistinguishable from w, the run of $\mathcal{D}_{\varphi,P}$ on w' is accepting. Hence, $[\![w', \varphi]\!] \in P$, which in turn implies that f realizes $\langle \varphi, P \rangle$ with noise ν.

We now analyze the complexity of the algorithm. Let m denote the size of $\mathcal{L}$, and let n denote the length of φ. By Theorem 7, the size of $\mathcal{A}_{\varphi,\overline{P}}$ is $m^{O(n)}$ and it has index at most n. Dualizing results in a UGCW of the same size and acceptance condition, and so is the transition to $\mathcal{D}'_{\varphi,P}$. Determinization involves an exponential blowup, such that $\mathcal{D}''_{\varphi,P}$ is of size $2^{m^{O(n)} \log m^{O(n)}} = 2^{m^{O(n)}}$ and index $m^{O(n)}$. Finally, solving the parity game can be done in time $(2^{m^{O(n)}})^{m^{O(n)}} = 2^{m^{O(n)}}$. We conclude that the LLTL-noisy-synthesis problem is in 2EXPTIME. Hardness in 2EXPTIME again follows from the hardness of the synthesis problem in the Boolean setting. □

Remark 1. The approach described in the proofs of Theorems 1, 8, and 9 is Safrafull, in the sense it involves a construction of a DPW. As has been the case with Boolean synthesis [19], it is possible to proceed Safralessly also in LLTL synthesis with noise. To see this, note that the starting point in Theorem 1 can also be a UGCW, and that Lemma 1 works with UGCWs. In more details, once we construct a UGCW $\mathcal{U}$ for the specification, possibly with noise incorporated, the Safraless approach expands $\mathcal{U}$ to a universal co-Büchi tree automaton that accepts winning strategies for the system in the corresponding synthesis game, and checks its emptiness. In terms of complexity, rather than paying an additional exponent in the translation of the specification to a deterministic automaton, we pay it in the non-emptiness check of the tree automaton.

5.3 Local Stability Revisited

In Section 3.2 we have seen that not all LLTL formulas are locally stable. This gives rise to the question of deciding whether a given LLTL formula is locally stable. In the context of synthesis, if φ is known to be locally stable and we have a transducer $\mathcal{T}$ that realizes $\langle \varphi, P \rangle$ with no noise, we know that $\mathcal{T}$ realizes $\langle \varphi, P \oplus \gamma \rangle$ with noise ν_γ, where $\nu_\gamma(\sigma) = \{\tau : d(\sigma, \tau) \leq \gamma\}$, and $P \oplus \gamma$ is the extension of P to noise ν_γ. Thus, $\ell \in P \oplus \gamma$ iff there is $\ell' \in P$ such that $d(\ell, \ell') \leq \gamma$.

Theorem 10. *Given an* LLTL *formula φ over a lattice $\mathcal{L}$, deciding whether φ is locally stable is PSPACE-complete.*

Proof. In order to show that the problem is in PSPACE, we consider the following, more general, problem: given an LLTL formula φ and a noise-threshold γ, we want to compute the maximal *distraction*, denoted $\Delta_{\varphi,\gamma}$, that noise γ may cause to φ. Formally,

$$\Delta_{\varphi,\gamma} = \max\{d([\![\pi,\varphi]\!],[\![\pi',\varphi]\!]) : \pi,\pi' \in (\mathcal{L}^{AP})^\omega \text{ and } ld(\pi,\pi') \leq \gamma\}.$$

Observe that finding $\Delta_{\varphi,\gamma}$ allows us to decide local stability by iterating over all elements $\gamma \in \{1,\ldots,|\mathcal{L}|\}$ and verifying that $\Delta_{\varphi,\gamma} \leq \gamma$. Furthermore, in order to compute $\Delta_{\varphi,\gamma}$, it is enough to decide whether $\Delta_{\varphi,\gamma} \leq \mu$ for a threshold $\mu \in \{1,...,|L|\}$, since we can then iterate over thresholds.

We solve the dual problem, namely deciding whether there exist $\pi,\pi' \in (\mathcal{L}^{AP})^\omega$ such that $ld(\pi,\pi') \leq \gamma$ and $d([\![\pi,\varphi]\!],[\![\pi',\varphi]\!]) > \mu$. In order to solve this problem, we proceed as follows. In Theorem 7 we showed how to how to construct n NGBW $\mathcal{A}_{\varphi,\ell}$ such that $\mathcal{A}_{\varphi,\ell}$ accepts a computation π iff $[\![\pi,\varphi]\!] = \ell$. In Section 5.2, we showed how to construct a UGCW $\mathcal{D}'_{\varphi,\ell\oplus\mu}$ such that $\mathcal{D}'_{\varphi,\ell\oplus\mu}$ accepts π iff $[\![\pi',\varphi]\!] \in \ell \oplus \mu$ for every computation π' that is ν_γ-indistinguishable from π. Now, there exist $\pi,\pi' \in (\mathcal{L}^{AP})^\omega$ such that $ld(\pi,\pi') \leq \gamma$ and $d([\![\pi,\varphi]\!],[\![\pi',\varphi]\!]) > \mu$ iff there exists $\ell \in \mathcal{L}$ such that $[\![\pi,\varphi]\!] = \ell$ and the latter conditions hold. Observe that these conditions hold iff there exists a computation π that is accepted by $\mathcal{A}_{\varphi,\ell}$ but not by $\mathcal{D}'_{\varphi,\ell\oplus\mu}$. Thus, it suffices to decide whether $L(\mathcal{A}_{\varphi,\ell}) \cap \overline{L(\mathcal{D}'_{\varphi,\ell\oplus\mu})} = \emptyset$ for every $\ell \in \mathcal{L}$.

Finally, we analyze the complexity of this procedure. Let $|\mathcal{L}| = m$ and $|\varphi| = n$. Complementation of $\mathcal{D}'_{\varphi,\ell\oplus\mu}$ can be done by constructing $\mathcal{D}'_{\varphi,\overline{\ell\oplus\mu}}$. Hence, both $\mathcal{A}_{\varphi,\ell}$ and $\overline{\mathcal{D}'_{\varphi,\ell\oplus\mu}}$ have $m^{O(n)}$ states. Checking the emptiness of their intersection can be done on-the-fly in PSPACE, implying the required upper bound.

We prove hardness in PSPACE by describing a polynomial time reduction from the satisfiability problem for LTL to the complement of the local-stability problem. Consider an LTL formula φ over AP. We assume that φ is not valid, thus there is a computation that does not satisfy it (clearly LTL satisfiability is PSPACE-hard also with this promise). We construct an LLTL formula ψ over the lattice $\mathcal{L} = \langle 2^{\{a,b\}}, \subseteq\rangle$ as follows. Let $AP' = \{p' : p \in AP\}$ be a tagged copy of AP. We define $\psi = \varphi \vee \varphi'$ over $AP \cup AP'$, where φ' is obtained form φ by replacing each atomic proposition by its tagged copy. Clearly this reduction is polynomial. In the full version, we show that φ is satisfiable iff ψ is not locally stable. □

References

1. Almagor, S., Boker, U., Kupferman, O.: Formalizing and reasoning about quality. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part II. LNCS, vol. 7966, pp. 15–27. Springer, Heidelberg (2013)
2. Alur, R., Kanade, A., Weiss, G.: Ranking automata and games for prioritized requirements. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 240–253. Springer, Heidelberg (2008)

3. Bloem, R., Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Better quality in synthesis through quantitative objectives. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 140–156. Springer, Heidelberg (2009)
4. Cerný, P., Henzinger, T.: From boolean to quantitative synthesis. In: EMSOFT, pp. 149–154 (2011)
5. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 385–400. Springer, Heidelberg (2008)
6. Chatterjee, K., Doyen, L., Henzinger, T.A., Raskin, J.-F.: Algorithms for omega-regular games with imperfect information. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 287–302. Springer, Heidelberg (2006)
7. Chatterjee, K., Majumdar, R.: Minimum attention controller synthesis for omega-regular objectives. In: Fahrenberg, U., Tripakis, S. (eds.) FORMATS 2011. LNCS, vol. 6919, pp. 145–159. Springer, Heidelberg (2011)
8. Chatterjee, K., Majumdar, R., Henzinger, T.A.: Controller synthesis with budget constraints. In: Egerstedt, M., Mishra, B. (eds.) HSCC 2008. LNCS, vol. 4981, pp. 72–86. Springer, Heidelberg (2008)
9. Chechik, M., Devereux, B., Gurfinkel, A.: Model-checking infinite state-space systems with fine-grained abstractions using SPIN. In: Dwyer, M.B. (ed.) SPIN 2001. LNCS, vol. 2057, pp. 16–36. Springer, Heidelberg (2001)
10. Church, A.: Logic, arithmetics, and automata. In: Proc. Int. Congress of Mathematicians, 1962, pp. 23–35. Institut Mittag-Leffle (1962)
11. Emerson, E., Jutla, C.: Tree automata, μ-calculus and determinacy. In: Proc. 32nd FOCS, pp. 368–377 (1991)
12. Faella, M., Legay, A., Stoelinga, M.: Model checking quantitative linear time logic. Electr. Notes Theor. Comput. Sci. 220(3), 61–77 (2008)
13. Filiot, E., Jin, N., Raskin, J.-F.: An antichain algorithm for LTL realizability. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 263–277. Springer, Heidelberg (2009)
14. Huth, M., Pradhan, S.: Consistent partial model checking. Electr. Notes Theor. Comput. Sci. 73, 45–85 (2004)
15. Jurdzinski, M., Paterson, M., Zwick, U.: A deterministic subexponential algorithm for solving parity games. SIAM Journal on Computing 38(4), 1519–1532 (2008)
16. Kupferman, O., Lustig, Y.: Lattice automata. In: Cook, B., Podelski, A. (eds.) VMCAI 2007. LNCS, vol. 4349, pp. 199–213. Springer, Heidelberg (2007)
17. Kupferman, O., Lustig, Y.: Latticed simulation relations and games. International Journal on the Foundations of Computer Science 21(2), 167–189 (2010)
18. Kupferman, O., Vardi, M.: Church's problem revisited. The Bulletin of Symbolic Logic 5(2), 245–263 (1999)
19. Kupferman, O., Vardi, M.: Safraless decision procedures. In: Proc. 46th FOCS, pp. 531–540 (2005)
20. Kwiatkowska, M.: Quantitative verification: models techniques and tools. In: ESEC/SIGSOFT FSE, pp. 449–458 (2007)
21. Martin, D.A.: Borel Determinacy. Annals of Mathematics 65, 363–371 (1975)
22. Miyano, S., Hayashi, T.: Alternating finite automata on ω-words. TCS 32, 321–330 (1984)
23. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. In: Proc. 21st LICS, pp. 255–264. IEEE (2006)
24. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Proc. 16th POPL, pp. 179–190 (1989)
25. Reif, J.: The complexity of two-player games of incomplete information. Journal of Computer and Systems Science 29, 274–301 (1984)
26. Safra, S.: Exponential Determinization for ω-Automata with Strong-Fairness Acceptance Condition. In: Proc. 24th STOC, pp. 275–282 (1992)

27. Schewe, S.: Solving Parity Games in Big Steps. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007. LNCS, vol. 4855, pp. 449–460. Springer, Heidelberg (2007)
28. Vardi, M.Y.: From verification to synthesis. In: Shankar, N., Woodcock, J. (eds.) VSTTE 2008. LNCS, vol. 5295, p. 2. Springer, Heidelberg (2008)
29. Vardi, M., Wolper, P.: Reasoning about infinite computations. Information and Computation 115(1), 1–37 (1994)
30. Velner, Y., Rabinovich, A.: Church synthesis problem for noisy input. In: Hofmann, M. (ed.) FOSSACS 2011. LNCS, vol. 6604, pp. 275–289. Springer, Heidelberg (2011)