Discrete Event Dyn Syst (2017) 27:547–572
DOI 10.1007/s10626-017-0242-0

Latticed-LTL synthesis in the presence of noisy inputs

Shaull Almagor[1] · **Orna Kupferman**[2]

Received: 25 December 2015 / Accepted: 9 March 2017 / Published online: 6 April 2017

Abstract In the classical *synthesis* problem, we are given a specification 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 multi-valued setting in which the truth values of the input and output signals are taken from a finite lattice, and so is the satisfaction value of specifications. We consider specifications in *latticed linear temporal logic* (LLTL). In LLTL, conjunctions and disjunctions correspond to the meet and join operators of the lattice, respectively, and the satisfaction values of formulas are taken from the lattice too. 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 the given specification in a desired satisfaction value. 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 the desired satisfaction value. We study the problem of noisy LLTL synthesis, as well as

A preliminary version appears in "Latticed-LTL synthesis in the presence of noisy inputs." Foundations of Software Science and Computation Structures, LNCS 8421, Springer, 2014. This research has received funding from the European Research Council under the EU's 7-th Framework Programme (FP7/2007-2013) / ERC grant agreement no 278410.

✉ Shaull Almagor
shaull.almagor@mail.huji.ac.il

Orna Kupferman
orna@cs.huji.ac.il

[1] Oxford University, Oxford, UK

[2] The Hebrew University, Jerusalem, Israel

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. We prove that the noisy-synthesis problem for LLTL is 2EXPTIME-complete, as is traditional LTL synthesis.

Keywords Synthesis · Lattice · Quantitative formal methods · Automata · Noise

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 (1963). The modern approach to synthesis was initiated by Pnueli and Rosner, who introduced LTL (linear temporal logic) synthesis (Pnueli and Rosner 1989a): 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 realizes ψ if all the computations that are generated by the interaction satisfy ψ. Synthesis has attracted a lot of research and interest (Vardi 2008).

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 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 (Chatterjee et al. 2008; Almagor et al. 2013), and arises indirectly in probabilistic settings, systems with multiple and inconsistent view-points, specifications with priorities, and more (Kwiatkowska 2007; Huth and Pradhan 2004; Alur et al. 2008). 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 (Bloem et al. 2009; Cerný and Henzinger 2011; Almagor et al. 2013).

A different extension of the classical synthesis framework considers settings in which the system has *incomplete information* about its environment. In early work on incomplete information, 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 $I \cup O$ (Kumar and Shayman 1995; Pnueli and Rosner 1989b; Kupferman and Vardi 1999; Chatterjee et al. 2006; Chatterjee and Majumdar 2011). The setting is equivalent to a game with incomplete information, extensively studied in Reif (1984). 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 Chatterjee et al. (2008), the authors study a case 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 Velner and Rabinovich (2011), the authors study games with *errors*. Such games correspond to a synthesis scenario 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 (Alur et al. 2008)), 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. In addition, we consider a probabilistic setting,

where the noise is given by some distribution, rather than by an adversary. Then, the goal is to synthesize a transducer that maximizes the probability of satisfying the specification. By utilizing the results of Chatterjee et al. (2004), we show that this problem can be solved in 2NEXPTIME∩co-2NEXPTIME.

1.1 Related work

As described above, researchers have extensively studied synthesis with incomplete information, as well as quantitative extensions to Boolean synthesis. Here, we describe work on synthesis in the presence of noisy input.

A variant of noisy synthesis was considered in Majumdar et al. (2011) for *metric automata* – deterministic automata equipped with a metric on the state space. There, the authors consider controller synthesis, where the automaton tries to generate a word in its language, namely one on which the run of the automaton is accepting, and an adversary is allowed to disturb the run of the automaton (i.e., to change the current state). The goal is to find a strategy for the automaton such that if the disturbance is bounded with respect to the metric, then the run of the automaton is close (with respect to the metric) to being accepting. The authors describe polynomial-time algorithms for robust synthesis for Büchi automata as well as, under certain conditions, for parity automata.

There are two main differences between our work and Majumdar et al. (2011). First, the measure of correctness in Majumdar et al. (2011) is Boolean, in the sense that the set of accepting runs has a Boolean characterization function, and the source of quantitativeness is the distance of the generated run from the set of accepting runs. In contrast, our definition of correctness is inherently multi-valued, as lattice automata are multi-valued. Second, in Majumdar et al. (2011), the only input from the environment is the disturbance, whereas our case is similar to conventional synthesis, in which the environment controls the input, and the noise is either adversarial or random.

Another approach for modeling noise was taken in Topcu et al. (2012), where systems with *unmodeled* transitions are considered. These systems are equipped with a set Δ of transitions that may or may not actually exist. Then, a controller is *robust* for a system S and a specification φ if it realizes φ in all systems that are obtained from S by adding to it a subset of the transitions in Δ. Since such robustness conditions are extremely strong, the authors define "levels" of robustness, which are assigned by giving a rank to every subset of Δ, and assigning to a transducer the maximal rank it realizes. The authors show that when the ranking function is induced by simple-enough partial orders (e.g. set inclusion), then finding an optimal transducer can be done in 2EXPTIME for LTL specifications.

Conceptually, our work differs from Topcu et al. (2012) in that the noise is given in an online manner by an adversary, as part of the input, and does not involve a structural change in the system. In addition, the ranking in Topcu et al. (2012) is again based on a Boolean notion of correctness, whereas in our case correctness is quantitative.

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 $\mathcal{L}$ (Chechik et al. 2001; Kupferman and Lustig 2007). 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 $\mathcal{L}$).

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

Example 1 Consider a setting in which three agents a, b, and c have different view-points on a system $\mathcal{S}$. A truth assignment for the atomic propositions is then a function in $(2^{\{a,b,c\}})^{AP}$ assigning to each $p \in AP$ the set of agents according to whose view-point 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.

Remark 1 [**Constants in** LLTL] Recall that the constants `True` and `False` in LTL do not add to its expressive power. Indeed, `True` can be replaced by $p \vee (\neg p)$, for a Boolean atomic proposition p, and similarly for `False`. This is not the case for the constants $\ell \in \mathcal{L}$ in LLTL. For example, consider the linear lattice $\langle \{1, ..., 5\}, \leq \rangle$. It is easy to show that for every formula φ (without constants) over the atomic propositions AP, the computation π for which $\pi_i(p) = 3$ for every $i \geq 0$ and $p \in AP$, satisfies $[\![\pi, \varphi]\!] = 3$. It follows that there is no LLTL formula φ that is equivalent to the constant 1.

Constants can be used to upper or lower bound the satisfaction value of an LLTL formula. For example, the truth value of the LLTL formula $\{a, b\} \wedge \psi$, defined with respect to the lattice $\langle 2^{\{a,b,c\}}, \subseteq \rangle$, is the set of agents that is both a subset of $\{a, b\}$ and according to whose viewpoint, the specification ψ is satisfied.

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 $\mathcal{L}^O$. 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 $\ell' \geq \ell$.

Example 2 Consider a system that grants requests to a server. Requests (r) have an importance ranking, and grants (g) have a quality ranking. Both rankings are in $\{1, ..., 10\}$. Consider a specification to the system such that the satisfaction value of the specification in a computation depends on the following three parameters: (1) the importance of the requests and the quality of the grants: the more important the requests is, the higher the quality of the response should be (2) Ideally, each request is immediately given a grant that holds for two time steps. We are willing, however, to compromise for a grant that only holds for one time

step, but this reduces the satisfaction value. Specifically, if the grant is given only in the single time step after the request, the satisfaction value is at most 8, and if it is given only in the single next time step, then the satisfaction value is at most 6. Finally, (3) high-quality grants are expensive. Specifically, for every window of three times steps, the satisfaction value is bounded by $10 - v$, where v is the lowest quality of a grant given in the window. We can specify the system by a conjunction $\varphi \wedge \psi$ of LLTL formulas over the lattice $\langle \{1, ..., 10\}, \leq \rangle$, where $\varphi = \mathsf{G}(r \rightarrow ((\mathsf{X}g \wedge \mathsf{XX}g) \vee (\mathsf{X}g \wedge 8) \vee (\mathsf{XX}g \wedge 6)))$ expresses the granting policy, and $\psi = \mathsf{G}(\neg(g \wedge \mathsf{X}g \wedge \mathsf{XX}g))$ expresses the need to have at least one low-quality grant in every window of three time steps.

It is not hard to see that the specification cannot be realizable with a satisfaction value above 6, as an input sequence in which requests of a high importance are received always, causes φ and ψ to conflict. We thus add an assumption about the demand to the server and require that consecutive request cannot be both important. Formally, we define $\theta = \mathsf{G}(\neg(r \wedge \mathsf{X}r))$, and the full specification to the system is $\theta \rightarrow (\varphi \wedge \psi)$.

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.

Remark 2 **[Incomplete information as noise]** As discussed in Section 1, synthesis with *incomplete information* has been extensively studied in the Boolean setting (Kupferman and Vardi 2000). Synthesis with incomplete information, in both the Boolean and the multi-valued settings, can be viewed as a special case of our noisy synthesis. To see this, let $I \cup H \cup O$ be a partition of the signals to input (that is, visible), hidden, and output signals, respectively. Consider the noise function ν in which for $i \subseteq I$ and $h \subseteq H$, we have $\nu(i \cup h) = \{i \cup h' : h' \subseteq H\}$. The function ν makes letters that agree on the assignment

to the input signals and differ only in the hidden signal indistinguishable, and thus models incomplete information.

The synthesis procedure described in Theorem 10 thus enables us to solve also synthesis with incomplete information. The obtained complexity coincides with the one given in (Kupferman and Vardi 2000).

Example 3 Recall our request-granting specification from Example 2. As described there, requests get values in $\{1, \ldots, 10\}$, reflecting their importance. Assume that the channel over which requests are sent is noisy and can perturb the value by 2. Thus, the noise function is $\nu(i) = \{i-2, i-1, i, i+1, i+2\} \cap \{1, \ldots, 10\}$. The effect of such a noise is an increase in the importance of all requests. Indeed, since the synthesized system has to satisfy the specification regardless of the noise, and requests appear negatively (that is, in the left-hand side of an implication) in the specification, then noise that increases their value may reduce the satisfaction value of the specification. Thus, in the presence of noise ν, the synthesized systems has to respond with grants of higher quality.

2.5 Automata and games

As described in Section 1, our solution to the LLTL noisy-synthesis problem is based on 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 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 (Martin 1975), 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 (Pnueli and Rosner 1989a).[1] It involves a translation of the specification into a deterministic parity 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 (Kupferman and Vardi 2005; Filiot et al. 2009). 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.[2] In Remark 3, 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 $\eta : Q \times 2^I \times 2^O \to Q$ 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 Emerson and Jutla (1991), the game is determined and one of the players has a memoryless winning strategy. Such a strategy for Player 1 in $\mathcal{G}_\varphi$ can then be viewed as a transducer that realizes φ, as follows: the states of the transducer are the states of the game, and at each state, the output corresponds to that prescribed by the strategy. The inputs then move the state of the transducer according to the transitions of the game.

Finally, the game $\mathcal{G}_\varphi$ is of size $O(t)$ and index k. Hence, by Jurdzinski et al. (2008) and Schewe (2007), 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 that are relevant in the context of noisy synthesis. We focus on the set of attainable satisfaction values of an LLTL formula and on

[1] In Pnueli and Rosner (1989a) and other early works the games are formulated by means of tree automata.

[2] State-of-the-art algorithms for solving parity games achieve a better complexity (Jurdzinski et al. 2008; Schewe 2007). 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.

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.

In particular, as will follow from Section 5, synthesis on pointed lattices can be optimized by reducing the number of lattice subsets that one needs to check in order to establish the values with which a formula is satisfiable. For example, in the subset lattice $2^{\{a,b\}}$ (which is pointed by Theorem 3 below), if a formula is realizable both with value b and with value b, then it is realizable with value $\{a, b\}$.

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. We then show that the lattices we focus on, are, however, pointed.

Theorem 2 *Not all distributive De-Morgan lattices are pointed.*

Proof 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 $S_1 \subseteq S_2$). (See Fig. 1). We define $\neg\langle S, v \rangle = \langle \{a, b\} \setminus S, 1 - v \rangle$. That is, negation negates both components. 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. □

Theorem 3 *Fully-ordered lattices and power-set lattices are pointed.*

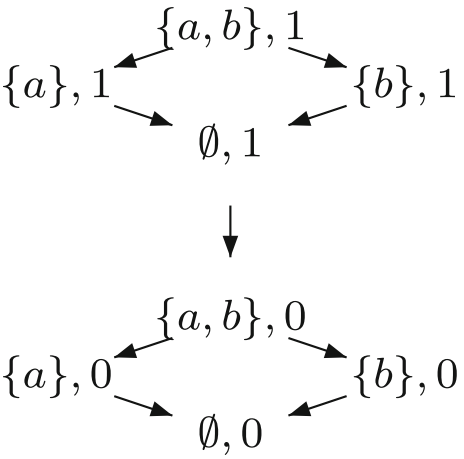


Fig. 1 The lattice $\langle 2^{\{a,b\}} \times \{0, 1\}, \leq \rangle$

Proof For fully-ordered lattices, we have $\ell_1 \vee \ell_2 \in \{\ell_1, \ell_2\}$, so pointed-ness is obvious. We prove the claim for power-set lattices. Consider a lattice $\mathcal{L} = \langle 2^X, \subseteq \rangle$ for some finite set X, and consider an LLTL formula φ over the atomic propositions $I \cup O$. For every set $\ell \in \mathcal{L}$ and element $x \in X$, we define the *projection* $\ell_{|x}$ *of* ℓ *on* x to be `True` if $x \in \ell$ and `False` otherwise. We extend the definition of projection to a letter $\sigma \in \mathcal{L}^{I \cup O}$ by letting $p \in \sigma_{|x}$ iff $\mathcal{L}(p)_{|x} = \texttt{True}$. Thus, $\sigma_{|x} \subseteq I \cup O$. Finally, we extend the definition to a computation $\pi \in (\mathcal{L}^{I \cup O})^\omega$ by setting $(\pi_{|x})_i = (\pi_i)_{|x}$. Observe that $\pi_{|x} \in (2^{I \cup O})^\omega$.

For an element $x \in X$, let $\varphi_{|x}$ be the LTL formula obtained from φ by replacing every element $\ell \in \mathcal{L}$ that appears in φ by $\ell_{|x}$. Since the syntax of LLTL differs from that of LTL only by allowing elements from the lattice, it follows that $\varphi_{|x}$ is indeed an LTL formula.

We prove that for every computation $\pi \in (\mathcal{L}^{I \cup O})^\omega$ and for every $\ell \in \mathcal{L}$, it holds that $[\![\pi, \varphi]\!] \geq \ell$ iff $\pi_{|x} \models \varphi_x$ for all $x \in \ell$. Observe that $[\![\pi, \varphi]\!] \geq \ell$ iff $x \in [\![\pi, \varphi]\!]$ for all $x \in \ell$. From here the claim easily follows by induction on the structure of φ.

Now, assume that φ is (I/O)-realizable with value at least ℓ_1 and with value at least ℓ_2. We claim that φ is realizable with value $\ell_1 \vee \ell_2$. W.l.o.g we can assume $\ell_1 \cap \ell_2 = \emptyset$ (otherwise we replace ℓ_2 by $\ell_2 \setminus \ell_1$). Let $\mathcal{T}_1 = \langle \mathcal{L}, I, O, S^1, s_0^1, \eta^1, \tau^1 \rangle$ and $T_2 = \langle \mathcal{L}, I, O, S^2, s_0^2, \eta^2, \tau^2 \rangle$ be transducers that realize φ with values ℓ_1 and ℓ_2, respectively. We obtain from $\mathcal{T}_1$ and $\mathcal{T}_2$ a new transducer $\mathcal{T} = \langle \mathcal{L}, I, O, S^1 \times S^2, \langle s_0^1, s_0^2 \rangle, \eta, \tau \rangle$ as follows. For every state $\langle s, t \rangle \in S^1 \times S^2$ and $\sigma \in \mathcal{L}^I$, we have $\eta(\langle s, t \rangle, \sigma) = \langle \eta^1(s, \sigma), \eta^2(t, \sigma) \rangle$. For every state $\langle s, t \rangle \in S^1 \times S^2$ and for every $o \in O$, we have $\tau(\langle s, t \rangle)(o) = (\ell_1 \cap \tau^1(s)(o)) \cup (\ell_2 \cap \tau^2(s)(o))$. We claim that $\mathcal{T}$ realizes φ with value $\ell_1 \vee \ell_2$.

Consider an environment-computation $\pi \in (\mathcal{L}^I)^\omega$, and consider the corresponding computations $\rho, \rho' \in \mathcal{L}^{I \cup O}$ of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. It holds that $[\![\rho, \varphi]\!] \geq \ell_1$ and $[\![\rho', \varphi]\!] \geq \ell_2$.

Consider the output computation $\theta \in (\mathcal{L}^O)^\omega$ of $\mathcal{T}$ on the input π. By the construction of $\mathcal{T}$, it is easy to prove that $\theta_i(o) = (\ell_1 \cap \rho_i(o)) \cup (\ell_2 \cap \rho'_i(o))$.

Consider the computation π' of $\mathcal{T}$ obtained by combining π and θ. For every $x \in \ell_1$, we have that $\pi'_{|x} = \rho_{|x}$. Thus, $\pi'_{|x} \models \varphi_x$ for every $x \in \ell_1$. Similarly, for every $x \in \ell_2$, we have that $\pi'_{|x} = \rho'_{|x}$, so $\pi'_{|x} \models \varphi_x$ for every $x \in \ell_2$. We conclude that for every $x \in \ell_1 \cup \ell_2$ it holds that $\pi'_{|x} \models \varphi$, and so $[\![\pi', \varphi]\!] \geq \ell_1 \vee \ell_2$. Thus, $\mathcal{T}$ realizes φ with value $\ell_1 \vee \ell_2$. □

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 6, we study the problem of deciding whether a given LLTL formula is locally stable, and discuss the relevancy of local 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.

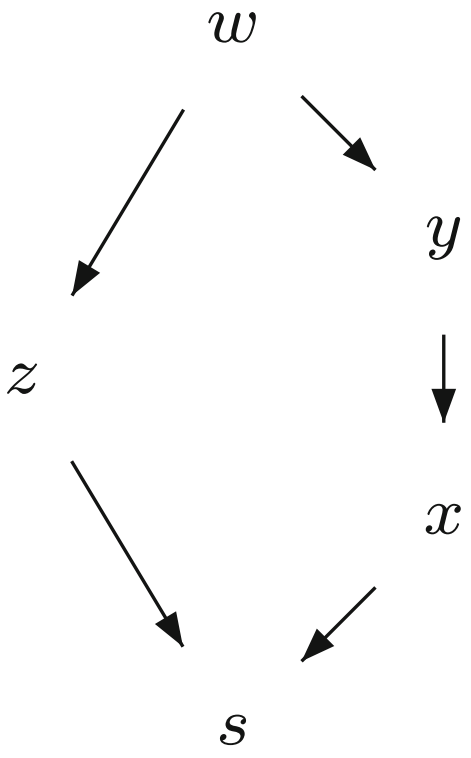
We start by breaking this intuition, showing that for non-distributive lattices, this is false. The proof makes use of an *N5 structure*, depicted in Fig. 2. Formally, an N5 structure in a lattice $\mathcal{L}$ is a tuple $\langle x, y, z, w, s\rangle$ such that the following relations hold: $s < x < y < w$, $s < z < w$, $y \not\leq z$, $z \not\leq y$, $x \not\leq z$, 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 4 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.

Fig. 2 An N5 structure



Theorem 5 LLTL *formulas are globally stable with respect to De-Morgan distributive lattices.*

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

To simplify the proof, we observe that since π and π' differ only in a single index, and in particular only in a finite prefix, we can avoid treating the case of U subformulas. Indeed, we can expand subformulas of the form $\psi \mathsf{U} \theta$ using propositional conjunctives and the X operator so that all the suffixes before π^i are not evaluated on subformulas that contain U. Clearly, the value of the remaining evaluation of U subformulas does not change, as π and π' agree on suffixes that start after the i-th position.

- If $\varphi = \ell \in \mathcal{L}$ the claim is trivial.
- If $\varphi = p \in AP$, then $d([\![\pi, \varphi]\!], [\![\pi', \varphi]\!]) = d([\![\pi_0, p]\!], [\![\pi'_0, p]\!])$, which, by the assumption, is at most 1. Moreover, $[\![\pi', p]\!] \in \{[\![\pi, p]\!], [\![\pi, p]\!] \vee u\}$.
- If $\varphi = \neg\psi$, then by the induction hypothesis it holds that $[\![\pi', \psi]\!] \in \{[\![\pi, \psi]\!] \wedge \neg u, [\![\pi, \psi]\!], [\![\pi, \psi]\!] \vee u\}$ and $d([\![\pi, \psi]\!], [\![\pi', \psi]\!]) \leq 1$. Hence, by the properties of negation, $d(\neg[\![\pi, \psi]\!], \neg[\![\pi', \psi]\!]) \leq 1$ and $[\![\pi', \varphi]\!] \in \{[\![\pi, \varphi]\!] \wedge \neg u, [\![\pi, \varphi]\!], [\![\pi, \varphi]\!] \vee u\}$, and we are done.
- If $\varphi = \psi \vee \theta$, then, by the induction hypothesis, it holds in particular that $[\![\pi, \psi]\!] \wedge \neg u \leq [\![\pi', \psi]\!] \leq [\![\pi, \psi]\!] \vee u$ and $[\![\pi, \theta]\!] \wedge \neg u \leq [\![\pi', \theta]\!] \leq [\![\pi, \theta]\!] \vee u$. Therefore, $[\![\pi, \varphi]\!] \wedge \neg u \leq [\![\pi', \varphi]\!] \leq [\![\pi, \varphi]\!] \vee u$. Figure 3 demonstrates the above relations. Intuitively, adding a lattice element between $\psi \vee \theta \vee u$ and $(\psi \vee \theta) \wedge u$ induces an

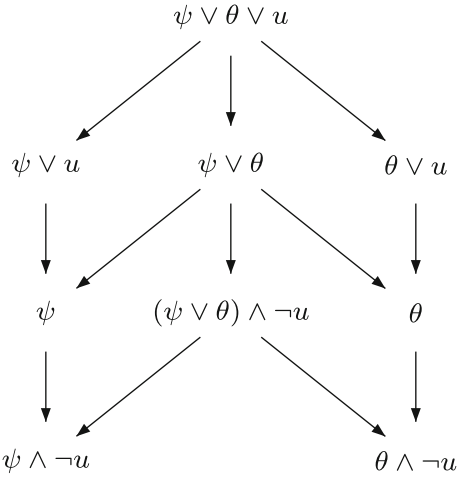


Fig. 3 The relations described in the disjunction case in the proof of Theorem 5

N5-structure. Thus, $[\![\pi', \varphi]\!]$ must be within distance 1 of $[\![\pi, \varphi]\!]$. This is the key point in the proof.

Formally, in order to prove that the distance is preserved, we distinguish between cases. First, if $[\![\pi', \varphi]\!] = [\![\pi, \varphi]\!]$, then we are done. If $[\![\pi', \varphi]\!] = [\![\pi, \varphi]\!] \vee v$, then since $[\![\pi', \varphi]\!] = [\![\pi', \psi]\!] \vee [\![\pi', \theta]\!]$, it must be that w.l.o.g $[\![\pi', \psi]\!] = [\![\pi, \psi]\!] \vee u$. That is, at least one of ψ and θ gets joined with u. Assume by way of contradiction that there exists $t \in \mathcal{L}$ such that $[\![\pi', \varphi]\!] < t < [\![\pi, \varphi]\!] \vee v$. We then have the N5 structure[3] $\langle \psi \vee \theta, t, \psi \vee u, \psi, \psi \vee \theta \vee u \rangle$. Since, however, $\mathcal{L}$ is distributive, it cannot have an N5 structure, and we have reached a contradiction.

The case $[\![\pi, \varphi]\!] \wedge \neg u \leq [\![\pi', \varphi]\!]$ is handled similarly.

– If $\varphi = \mathsf{X}\psi$, the claim follows immediately from the induction hypothesis.

□

Consider an LLTL formula φ over a distributive de-Morgan lattice, and assume we have a transducer $\mathcal{T}$ that realizes φ with some value v. Intuitively, Theorem 5 assures us that if very little and few perturbations occur (either in the inputs or outputs), then $\mathcal{T}$ still realizes φ with a value that is "close" to v. While this is a very weak notion of stability, it still offers some ability to handle noise.

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 6 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 7 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. □

In Section 6 we study local stability further and prove that the problem of deciding whether a given LLTL formula is locally stable is PSPACE-complete.

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* (Kupferman and Lustig 2007). 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

[3] It may be the case that some of the nodes coincide, and this is not a proper N5. However, these cases are easy to handle.

model-checking problems for LLTL (Kupferman and Lustig 2007). 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 (Kupferman and Lustig 2010). 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 that LLTL formulas have finitely many possible satisfaction values suggests that this is possible. For fully-ordered lattices, a similar approach has been taken in Faella et al. (2008) and Almagor et al. (2013). 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 word automata (NGBW), as introduced in Vardi and Wolper (1994). 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 Faella et al. (2008) and Almagor et al. (2013), 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 (Vardi and Wolper 1994; Miyano and Hayashi 1984), 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 8 *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.

We now proceed to prove the correctness of the construction and analyze the blow-up it involves. In the proof, we identify a set $S \subseteq \mathrm{JI}(\mathcal{L})$ with the element $\bigvee_{s\in S} s \in \mathcal{L}$. Observe that it suffices to prove that for every $\ell \in \mathcal{L}$, the NGBW $\mathcal{A}_{\varphi,\{\ell\}}$ accepts a computation π iff $[\![\pi, \varphi]\!] = \ell$. We first prove that if $\pi \in (\mathcal{L}^{AP})^\omega$ is such that $[\![\pi, \varphi]\!] = \ell$ for some $\ell \in \mathcal{L}$, then $\mathcal{A}_{\varphi,\{\ell\}}$ accepts π. For every $i \in \mathbb{N}$, let $g_i \in G_\varphi$ be such that for all $\psi \in cl(\varphi)$, we have that $g_i(\psi) = [\![\pi^i, \psi]\!]$. Also, let $f_0 : ucl(\varphi) \to \mathcal{L}$ be such that for every subformula of the form $\psi_1 \mathsf{U} \psi_2$, we have $f_0(\psi_1 \mathsf{U} \psi_2) = \mathrm{JI}(g_0(\psi_1 \mathsf{U} \psi_2)) \setminus \mathrm{JI}(g_0(\psi_2))$. Finally, for $i \in \mathbb{N}$, let f_{i+1} be induced from f_i and g_{i+1} in the single way that satisfies the conditions in the definition of δ.

We claim that $r = \langle g_0, f_0\rangle, \langle g_1, f_1\rangle, \ldots$ is an accepting run of $\mathcal{A}_{\varphi,\{l\}}$ on π. First, the semantics of LLTL implies that the consistency conditions, both the local ones and these imposed by δ are satisfied. In particular, for the conditions imposed by δ, this follows from the fact that for all positions $i \in \mathbb{N}$, we have that $[\![\pi^i, X\psi_1]\!] = [\![\pi^{i+1}, \psi_1]\!]$ and $[\![\pi^i, \psi_1 \mathsf{U} \psi_2]\!] = [\![\psi_2]\!] \vee ([\![\pi^i, \psi_1]\!] \wedge [\![\pi^i, \psi_1 \mathsf{U} \psi_2]\!])$. Also, since $g_0(\varphi) = \ell$, then $\langle g_0, f_0\rangle \in Q_0$

It is left to prove that r is accepting. Consider a sub-formula of the form $\psi_1 \mathsf{U} \psi_2$. We prove that r visits $F_{\psi_1 \mathsf{U} \psi_2}$ infinitely often. Consider a position $i \in \mathbb{N}$ and let $[\![\pi^i, \psi_1 \mathsf{U} \psi_2]\!] = y$. We prove that there is a position $n \geq i$ such that $f_n = \emptyset$, thus $\langle g_n, f_n \rangle \in F_{\psi_1 \mathsf{U} \psi_2}$. By the semantics of U and the finiteness of $\mathcal{L}$, there is a (minimal) index $n \geq i$ such that $y = \bigvee_{i \leq j \leq n} ([\![\pi^j, \psi_2]\!] \wedge \bigwedge_{i \leq k < j} [\![\pi^k, \psi_1]\!])$. It is easy to prove by induction on $n - i$ that there exists some $i \leq k \leq n$ such that $\mathrm{JI}(f_k(\psi_1 \mathsf{U} \psi_2)) = \emptyset$, using the fact that $f_j(\psi_1 \mathsf{U} \psi_2) \leq g_j(\psi_1 \mathsf{U} \psi_2)$ for all $j \geq 0$.

The other direction is more complicated. Let $\pi \in (\mathcal{L}^{AP})^\omega$ be such that π is accepted by $\mathcal{A}_{\varphi,\{\ell\}}$. We prove that $[\![\pi, \varphi]\!] = \ell$. Let $\rho = \langle g_1, f_1 \rangle, \langle g_2, f_2 \rangle, \ldots$ be an accepting run of $\mathcal{A}_{\varphi,\{\ell\}}$ on π, and let $h_1, h_2, \ldots \in (G_\varphi)^\omega$ be such that for all $i \in \mathbb{N}$ and $\psi \in cl(\varphi)$, we have that $h_i(\psi) = [\![\pi^i, \psi]\!]$. We claim that $h_i = g_i$ for all $i \in \mathbb{N}$. The proof is by induction on the structure of the formulas in $cl(\varphi)$. Consider a formula $\psi \in cl(\varphi)$. If $\psi = p \in AP$, then since ρ is a legal run, a transition from state $\langle g_i, f_i \rangle$ is possible with letter σ iff $\sigma(p) = [\![\pi^i, p]\!] = \rho(p)$, and we are done. If $\psi = v \in \mathcal{L}$, $\psi = \psi_1 \vee \psi_2$, or $\psi = \mathsf{X}\psi_1$, then the claim follows from the consistency rules and the induction hypothesis. Finally, if $\psi = \psi_1 \mathsf{U} \psi_2$, then, as we prove in Lemma 1 below, the fact that ρ is an accepting run implies the first equality in the chain below. The second equality follows from the induction hypothesis, and the third equality is from the semantics of LLTL.

$$g_i(\psi) = \bigvee_{i \leq j} \Big(g_j(\psi_2) \wedge \bigwedge_{i \leq k < j} g_k(\psi_1) \Big) = \bigvee_{i \leq j} \Big([\![\pi^j, \psi_2]\!] \wedge \bigwedge_{i \leq k < j} [\![\pi^k, \psi_1]\!] \Big) = [\![\pi^i, \psi]\!].$$

We conclude that $h_0 = g_0$. Since $g_0 \in Q_0$, it follows that $[\![\pi, \varphi]\!] = \ell$ and we are done. □

Lemma 1 *Under the notations of the proof of Theorem 8, we have that* $g_i(\psi_1 \mathsf{U} \psi_2) = \bigvee_{i \leq j} (g_j(\psi_2) \wedge \bigwedge_{i \leq k < j} g_k(\psi_1))$.

Proof Since ρ is a legal run, then for every $i \in \mathbb{N}$, it holds that

$$g_i(\psi_1 \mathsf{U} \psi_2) = g_i(\psi_2) \vee (g_i(\psi_1) \wedge g_{i+1}(\psi_1 \mathsf{U} \psi_2)). \quad (*)$$

We prove the lemma by proving that for every $v \in \mathrm{JI}(\mathcal{L})$ it holds that $g_i(\psi_1 \mathsf{U} \psi_2) \geq v$ iff $\bigvee_{i \leq j} (g_j(\psi_2) \wedge \bigwedge_{i \leq k < j} g_k(\psi_1) \geq v$. The equality then follows from Birkhoff's theorem.

Using $(*)$, it is easy to prove by induction that for every index i and for every $n \in \mathbb{N}$ it holds that

$$g(\psi_1 \mathsf{U} \psi_2) = \bigvee_{i \leq j \leq n} \left(g_j(\psi_2) \wedge \bigwedge_{i \leq k < j} g_k(\psi_1) \right) \vee \left(g_{n+1}(\psi_1 \mathsf{U} \psi_2) \wedge \bigwedge_{i \leq k \leq n} g_k(\psi_1) \right).$$

We denote the above equation by $(**)$.

Let $v \in \mathrm{JI}(\mathcal{L})$ and assume that

$$\bigvee_{i \leq j} (g_j(\psi_2) \wedge \bigwedge_{i \leq k < j} g_k(\psi_1)) \geq v.$$

Since v is join-irreducible, it follows that there exist some $n \in \mathbb{N}$ such that

$$g_n(\psi_2) \wedge \bigwedge_{i \leq k < n} g_k(\psi_1) \geq v.$$

Since $(**)$ is true for every n, then in particular, we have that $g(\psi_1 \mathsf{U} \psi_2) \geq v$, which concludes the first direction of the proof.

For the second direction, assume that $g(\psi_1 \mathsf{U} \psi_2) \geq v$. If there exists $n \geq i$ such that $\bigvee_{i \leq j \leq n} \left(g_j(\psi_2) \wedge \bigwedge_{i \leq k < j} g_k(\psi_1) \right) \geq v$, then we are done. Assume by way of contradiction that there is no such n. Thus, by $(**)$, for every $n \geq i$ it holds that $g_{n+1}(\psi_1 \mathsf{U} \psi_2) \wedge \bigwedge_{i \leq k \leq n} g_k(\psi_1) \geq v$, which means that $g_{n+1}(\psi_1 \mathsf{U} \psi_2) \geq v$ and $\bigwedge_{i \leq k \leq n} g_k(\psi_1) \geq v$. In particular, there cannot exist $n \geq i$ such that $g_n(\psi_2) \geq v$, otherwise it would contradict our assumption.

Since the run is accepting, there exists $n_1 > i$ such that $f_{n_1}(\psi_1 \mathsf{U} \psi_2) = \emptyset$. Consider the suffix of the run starting from $n_1 + 1$. For every $t \geq n_1$, We have that $g_t(\psi_1 \mathsf{U} \psi_2) \geq v$ but $g_t(\psi_2) \not\geq v$. Thus, $v \in \mathrm{JI}(f_{n+1}(\psi_1 \mathsf{U} \psi_2))$, and v will never be removed from the f component, this is in contradiction to the fact that the run is accepting, and we are done. □

Example 4 We demonstrate the construction presented in Theorem 8 with a partial example. Consider the formula $\varphi = \mathsf{F} p = \{a, b, c\} \mathsf{U} p$ over the subset lattice $\{2^{\{a,b,c\}}, \subseteq\}$ and the predicate $P = \{\{a, b\}\}$. An example of an initial state of $\mathcal{A}_{\varphi,P}$ is a state in which $g(\varphi) = \{a, b\}$, $f(\varphi) = \{a\}$ and $g(p) = \{b\}$. That is, we expect that in the next step, the value of p will be $\{b, c\}$, which means that in order for φ to get value $\{a, b\}$ it remains for p to get value $\{a\}$. Then, upon reading value $\{b\}$ for p, a possible transition is to a state where $g(\varphi) = \{a, b\}$, $f(\varphi) = \emptyset$ and $g(p) = \{a\}$, which is an accepting state.

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 8 in order to solve both variants of the synthesis problem. In addition, we describe an extension to a probabilistic setting in which both the noise the realizability criteria are probabilistic.

5.1 Solving the LLTL synthesis problem

We start with the non-noisy case. Here, the algorithm is similar to the one developed for the Boolean setting, except that the parity game is obtained from LLTL formulas:

Theorem 9 *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 8 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 $m^{O(n)}$ (Safra 1992; Piterman 2006). 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 $\mathcal{D}_{\varphi,P}$, 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

We now turn to the noisy case. 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 2 *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 10 *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 8. 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 2 to $\mathcal{D}_{\varphi,P}$ and obtain a UGCW $\mathcal{D}'_{\varphi,P}$ that accepts a computation 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 8, 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 3 **[A Safraless approach]** The approach described in the proofs of Theorems 1, 9, and 10 is *Safraful*, in the sense it involves a construction of a DPW. As has been the case with Boolean synthesis (Kupferman and Vardi 2005), 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 2 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.

Remark 4 **[Noisy output signals]** Noisy input signals are often considered in settings where the output can also be perturbed. Indeed, assuming the system and the environment interact along noisy channels, there is no reason to assume that the output signals are immunized against noise. In our setting, we can consider the case where there is also a noise function on the outputs, and the goal is to synthesize a transducer whose output, after perturbation, realizes the specification. More formally, no matter how the output is perturbed (assuming a known noise function), the generated computation should satisfy the specification.

Going over the proof of Lemma 2, it is not hard to see that a similar construction can be applied to the outputs, which would enable us to use the construction in Theorem 10 and solve the noisy-synthesis problem in the presence of noisy outputs.

5.3 Probabilistic noise

Our definition of noisy synthesis takes a worst-case approach. Indeed, a transducer does not realize a specification even if there is a single input sequence for which the resulting noisy computation does not satisfy the specification. In practice, however, noise typically occurs according to some probability, for which the worst-case approach may not be appropriate. Consider, for example, a noisy channel that may flip a bit with some positive probability p. A corresponding noise function should allow the input 1 (that is, an input signal that takes values in $\{0, 1\}$ and whose actual value is 1) to be read as 0 with some positive probability. Then, no non-trivial specification is realizable, as all computations are indistinguishable.

In practice, however, the event where a computation differs significantly from its noisy version has low probability. Thus, we can relax the worst-case approach, and require that given a noise function with the respective noise probabilities, we synthesize a transducer that realizes the specification with maximal probability.

Formally, a *probabilistic noise function* is $\nu : \mathcal{L}^I \to \Delta(\mathcal{L}^I)$ where $\Delta(\mathcal{L}^I) = \{f : \mathcal{L}^I \to [0, 1] : \sum_{i \in \mathcal{L}^I} f(i) = 1\}$ is the set of probability functions over $\mathcal{L}^I$. Intuitively, for inputs $i, j \in \mathcal{L}^I$, we have that $\nu(i)(j)$ is the probability that input j is read by the system when the environment generates input i. Given an input sequence $\pi \in (\mathcal{L}^I)^\omega$, we obtain from ν a probability distribution $\Pr_{\nu,\pi}$ over $(\mathcal{L}^I)^\omega$ in the standard manner, by considering cylinder sets (see e.g., (Baier et al. 2014)).

Consider an LLTL formula φ over $I \cup O$, a predicate $P \subseteq \mathcal{L}$, a probabilistic noise function ν, a transducer $\mathcal{T}$, and a threshold $t \in [0, 1]$. We say that $\mathcal{T}$ *realizes* $\langle \varphi, P \rangle$ *with probability* t *under noise* ν if for every computation $\pi \in \mathcal{L}^I$, we have that the probability that $\mathcal{T}$ realizes φ with a value in P, given a computation π' distributed according to $\Pr_{\nu,\pi}$, is at least t.

The *Probabilistic noisy-synthesis problem* for LLTL is then to synthesize, given φ, P, and ν as above, a transducer $\mathcal{T}$ that maximizes the value t for which $\mathcal{T}$ realizes $\langle \varphi, P \rangle$ with probability t under noise ν.

Theorem 11 *The probabilistic noisy-synthesis problem for* LLTL *is in 2NEXPTIME ∩ co-2NEXPTIME. Given an* LLTL *formula* φ *over a lattice* $\mathcal{L}$*, a predicate* $P \subseteq \mathcal{L}$*, and a probabilistic noise function* ν*, we can solve the probabilistic noisy-synthesis problem for* $\langle \varphi, P \rangle$ *and* ν *in time* $2^{m^{O(n)}}$.

Proof Similarly to Theorem 9, we start by obtaining from φ and P a DPW $\mathcal{D}_{\varphi,P}$ of size $2^{m^{O(n)}}$ and index $m^{O(n)}$, where m is the size of $\mathcal{L}$, and n is the length of φ. We then obtain from $D_{\varphi,P}$ a parity game $\mathcal{G}_{\varphi,P} = \langle 2^O, 2^I, Q, q_0, \eta, \alpha \rangle$ as per the proof of Theorem 1. We now incorporate the noise function ν into $\mathcal{G}_{\varphi,P}$ as follows. Recall that $\eta : Q \times 2^I \times 2^O \to Q$ is such that $\eta(q, i, o) = \delta(q, i \cup o)$, where Q and δ are the states and transition function of $\mathcal{D}_{\varphi,P}$, respectively. We define a $2\frac{1}{2}$ *player parity game* (Chatterjee et al. 2004) $\mathcal{D}'_{\varphi,P,\nu} = \langle 2^O, 2^I, Q, q_0, \eta', \alpha \rangle$, where $\eta' : Q \times 2^O \times 2^I \to \Delta(Q)$ is the probabilistic transition function given by setting $\eta'(q, o, i)$ to be the distribution that assigns to the state $\delta(q, i' \cup o)$ the probability $\nu(i)(i')$ that input i' is read when the environment generates input i.

It is not hard to see that a strategy for the system that wins with probability t corresponds to a transducer that realizes $\langle \varphi, P \rangle$ with probability t under noise ν. In Chatterjee et al. (2004), the authors prove that the system player in a $2\frac{1}{2}$-game has a deterministic and memoryless strategy that maximizes the probability of winning, and that finding it can be done in NP∩coNP. Such a strategy corresponds to a transducer that maximizes the probability of realizing φ with a value in P. Since the size of the game is doubly exponential, we conclude that the probabilistic-noise synthesis problem is in 2NEXPTIME∩co-2NEXPTIME. □

6 Local stability revisited

In Section 3.2 we studied stability and proved that not all LLTL formulas are locally stable (see Theorem 7). 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 12 *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 8 we showed how to construct an 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.

As discussed above, this suggest a PSPACE algorithm for deciding local stability. We now complete the picture by presenting a matching lower bound.

We prove hardness 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. We now show that φ is satisfiable iff ψ is not locally stable.

Consider φ as an LLTL formula over $\mathcal{L}$. We observe that if there exists a computation π such that $[\![\pi, \varphi]\!] = \{a\}$, then there exists a computation τ such that $[\![\tau, \varphi]\!] = \{b\}$. Indeed, the lattice $\mathcal{L}$ is symmetric and φ does not contain elements of the form $\{a\}$ or $\{b\}$ to break the symmetry. Thus, we can obtain τ by swapping the roles of a and b in π. From Theorem 3 we know that $\mathcal{L}$ is pointed, so in this case there also exists a computation ρ such that $[\![\rho, \varphi]\!] = \{a, b\}$.

We first prove that if φ is not satisfiable, then ψ is locally stable. Observe that if we view φ as an LLTL formula and there exists a computation π such that $[\![\pi, \varphi]\!] = \{a\}$, then φ is satisfiable as an LTL formula. Indeed, a satisfying computation π' for φ can be obtained

from π by defining $p \in \pi'_i$ iff $a \in \pi_i(p)$ for all $p \in AP$ and $i \geq 0$. Thus, if φ is not satisfiable, then $[\![\pi, \varphi]\!] = \emptyset$ for every computation $\pi \in (\mathcal{L}^{AP})^\omega$, and similarly $[\![\pi', \psi]\!] = \emptyset$ for every $\pi' \in (\mathcal{L}^{AP \cup AP'})^\omega$. Hence, ψ is locally stable.

For the second direction, assume that φ is satisfiable. Thus, there exists a computation $\pi \in (2^{AP})^\omega$ such that $\pi \models \varphi$. By our assumption, there also exists a computation π' such that $\pi' \not\models \varphi$. It is easy to see that by identifying `True` with $\{a, b\}$ and `False` with $\emptyset$, we get $[\![\pi, \varphi]\!] = \{a, b\}$ and $[\![\pi', \varphi]\!] = \emptyset$. For a computation $w \in (\mathcal{L}^{AP})^\omega$, let $\widehat{w} \in (\mathcal{L}^{AP \cup AP'})^\omega$ be the computation obtained from w by copying the behavior of the atoms in AP to their tagged atoms. Thus, for all $i \geq 0$, we have $p, p' \in \widehat{w}_i$ iff $p \in w_i$.

Since the maximal distance between elements in $\mathcal{L}$ is 2, there exists a computation τ such that $ld(\pi', \tau) \leq 1$ and $ld(\tau, \pi) \leq 1$. That is, we can "get" from π' to π by two local changes of 1. From this follows that $ld(\widehat{\pi'}, \widehat{\tau}) \leq 1$ and $ld(\widehat{\tau}, \widehat{\pi}) \leq 1$. Consider $[\![\tau, \varphi]\!]$. If $[\![\tau, \varphi]\!] = \{a, b\}$, then ψ is not locally stable, since $ld(\widehat{\pi'}, \widehat{\tau}) \leq 1$ but $d([\![\widehat{\tau}, \psi]\!], [\![\widehat{\pi'}, \psi]\!]) = 2$. Similarly, if $[\![\tau, \varphi]\!] = \emptyset$, then ψ is not locally stable. Otherwise, w.l.o.g $[\![\tau, \varphi]\!] = \{a\}$. Then, there exists a computation τ' such that $[\![\tau', \varphi]\!] = \{b\}$. The computation τ' is obtained by swapping a and b in τ. Observe that since π' contains only symmetric elements from $\mathcal{L}$ (i.e. $\emptyset$ and $\{a, b\}$), then $ld(\pi', \tau) = ld(\pi', \tau') \leq 1$. Finally, since AP and AP' are disjoint, then by assigning τ over AP and τ' over AP', we obtain a computation ρ such that $ld(\widehat{\pi'}, \rho) \leq 1$ but $[\![\rho, \psi]\!] = \{a\} \vee \{b\} = \{a, b\}$, and ψ is not locally stable. □

7 Discussion

We introduced and studied the noisy-synthesis problem, where we seek a transducer that realizes a multi-valued specification in LLTL in the highest possible satisfaction value, in the presence of noisy input signals. Our study includes relevant properties of multi-valued specification formalisms, like their global and local stability, which essentially measure the sensitivity of the satisfaction value of specifications to perturbations on the input signals. We prove that the noisy-synthesis problem for LLTL is 2EXPTIME-complete, as is traditional LTL synthesis.

Our future research includes the following directions. (1) The logic LLTL is a multi-valued logic in which values are taken from and manipulated according to a lattice. While many scenarios can be captured by the two lattices on which we focus, there is growing interest in weighted formalisms and multi-valued temporal logics that can express rich behaviors over a wide range of domains. For example, the multi-valued logics LTL[$\mathcal{F}$] and LTL$^{\text{disc}}$[$\mathcal{D}$] (Almagor et al. 2016) can express the quantitative propositional and temporal aspects of computations, and can, for example, prioritize different satisfaction possibilities or refer to the waiting time to the satisfaction of eventualities. The logics can be interpreted with respect to both Boolean and multi-valued computations. From a technical point of view, our solution to the noisy-synthesis problem is based on a translation of LLTL formulas to automata. For other multi-valued logics, and in particular for LTL^{disc}[$\mathcal{D}$], where the set of possible satisfaction values is not bounded, such a translation is not always possible. We plan to investigate the extension of the noisy-synthesis problem to richer multi-valued logics, and in particular to LTL[$\mathcal{F}$] and LTL^{disc}[$\mathcal{D}$]. (2) In some settings, the designer can control the noise. Such a control requires resources, say sensors of high quality. In the *budgeted noisy-synthesis problem*, we are given, together with the noise function ν, also a budget and information on the cost of reducing noise. The goal is to use the budget in a way that would generate a transducer with the highest possible satisfaction value. Note that the information about the cost can be of different types. For example, it may refer to the

cost of reducing the noise of a particular input signal in a single state of the transducer, in specific time points, or throughout the computation, and the same for sets of input signals. This is related to the different ways in which the cost of sensing may be defined (Almagor et al. 2015), and also to a formalization of the trade-offs among the noise that the system experiences, its size, and the best satisfaction value that it can guarantee. We plan to relate noisy-synthesis and sensing-cost-aware synthesis, and to study these trade-offs.

References

Almagor S, Boker U, Kupferman O (2013) ForMalizing and reasoning about quality. In: Proceedings of the 40th int. Colloq. on automata, languages, and programming, volume 7966 of lecture notes in computer science, pp 15–27. Springer

Almagor S, Boker U, Kupferman O (2016) Formally reasoning about quality. J ACM 63(3):24

Almagor S, Kuperberg D, Kupferman O (2015) The sensing cost of monitoring and synthesis. In: 35th IARCS annual conference on foundation of software technology and theoretical computer science, FSTTCS 2015, december 16-18, 2015, Bangalore, India, pp 380–393

Alur R, Kanade A, Weiss G (2008) Ranking automata and games for prioritized requirements. In: Proc. 20th int. Conf. on computer aided verification, volume 5123 of lecture notes in computer science, pp 240–253. Springer

Baier C, Klein J, Klüppelholz S, Märcker S (2014) Computing conditional probabilities in markovian models efficiently. In: 20Th TACAS, pp 515–530

Bloem R, Chatterjee K, Henzinger T, Jobstmann B (2009) Better quality in synthesis through quantitative objectives. In: Proc. 21st int. Conf. on computer aided verification, volume 5643 of lecture notes in computer science, pp 140–156. Springer

Cerný P, Henzinger T (2011) From boolean to quantitative synthesis. In: EMSOFT, pp 149–154

Chatterjee K, Doyen L, Henzinger T (2008) Quantative languages. In: Proceedings of the 17th annual conference of the european association for computer science logic, pp 385–400

Chatterjee K, Doyen L, Henzinger TA, Raskin J-F (2006) Algorithms for omega-regular games with imperfect information. In: Proceedings of the 15th annual conference of the european association for computer science logic, volume 4207 of lecture notes in computer science, pp 287–302

Chatterjee K, Jurdzinski M, Henzinger TA (2004) Quantitative stochastic parity games. In: Proceedings of the fifteenth annual ACM-SIAM symposium on discrete algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004, pp 121–130

Chatterjee K, Majumdar R (2011) Minimum attention controller synthesis for omega-regular objectives. In: FORMATS, pp 145–159

Chatterjee K, Majumdar R, Henzinger TA (2008) Controller synthesis with budget constraints. In: Proceedings of the 11th international workshop on hybrid systems: computation and control, volume 4981 of lecture notes in computer science, pp 72–86. Springer

Chechik M, Devereux B, Gurfinkel A (2001) Model-checking infinite state-space systems with fine-grained abstractions using SPIN. In: Proceedings of the 8th international SPIN workshop on model checking software, volume 2057 of lecture notes in computer science, pp 16–36. Springer

Church A (1963) Logic, arithmetics, and automata. In: Proceedings of the international congress of mathematicians, 1962, pp 23–35. Institut Mittag-Leffler

Emerson E, Jutla C (1991) Tree automata, μ-calculus and determinacy. In: Proceedings of the 32nd IEEE symposium on foundations of computer science, pp 368–377

Faella M, Legay A, Stoelinga M (2008) Model checking quantitative linear time logic. Electr Notes Theor Comput Sci 220(3):61–77

Filiot E, Jin N, Raskin J-F (2009) An antichain algorithm for LTL realizability. In: Proceedings of the 21st international conference on computer aided verification, vol 5643, pp 263–277

Huth M, Pradhan S (2004) Consistent partial model checking. Electr Notes Theor Comput Sci 73:45–85

Jurdzinski M, Paterson M, Zwick U (2008) A deterministic subexponential algorithm for solving parity games. SIAM J Comput 38(4):1519–1532

Kumar R, Shayman M (1995) Supervisory control of nondeterministic systems under partial observation and decentralization. SIAM Journal of Control and Optimization

Kupferman O, Lustig Y (2007) Lattice automata. In: Proceedings of the 8th international conference on verification, model checking, and abstract interpretation, volume 4349 of lecture notes in computer science, pp 199–213. Springer
Kupferman O, Lustig Y (2010) Latticed simulation relations and games. Int J Found Comput Sci 21(2):167–189
Kupferman O, Vardi M (1999) Church's problem revisited. Bull Symb Log 5(2):245–263
Kupferman O, Vardi M (2000) Synthesis with incomplete information. In: Advances in temporal logic, pp 109–127. Kluwer Academic Publishers
Kupferman O, Vardi M (2005) Safraless decision procedures. In: Proceedings of the 46th IEEE symposium on foundations of computer science, pp 531–540
Kwiatkowska M (2007) Quantitative verification: models techniques and tools. In: ESEC/SIGSOFT FSE, pp 449–458
Majumdar R, Render E, Tabuada P (2011) Robust discrete synthesis against unspecified disturbances. In: Proceedings of the 14th ACM international conference on hybrid systems: computation and control, HSCC 2011, Chicago, IL, USA, April 12-14, 2011, pp 211–220
Martin D (1975) Borel determinacy. Ann Math 65:363–371
Miyano S, Hayashi T (1984) Alternating finite automata on ω-words. Theor Comput Sci 32:321–330
Piterman N (2006) From nondeterministic büchi and Streett automata to deterministic parity automata. In: Proceedings of the 21st IEEE symposium on logic in computer science, pp 255–264. IEEE press
Pnueli A, Rosner R (1989) On the synthesis of a reactive module. In: Proceedings of the 16th ACM symposium on principles of programming languages, pp 179–190
Pnueli A, Rosner R (1989) On the synthesis of an asynchronous reactive module. In: Proceedings of the 16th int. Colloq. on automata, languages, and programming, volume 372 of lecture notes in computer science, pp 652–671. Springer
Reif J (1984) The complexity of two-player games of incomplete information. J Comput Syst Sci 29:274–301
Safra S (1992) Exponential determinization for ω-automata with strong-fairness acceptance condition. In: Proceedings of the 24th ACM symposium on theory of computing
Schewe S (2007) Solving parity games in big steps. In: Proceedings of the 27th conference on foundations of software technology and theoretical computer science, pp 449–460
Topcu U, Ozay N, Liu J, Murray RM (2012) On synthesizing robust discrete controllers under modeling uncertainty. In: Hybrid systems: computation and control (part of CPS week 2012), HSCC'12, Beijing, China, April 17-19, 2012, pp 85–94
Vardi M (2008) From verification to synthesis. In: Proceedings of the 2nd international conference on verified software: theories, tools, experiments, volume 5295 of lecture notes in computer science, p 2. Springer
Vardi M, Wolper P (1994) Reasoning about infinite computations. Inf Comput 115(1):1–37
Velner Y, Rabinovich A (2011) Church synthesis problem for noisy input. In: Proceedings of the 14th international conference on foundations of software science and computation structures, pp 275–289

Shaull Almagor is a research assistant in the Computer Science department at Oxford University. His research interests include theoretical aspects of formal verification, and dynamical systems. Dr. Almagor received his PhD from The Hebrew University in 2016, and has joined Oxford University the same year.

Orna Kupferman is a professor in the School for Computer Science and Engineering at The Hebrew University. She researches the theoretical aspects of formal verification. Prof. Kupferman graduated from The Technion at 1995 and joined The Hebrew University at 1998. She was a visiting researcher in Bell Laboratories, UC Berkeley, Microsoft Research, Cadence Berkeley Labs, and Rice University.