Synthesis from Weighted Specifications with Partial Domains over Finite Words

Emmanuel Filiot
Université libre de Bruxelles, Belgium
efiliot@ulb.ac.be

Christof Löding
RWTH Aachen University, Germany
loeding@cs.rwth-aachen.de

Sarah Winter
Université libre de Bruxelles, Belgium
swinter@ulb.ac.be

Abstract

In this paper, we investigate the synthesis problem of terminating reactive systems from quantitative specifications. Such systems are modeled as finite transducers whose executions are represented as finite words in $(\Sigma_{\mathtt{i}} \times \Sigma_{\mathtt{o}})^*$, where $\Sigma_{\mathtt{i}}$, $\Sigma_{\mathtt{o}}$ are finite sets of input and output symbols, respectively. A weighted specification S assigns a rational value (or $-\infty$) to words in $(\Sigma_{\mathtt{i}} \times \Sigma_{\mathtt{o}})^*$, and we consider three kinds of objectives for synthesis, namely threshold objectives where the system's executions are required to be above some given threshold, best-value and approximate objectives where the system is required to perform as best as it can by providing output symbols that yield the best value and ε-best value respectively w.r.t. S. We establish a landscape of decidability results for these three objectives and weighted specifications with partial domain over finite words given by deterministic weighted automata equipped with sum, discounted-sum and average measures. The resulting objectives are not regular in general and we develop an infinite game framework to solve the corresponding synthesis problems, namely the class of (weighted) critical prefix games.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Transducers; Theory of computation → Quantitative automata

Keywords and phrases synthesis, weighted games, weighted automata on finite words

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2020.46

Funding *Emmanuel Filiot*: This work is partially supported by the MIS project F451019F (F.R.S.-FNRS). Emmanuel Filiot is a research associate at F.R.S.-FNRS.

1 Introduction

Reactive synthesis. The goal of automatic synthesis is to automatically construct programs from specifications of correct pairs of input and output. The goal is to liberate the developer from low-level implementation details, and to automatically generate programs which are correct by construction. In the automata-based approach to synthesis [14, 20], the programs to be synthesized are finite-state reactive programs, which react continuously to stimuli received from an environment. Such systems are not assumed to terminate and their executions are modeled as ω-words in $(\Sigma_{\mathtt{i}}\Sigma_{\mathtt{o}})^\omega$, alternating between input symbols in $\Sigma_{\mathtt{i}}$ and output symbols in $\Sigma_{\mathtt{o}}$. Specifications of such systems are then languages $S \subseteq (\Sigma_{\mathtt{i}}\Sigma_{\mathtt{o}})^\omega$ representing the set of acceptable executions. The synthesis problem asks to check whether there exists a total synchronous[1] function $f\colon \Sigma_{\mathtt{i}}^\omega \to \Sigma_{\mathtt{o}}^\omega$ such that for all input sequences $u = i_0 i_1 \dots$, there exists

[1] $f\colon \Sigma_{\mathtt{i}}^\omega \to \Sigma_{\mathtt{o}}^\omega$ is synchronous if it is induced by a strategy $s\colon \Sigma_{\mathtt{i}}^+ \to \Sigma_{\mathtt{o}}$ in the sense that $f(i_0 i_1 \dots) = s(i_0)s(i_0 i_1)s(i_0 i_1 i_2)\dots$ for all $i_0 i_1 \dots \in \Sigma^\omega$

40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020).
Editors: Nitin Saxena and Sunil Simon; Article No. 46; pp. 46:1–46:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

an output sequence $v = o_0 o_1 \dots$ such that $f(u) = v$ and the convolution $u \otimes v = i_0 o_0 i_1 o_1 \dots$ belongs to S. The function f is called a realizer of S. Automatic synthesis of non-terminating reactive systems has first been introduced by Church [19], and a first solution has been given by Büchi and Landweber [14] when the specification S is ω-regular. In this setting, when a realizer exists, there is always one which can be computed by a finite-state sequential transducer, a finite-state automaton which alternates between reading one input symbol and producing one output symbol. This result has sparked much further work to make synthesis feasible in practice, see e.g., [31, 27, 7]. The synthesis problem is classically modeled as an infinite-duration game on a graph, played by two players, alternatively picking input and output symbols. One player, representing the system, must enforce an objective that corresponds to the specification. Finite-memory winning strategies are in turn systems that realize the specification. This game metaphor has triggered a lot of research on graph games [20, Chapter 27]. There has also been a recent effort to increase the quality of the automatically generated systems by enhancing Boolean specifications with quantitative constraints, e.g., [5, 16, 12, 2]. This has also triggered a lot of research on quantitative extensions of infinite-duration games, for example mean-payoff, energy, and discounted-sum games, see, e.g., [24, 34, 22, 10, 11, 4, 29].

Partial-domain specifications. In the classical formulation of the synthesis problem, it is required that a realizer f meets the specification *for all* possible input sequences. In particular, if there is a single input sequence u such that $u \otimes v \notin S$ for all output sequences v, then S admits no realizer. In other words, when the *domain* of S is partial, then S is unrealizable. Formally, the domain of S is $\mathrm{dom}(S) = \{u \in \Sigma_{\mathtt{i}}^\omega \mid \exists v \colon u \otimes v \in S\}$. As noticed recently and independently in [1], asking that the realizer meets the specification for all input sequences is often too strong and a more realistic setting is to make some assumptions on the environment's behaviour, namely, that the environment plays an input sequence in the domain of the specification. This problem is called good-enough synthesis in [1] and can be formulated as follows: given a specification S, check whether there exists a *partial* synchronous function $f \colon \Sigma_{\mathtt{i}}^\omega \to \Sigma_{\mathtt{o}}^\omega$ whose domain is $\mathrm{dom}(S)$, and such that for all input sequence $u \in \mathrm{dom}(S) = \mathrm{dom}(f)$, $u \otimes f(u) \in S$. Decidability of the latter problem is entailed by decidability of the classical synthesis problem when the specification formalism used to describe S is closed under expressing the assumption that the environment provides inputs in $\mathrm{dom}(S)$. It is the case for instance when S is ω-regular, because the specification $S \cup \overline{\mathrm{dom}(S)} \otimes \Sigma_{\mathtt{o}}^\omega$ has total domain and is effectively ω-regular. [1] investigates the more challenging setting of S being expressed by a multi-valued (in contrast to Boolean) LTL logic. More generally, there is a series of works on solving games under assumptions on the behaviour of the environment [18, 6, 30, 21, 13, 2].

Our setting: Partial-domain weighted specifications. In this paper, motivated by the line of work on quantitative extensions of synthesis and the latter more realistic setting of partial-domain specifications, we investigate synthesis problems from partial-domain weighted specifications (hereafter just called weighted specifications). We conduct this investigation in the setting of *terminating reactive systems*, and accordingly our specifications are over finite words. Formally, a specification is a mapping $S \colon (\Sigma_{\mathtt{i}}.\Sigma_{\mathtt{o}})^* \to \mathbb{Q} \cup \{-\infty\}$. The domain $\mathrm{dom}(S)$ of S is defined as all the input sequences $u \in \Sigma_{\mathtt{i}}^*$ such that $S(u \otimes v) \in \mathbb{Q}$ for some $v \in \Sigma_{\mathtt{o}}^*$. We consider three quantitative synthesis problems, which all consists in checking whether there exists a function f computable by a finite transducer such that $\mathrm{dom}(f) = \mathrm{dom}(S)$ and which satisfies respectively the following conditions:

Table 1 Complexity results for weighted specifications. Here, D stands for decidable, the suffix -c for complete, λ for discount factor, and n for a natural number.

Problem \ Spec	Sum-automata	Avg-automata	Dsum-automata
strict threshold	NP ∩ coNP	NP ∩ coNP	NP
non-strict threshold	NP ∩ coNP	NP ∩ coNP	NP ∩ coNP
best-value	PTIME [3]	PTIME [3]	NP ∩ coNP
strict approximate	EXPTIME-c [26]	D	NEXPTIME for $\lambda = 1/n$
non-strict approx.	EXPTIME-c [26]	D	EXPTIME for $\lambda = 1/n$

- for all $u \in \mathrm{dom}(S)$ it holds that $S(u \otimes f(u)) \triangleright t$ for a given threshold $t \in \mathbb{Q}$ and $\triangleright \in \{>, \geq\}$, called *threshold synthesis*, or
- $S(u \otimes f(u)) = \mathsf{bestVal}_S(u)$, that is, the maximal value that can be achieved for the input u, i.e., $\mathsf{bestVal}_S(u) = \sup\{S(u \otimes v) \mid v \in \Sigma_{\mathtt{o}}^*\}$, called *best-value synthesis*, or
- $\mathsf{bestVal}_S(u) - S(u \otimes f(u)) \triangleleft r$ for a given threshold $r \in \mathbb{Q}$ and $\triangleleft \in \{<, \leq\}$, called *approximate synthesis*.

Following the game metaphor explained before, those quantitative synthesis problems can be formulated as two-player games in which Adam (environment) and Eve (system) alternatively pick symbols in $\Sigma_{\mathtt{i}}$ and $\Sigma_{\mathtt{o}}$ respectively. Additionally, Adam has the power to stop the game. If it does not, then Eve wins the game. Otherwise, a finite play spells a word $u \otimes v$. For the Boolean synthesis problem, Eve has won if either $u \notin \mathrm{dom}(S)$ where S is the specification, or $u \otimes v \in S$. Additionally, for the threshold synthesis problem, the value $S(u \otimes v)$ must be greater than the given threshold; for the best-value synthesis problem, it must be equal to $\mathsf{bestVal}_S(u)$ and for approximate synthesis it must be r-close to $\mathsf{bestVal}_S(u)$.

Contributions. Our main contribution is a clear picture about decidability of threshold synthesis, best-value synthesis and approximate synthesis for weighted specifications over finite words defined by deterministic weighted finite automata [23], equipped with either sum, average or discounted-sum measure. Such automata extend finite automata with integer weights on their transitions, computing a value through a payoff function that combines those integers, with sum, average, or discounted-sum. The results (presented in Section 4) are summarized in Table 1. We also give an application of our results to the decidability of quantitative extensions of the Church synthesis problem over infinite words, for some classes of weighted safety specifications, which intuitively require that all prefixes satisfy a quantitative requirement (being above a threshold, equal to the best-value, or close to it).

As we explain in the related works section, some of our results are obtained via reduction to solving known quantitative games or to the notions of r-regret determinization for weighted automata. We develop new techniques to solve the strict threshold synthesis problem for discounted-sum specifications in NP (Theorem 9), the best-value synthesis problem for discounted-sum specifications in NP ∩ coNP (Theorem 12) and approximate synthesis for average specifications (Theorem 13), which are to the best of our knowledge new results.

Moreover, as our main tool to obtain our synthesis results, we introduce in Section 3 a new kind of (weighted) games called *critical prefix games* tailored to handle weighted specifications with *partial* domain of *finite* words. We believe these kind of games are interesting on their own and are described below in more detail.

Critical prefix games. Following the classical game metaphor of synthesis, we design weighted games into which some of our synthesis problems can be directly encoded. Those games still have infinite-duration, but account for the fact that specifications are on finite words and have partial domains. In particular, the quantitative constraints must be checked only for play prefixes that correspond to input words of the environment which are in the domain of the specification. So, a critical prefix game is defined as a two-player turn-based weighted game with some of the vertices being declared as critical. When the play enters a critical vertex, a quantitative requirement must be fulfilled, otherwise Eve loses. For instance, critical prefix *threshold* games require that the payoff value when entering a critical vertex is at least or above a certain threshold. We show that these threshold games are all decidable for sum, average, and discounted-sum payoffs, see Theorems 3 and 4. For solving approximate average synthesis, we use a reduction to critical prefix energy games of imperfect information starting with fixed initial credit (the energy level must be at least zero whenever the play is in a critical vertex). Without critical vertices (where the energy level must be at least zero all the time) these games are known to be decidable [22]. We show that adding critical vertices makes these games undecidable, in general, see Theorem 7. However, a large subclass of imperfect information critical prefix energy games, sufficient for our synthesis problems, is shown to be decidable, see Theorem 8.

Domain-safe weighted specifications. Most of our quantitative synthesis problems reduce to two-player games. While we need games of different natures, they all model the fact that Eve constructs a run of the (deterministic) automaton, given the input symbols provided by Adam so far. By choosing outputs, Eve must make sure that this run is accepting whenever the input word played by Adam so far is in the domain of S. Otherwise Adam can stop and Eve loses. While this condition can be encoded in the game by enriching the vertices with subsets of states (in which Eve could have been by choosing alternative output symbols), this would result in an exponential blow-up of the game. We instead show that the weighted automaton can be preprocessed in polynomial-time into a so called domain-safe automaton, in which there is no need to monitor the input domain when playing, see Theorem 2.

Related works. Boolean synthesis problems for finite words have been considered in [33, 32] where the specification is given as an LTL formula over finite traces. In the quantitative setting, it has also been considered in [25] for weighted specifications given by deterministic weighted automata. In these works however, it is the role of Eve to eventually stop the game. While this makes sense for reachability objectives and planning problems, this setting does not accurately model a synthesis scenario where the system has no control over the provided input sequence. Our setting is different and needs new technical developments.

Threshold problems in quantitative infinite-duration two-player games with discounted- and mean-payoff measures are known to be solvable in $\mathrm{NP} \cap \mathrm{coNP}$ [4, 34]. Our threshold synthesis problems all directly reduce to critical prefix threshold games with corresponding payoff functions. The latter games, for sum and average, are shown to reduce to mean-payoff games, so our $\mathrm{NP} \cap \mathrm{coNP}$ upper-bound follows from [34]. For critical prefix discounted-sum games with a non-strict threshold, we show a polynomial time reduction to infinite-duration discounted sum games and hence our result follows from [4]. Such a reduction fails for a strict threshold and we develop new techniques to solve critical prefix discounted-sum games with strict threshold, by first showing that memoryless strategies suffice for Eve to win, and then by showing how to check in PTime whether a memoryless strategy is winning for Eve. The latter result actually shows how to test in PTime whether there exists,

in a weighted graph, a path from a source to a target vertex of discounted-sum greater or equal to some given threshold. This result entails that the non-emptiness problem for non-deterministic discounted-sum max-automata[2] is solvable in PTime (Theorem 6). To the best of our knowledge, up to now this problem is only known to be in PSpace for the subcase of functional discounted-sum automata [25, 9].

As we show, the best-value synthesis problems correspond to zero-regret determinization problems for non-deterministic weighted automata, i.e., deciding whether there is a non-determinism resolving strategy for Eve that guarantees the same value as the maximal value of an accepting run in the non-deterministic weighted automaton. Such a problem is in PTime for sum-automata [3] and the average case easily reduces to the sum-case. For discounted-sum, zero-regret determinization is known to be decidable in NP for dsum-automata over infinite words [29]. We improve this bound to NP $\cap$ coNP for finite words.

Finally, approximate synthesis corresponds to a problem known as r-regret determinization of non-deterministic weighted automata. For sum-automata, it is known to be ExpTime-complete [26]. For average-automata, there is no immediate reduction to the sum case, because the sum value computed by an r-regret determinizer can be arbitrarily faraway from the best sum, while its averaged value remains close to the best average. Instead, we show a reduction to the new class of partial observation critical prefix energy games. For dsum-automata over infinite words, total domain and integral discount factor, r-regret determinization is known to be decidable [29]. Our setting does not directly reduce to this setting, but we use similar ideas.

2 Preliminaries

Languages and relations. Let $\mathbb{N}$ be the set of non-negative integers. Let Σ be a finite alphabet. We denote by Σ^*, respectively Σ^ω, the set of finite, respectively infinite, words over Σ, and Σ^+ the set of non-empty finite words over Σ. The empty word is denoted by ε. A *language* over Σ is a set of words over Σ. A (binary) *relation* R is a subset of $\Sigma_{\mathtt{i}}^* \times \Sigma_{\mathtt{o}}^*$, i.e., a set of pairs of words. Its domain is the set $\mathrm{dom}(R) = \{u \mid \exists v\colon (u,v) \in R\}$. Given a pair of words, we refer to the first (resp. second) component as input (resp. output) component, the alphabets $\Sigma_{\mathtt{i}}$ and $\Sigma_{\mathtt{o}}$ are referred to as input resp. output alphabet. We let $\Sigma_{\mathtt{io}} = \Sigma_{\mathtt{i}} \cup \Sigma_{\mathtt{o}}$.

Automata. A *nondeterministic finite state automaton (NFA)* is a tuple $\mathcal{A} = (Q, q_i, \Sigma, \Delta, F)$, where Q is a finite state set, $q_i \in Q$ is the initial state, Σ is a finite alphabet, $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation, and $F \subseteq Q$ is a set of final states. A *run* of the automaton on a word $w = a_1 \dots a_n$ is a sequence $\rho = \tau_1 \dots \tau_n$ of transitions such that there exist $q_0, \dots, q_n \in Q$ such that $\tau_j = (q_{j-1}, a_j, q_j)$ for all j. A run on ε is a single state. A run is *accepting* if it begins in the initial state and ends in a final state. The *language recognized* by the automaton is defined as $L(\mathcal{A}) = \{w \mid \text{there is an accepting run of } \mathcal{A} \text{ on } w\}$. The automaton is *deterministic (a DFA)* if Δ is given as a partial function $\delta : Q \times \Sigma \to Q$.

Transducers. A *transducer* is a tuple $\mathcal{T} = (Q, q_i, \Sigma_{\mathtt{i}}, \Sigma_{\mathtt{o}}, \delta, F)$, where Q is a finite state set, $q_i \in Q$ is the initial state, $\Sigma_{\mathtt{i}}$ and $\Sigma_{\mathtt{o}}$ are finite alphabets, $\delta\colon \big(Q \times \Sigma_{\mathtt{i}}\big) \to \big(\Sigma_{\mathtt{o}} \times Q\big)$ is a transition function, and $F \subseteq Q$ is a set of final states. A transition is also denoted as a tuple for convenience. A *run* is either a non-empty sequence of transitions $\rho = (q_0, u_1, v_1, q_1)(q_1, u_2, v_2, q_2) \dots (q_{n-1}, u_n, v_n, q_n)$ or a single state. The *input (resp. output)* of

[2] i.e., checking whether there exists a word with value greater or equal to some threshold, where the value is defined by taking the max over all accepting runs.

ρ is $u = u_1 \dots u_n$ (resp. $v = v_1 \dots v_n$) if $\rho \in \Delta^+$, both are ε if $\rho \in Q$. We denote by $p \xrightarrow{u|v} q$ that there exists a run from p to q with input u and output v. A run is *accepting* if it starts in the initial and ends in a final state. The *partial function recognized* by the transducer is $f_{\mathcal{T}} \colon \Sigma_{\mathtt{i}}^* \to \Sigma_{\mathtt{o}}^*$ defined as $f_{\mathcal{T}}(u) = v$ if there is an accepting run of the form $p \xrightarrow{u|v} q$.

Weighted automata. Let $n > 0$. Given a finite sequence $\phi = j_1 \dots j_n$ of integers, and a discount factor $\lambda \in \mathbb{Q}$ such that $0 < \lambda < 1$, we define the following functions: $\mathsf{Sum}(\phi) = \sum_{i=1}^{n} j_i$, $\mathsf{Avg}(\phi) = \frac{\mathsf{Sum}(\phi)}{n}$, $\mathsf{Dsum}(\phi) = \sum_{i=1}^{n} \lambda^i j_i$ if ϕ is non-empty and $\mathsf{Sum}(\phi) = \mathsf{Avg}(\phi) = \mathsf{Dsum}(\phi) = 0$ otherwise. Let $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}\}$. A *weighted V-automaton (WFA)* is a tuple $\mathcal{A} = (Q, \Sigma, q_i, \Delta, F, \gamma)$, where $(Q, \Sigma, q_i, \Delta, F)$ is a classical *deterministic* finite state automaton, and $\gamma \colon \delta \to \mathbb{Z}$ is a *weight function*. Its recognized language, etc., is defined as for classical finite state automata. The value $V(\rho)$ of a run $\rho = \tau_1 \dots \tau_n$ is defined as $V(\gamma(\tau_1) \dots \gamma(\tau_n))$ if ρ is accepting and $-\infty$ otherwise. The value $\mathcal{A}(w)$ of a word w is given by the total function, called the function *recognized* by $\mathcal{A}$, $\mathcal{A} \colon \Sigma^* \to \mathbb{Q} \cup \{-\infty\}$ defined as $w \mapsto V(\rho)$, where ρ is the run of $\mathcal{A}$ on w, that is, the value of a word is the value of its accepting run, or $-\infty$ if there exists none.

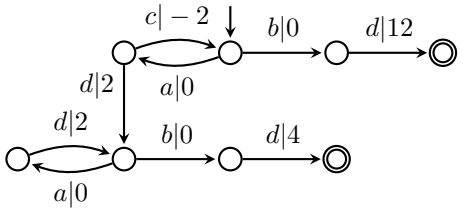
Weighted specifications. A *weighted specification* is a total function $S \colon (\Sigma_{\mathtt{i}}\Sigma_{\mathtt{o}})^* \to \mathbb{Q} \cup \{-\infty\}$ recognized by a WFA $\mathcal{A}$. Note that by our definition, $\mathcal{A}$ is deterministic by default. Given $u = u_1 \dots u_n \in \Sigma_{\mathtt{i}}^*$ and $v = v_1 \dots u_n \in \Sigma_{\mathtt{o}}^*$, $u \otimes v$ denotes its *convolution* $u_1 v_1 \dots u_n v_n \in (\Sigma_{\mathtt{i}}\Sigma_{\mathtt{o}})^*$. We usually write $S(u \otimes v)$ instead of $S(u_1 v_1 \dots u_n v_n)$. The relation (or Boolean specification) of S, denoted by $R(S)$, is given by the set of pairs that are mapped to a rational number, i.e., $R(S) = \{(u, v) \mid S(u \otimes v) > -\infty\}$. We usually write $u \otimes v \in S$ instead of $(u, v) \in R(S)$. The domain of S, denoted by $\mathrm{dom}(S)$, is defined as $\{u \in \Sigma_{\mathtt{i}}^* \mid \exists v \in \Sigma_{\mathtt{o}}^* \colon u \otimes v \in S\}$. If a weighted specification is given by some V-automaton, we refer to it as V-specification.

Quantitative synthesis problems. The (Boolean) *synthesis problem* asks, given a weighted specification S, whether there exists a partial function $f \colon \Sigma_{\mathtt{i}}^* \to \Sigma_{\mathtt{o}}^*$ defined by a transducer with $\mathrm{dom}(f) = \mathrm{dom}(S)$ such that $u \otimes f(u) \in S$ for all $u \in \mathrm{dom}(f)$.

We define three quantitative synthesis problems that pose additional conditions, we only state the additions. The *threshold synthesis problem* additionally asks, given a threshold $\nu \in \mathbb{Q}$, and $\triangleright \in \{>, \geq\}$, that $S(u \otimes f(u)) \triangleright \nu$ for all $u \in \mathrm{dom}(f)$. The *best-value synthesis problem* additionally asks that $S(u \otimes f(u)) = \mathsf{bestVal}_S(u)$, where $\mathsf{bestVal}_S(u) = \sup\{S(u \otimes v) \mid u \otimes v \in S\}$ for all $u \in \mathrm{dom}(f)$. The *approximate synthesis problem* additionally asks, given a threshold $\nu \in \mathbb{Q}$, and $\triangleleft \in \{<, \leq\}$, that $\mathsf{bestVal}_S(u) - S(u \otimes f(u)) \triangleleft \nu$ for all $u \in \mathrm{dom}(f)$.

In these settings, if such a function f exists, it is called *S-realization*, a transducer that defines f is called *S-realizer*, and is said to *implement an S-realization*. A transducer whose implemented function f only satisfies the Boolean condition is called *Boolean S-realizer*.

▶ **Example 1.** Let $\Sigma_{\mathtt{i}} = \{a, b\}$ and $\Sigma_{\mathtt{o}} = \{c, d\}$, and consider the weighted specification S defined by the following automaton $\mathcal{A}$.



Clearly, S has a Boolean realizer (infinitely many, in fact). First, we view $\mathcal{A}$ as a Sum-automaton. There exists a realizer that ensures a value of at least 6, for example, the transducer that always outputs d. There exists no best-value realizer. To see this, we look at the maximal values. We have $\mathsf{bestVal}(b) = 12$, $\mathsf{bestVal}(ab) = 10$, and $\mathsf{bestVal}(a^i b) = 2i + 4$ for $i > 1$. The maximal value for ab is achieved with cd and the maximal value for $aaab$ with $dddd$. So, the first output symbol depends on the length of the input word, which is unknown to a transducer when producing the first output symbol. However, there exists an approximate realizer for the non-strict threshold 4: the transducer that outputs c solely for the first a. The difference to the maximal value is 0 for the inputs b and ab, and 4 for all other inputs. Secondly, we view $\mathcal{A}$ as an Avg-automaton. With the same argumentation as for Sum, it is easy to see that there exists no best-value realizer, there exists an approximate realizer for the non-strict threshold $\frac{2}{3}$: the transducer that outputs c solely for the first a. The difference to the maximal value is 0 for the inputs b and ab, and $\frac{2}{i+1}$ for inputs of the form $a^i b$ for $i > 1$. Note that the difference decreases with the input length unlike for Sum.

Boolean synthesis and domain-safe automata. The quantitative synthesis problems that we have defined, ask for Boolean realizers that additionally satisfy a quantitative condition. We start by showing that a weighted specification $\mathcal{A}$ can be preprocessed in polynomial time such that dealing with the Boolean part becomes very simple. Basically, we remove all parts of $\mathcal{A}$ that cannot be used by a Boolean realizer. We call the result of this preprocessing a *domain-safe weighted specification*, to be defined formally below. In Section 4 we use domain-safe specifications.

Denote by $\mathrm{dom}(\mathcal{A}) \subseteq \Sigma_{\mathrm{i}}^*$ the domain of the weighted specification defined by $\mathcal{A}$. We can easily obtain an NFA (with ε-transitions) for $\mathrm{dom}(\mathcal{A})$ by removing the weights and turning all transitions that are labelled by an output letter into an ε-transition. We call the resulting NFA the domain automaton of $\mathcal{A}$, and denote it by $\mathcal{A}_{\mathrm{dom}}$. For a state q of $\mathcal{A}$, we denote by $L(\mathcal{A}_{\mathrm{dom}}, q)$ the language of $\mathcal{A}_{\mathrm{dom}}$ accepted by runs starting in q. An output transition (q, a, q') of $\mathcal{A}$ is called *domain-safe* if $L(\mathcal{A}_{\mathrm{dom}}, q) = L(\mathcal{A}_{\mathrm{dom}}, q')$, i.e., it does not restrict the language of input words that can be accepted by $\mathcal{A}_{\mathrm{dom}}$. Otherwise, such a transition is called *domain-unsafe*. We call a weighted specification $\mathcal{A}$ *domain-safe* if it is trim, i.e., all states are accessible and co-accessible, and all its output transitions are domain-safe.

A transducer that produces an input/output pair whose run in $\mathcal{A}$ uses a domain-unsafe transition of $\mathcal{A}$ cannot be a Boolean realizer of $\mathcal{A}$ because it cannot complete all inputs in the domain with an output in the relation $R(\mathcal{A})$. We now show that we can compute in polynomial time for a given weighted specification $\mathcal{A}$ a sub-automaton $\mathcal{A}'$ of $\mathcal{A}$ that is domain-safe and has the same Boolean realizers as $\mathcal{A}$. We would like to mention that there is a tight connection between domain-safe automata and the problem of "determinization by pruning" (DBP) as it is studied in [3]. The following result can also be derived from the proof of [3, Theorem 4.1]. Furthermore, the proof of Theorem 2 directly yields an alternative game-based proof of the "determinization by pruning" problem.

▶ **Theorem 2.** *There is a polynomial time procedure that takes as input a weighted specification $\mathcal{A}$, and either returns "no realizer" if $\mathcal{A}$ does not have Boolean realizers, or, otherwise, returns a sub-automaton $\mathcal{A}'$ of $\mathcal{A}$ that is domain-safe, has the same domain as $\mathcal{A}$, and has the same Boolean realizers as $\mathcal{A}$.*

A direct consequence of the above theorem is that the Boolean synthesis problem is decidable in polynomial time.

3 Critical prefix games

In this section we introduce the necessary definitions and notations regarding games. Moreover, we introduce critical prefix games and establish our results for these kind of games.

Games. A *weighted game with imperfect information* is an infinite-duration two-player game played on a game arena $G = (V, v_0, A, E, \mathcal{O}, w)$, where V is a finite set of vertices, $v_0 \in V$ is the initial vertex, A is a finite set of actions, $E \subseteq V \times A \times V$ is a labeled transition relation, $\mathcal{O} \subseteq 2^V$ is a set of *observations* that partition V, and $w\colon E \to \mathbb{Z}$ is a weight function. Without loss of generality, we assume that the arena has no dead ends, i.e., for all $v \in V$ there exists $a \in A$ and $v' \in V$ such that $(v, a, v') \in E$. The unique observation containing a vertex v is denoted $\mathsf{obs}(v)$. A game with *perfect information* is such that $\mathcal{O} = \{\{v\} \mid v \in V\}$. In that case we omit $\mathcal{O}$ from the tuple G.

Games are played in rounds in which Eve chooses an action $a \in A$, and Adam chooses an a-successor of the current vertex. The first round starts in the initial vertex v_0. A *play* π in G is an infinite sequence $v_0 a_0 v_1 a_1 \dots$ such that $(v_i, a_i, v_{i+1}) \in E$ for all $i \in \mathbb{N}$. The prefix of π up to v_n is denoted $\pi(n)$, its last element v_n is denoted by $\mathsf{last}(\pi(n))$. The set of all plays resp. prefixes of plays in G is denoted by $\mathsf{Plays}(G)$ resp. $\mathsf{Prefs}(G)$. The *observation sequence* of the play π is defined as $\mathsf{obs}(\pi) = \mathsf{obs}(v_0)a_0\mathsf{obs}(v_1)a_1 \dots$ and the finite observation sequence of the play prefix $\pi(n)$ is $\mathsf{obs}(\pi(n)) = \mathsf{obs}(v_0)a_0 \dots \mathsf{obs}(v_n)$. Naturally, obs extends to sets of (prefixes of) plays.

A game is defined by an arena G and an *objective* $\mathsf{Win} \subseteq \mathsf{Plays}(G)$ describing a set of good plays in G for Eve. A *strategy for Eve* in G is a mapping $\sigma\colon \mathsf{Prefs}(G) \to A$, it is called *observation-based* if for all play prefixes $\rho, \rho' \in \mathsf{Prefs}(G)$, if $\mathsf{obs}(\rho) = \mathsf{obs}(\rho')$, then $\sigma(\rho) = \sigma(\rho')$. Equivalently, an observation-based strategy is a mapping $\sigma\colon \mathsf{obs}(\mathsf{Prefs}(G)) \to A$. We do not formally introduce strategies for Adam, intuitively, given a play prefix and an action a, a strategy of Adam selects an a-successor of its last vertex. Given a strategy σ, let $\mathsf{Plays}_\sigma(G)$ denote the set of plays compatible with σ in G, and $\mathsf{Prefs}_\sigma(G)$ denote the set of play prefixes of $\mathsf{Plays}_\sigma(G)$. An Eve's strategy σ in G is *winning* if $\mathsf{Plays}_\sigma(G) \subseteq \mathsf{Win}$.

We now define quantitative objectives. The *energy level* of the play prefix $\pi(n)$ is $\mathsf{EL}(\pi(n)) = \sum_{i=1}^{n} w((v_{i-1}, a_{i-1}, v_i))$, the *sum value* is $\mathsf{Sum}(\pi(n)) = \sum_{i=1}^{n} w((v_{i-1}, a_{i-1}, v_i))$, the *average value* is $\mathsf{Avg}(\pi(n)) = \frac{1}{n}\mathsf{Sum}(\pi(n))$, and the *discounted-sum value* is $\mathsf{Dsum}(\pi(n)) = \sum_{i=1}^{n} \lambda^i w((v_{i-1}, a_{i-1}, v_i))$, and we let $\mathsf{Dsum}(\pi) = \sum_{i=1}^{\infty} \lambda^i w((v_{i-1}, a_{i-1}, v_i))$ (we do not explicitly mention the discount factor λ in this notation because it is always clear from the context).

The *energy objective* in G is parameterized by an initial credit $c_0 \in \mathbb{N}$ and is given by $\mathsf{PosEn}_G(c_0) = \{\pi \in \mathsf{Plays}(G) \mid \forall i \in \mathbb{N}\colon c_0 + \mathsf{EL}(\pi(i)) \geq 0\}$. It requires that the energy level of a play never drops below zero when starting with initial energy level c_0. The *fixed initial credit problem for imperfect information games* asks whether there exists an observation-based winning strategy for Eve for the objective $\mathsf{PosEn}_G(c_0)$. The *discounted-sum objective* in G is parameterized by a threshold $\nu \in \mathbb{Q}$, and $\triangleright \in \{>, \geq\}$. It is given by $\mathsf{DS}^{\triangleright}_G(\nu) = \{\pi \in \mathsf{Plays}(G) \mid \mathsf{Dsum}(\pi) \triangleright \nu\}$ and requires that the discounted-sum value of a play is greater than resp. at least ν. The *discounted-sum game problem* asks whether there exists a winning strategy for Eve for the objective $\mathsf{DS}^{\triangleright}_G(\nu)$.

A game with perfect information is a special case of an imperfect information game. Classically, instead of using the above model with full observation, a (weighted) perfect information game, simply called game, is defined over an arena $(V, V_\exists, v_0, E, w)$, where the set of vertices V is partitioned into $V_\exists$ and $V \setminus V_\exists$, the vertices belonging to Eve and Adam,

respectively, $v_0 \in V$ is the initial vertex, $E \subseteq V \times V$ is a transition relation, and $w\colon E \to \mathbb{Z}$ is a weight function. In a play on such a game arena, Eve chooses a successor if the current vertex belongs to her, otherwise Adam chooses. For games with perfect information the two models are equivalent and we shall use both.

Critical prefix games. A *critical prefix game* is a game, where the winning objective is parameterized by a set $C \subseteq V$ of *critical vertices*, and a set of play prefixes $W \subseteq \mathsf{Prefs}(G)$. Its objective is defined as $\mathsf{Crit}_{C,W}(G) = \{\pi \in \mathsf{Plays}(G) \mid \forall i\ \mathsf{last}(\pi(i)) \in C \to \pi(1)\dots\pi(i) \in W\}$. The idea of a critical prefix game is that the state of a play is only relevant whenever the play is in a critical vertex. For convenience, in the case of critical prefix games, we also refer to the set W as objective.

The *threshold problem for critical prefix games* asks whether there exists a winning strategy for Eve for the objective $\mathsf{Crit}_{C,W}(G)$, where W is of the form $\mathsf{Thres}_G^{V\triangleright}(\nu) = \{\varphi \in \mathsf{Prefs}(G) \mid V(\varphi) \triangleright \nu\}$ parameterized by a threshold $\nu \in \mathbb{Q}$, $\triangleright \in \{>, \geq\}$, and $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}\}$.

The *initial credit problem for critical prefix imperfect information energy games* asks whether there exists an observation-based winning strategy for Eve for the objective $\mathsf{Crit}_{C,W}(G)$, where W is of the form $\mathsf{PrefPosEn}_G(c_0) = \{\varphi \in \mathsf{Prefs}(G) \mid c_0 + \mathsf{EL}(\varphi) \geq 0\}$ parameterized by an initial credit $c_0 \in \mathbb{N}$.

▶ **Theorem 3.** *The threshold problem for critical prefix games for $V \in \{\mathsf{Sum}, \mathsf{Avg}\}$ and a strict or non-strict threshold is decidable in NP ∩ coNP. Moreover, positional strategies are sufficient for Eve to win such games.*

Proof sketch. For Sum and Avg and a strict or non-strict threshold, the critical prefix threshold games reduce to mean-payoff games which are solvable in NP ∩ coNP [34]. Positional strategies suffice for mean-payoff games, a winning strategy in the constructed mean-payoff game directly yields a positional winning strategy in the critical prefix threshold game. ◀

▶ **Theorem 4.** *The threshold problem for critical prefix games for* Dsum *and a strict resp. non-strict threshold is decidable in NP resp. NP ∩ coNP. Moreover, positional strategies are sufficient for Eve to win such games.*

To prove the above theorem, we first show a result on weighted graphs which is interesting in itself.

▶ **Lemma 5.** *Given a weighted graph G, a source vertex $v_0 \in V$, a target set $T \subseteq V$ and a threshold $\nu \in \mathbb{Q}$, checking whether there exists a path π from v_0 to some vertex $v \in T$ such that $\mathsf{Dsum}(\pi) \leq \nu$ can be done in PTIME.*

Lemma 5 can be used to show that the $\geq \nu$-non-emptiness problem for nondeterministic discounted-sum automata[3] can be checked in PTIME, a result which is, to the best of our knowledge, new. It was known to be in PSPACE for unambiguous discounted-sum automata [25, 9]. This problem asks for the existence of a word of value greater or equal than a given threshold ν. Since the value of a word is the maximal value amongst its accepting runs, it suffices to check for the existence of a run from the initial state to an accepting state of discounted-sum value $\geq \nu$. By inverting the weights, the latter is equivalent to checking

[3] In contrast to deterministic weighted automata, there might be serveral accepting runs on an input and the value of the word is defined as the maximal value of its accepting runs [25, 28].

whether there exists a run from the initial state to an accepting state of discounted-sum value $\leq -\nu$. By seeing the (inverted) discounted-sum automaton as a weighted graph, the latter property can be checked in PTIME by Lemma 5, thus proving the following theorem.

▶ **Theorem 6.** *The $\geq \nu$ non-emptiness problem is decidable in PTIME for nondeterministic discounted-sum automata.*

We now go back to the proof of Theorem 4.

Proof sketch of Theorem 4. For Dsum, and a non-strict threshold, the problem can be directly reduced to discounted-sum games which are solvable in NP $\cap$ coNP [4].

For Dsum, and a strict threshold, such a reduction fails. To solve the problem, we first show that positional strategies are sufficient for Eve to win in a critical prefix threshold discounted-sum game (for strict and non-strict thresholds). The NP-algorithm guesses a positional strategy σ for Eve, and then verifies in polynomial time whether σ is winning. Let G' be the game restricted to Eve's σ-edges, seen as a weighted graph. The strategy σ is not winning iff Adam can form a path in G' from the initial vertex to a critical vertex that has weight $\leq \nu$. This property can be checked in PTIME thanks to Lemma 5 (by taking as target set the set of critical vertices). ◀

The following is shown by reduction from the halting problem for 2-counter machines.

▶ **Theorem 7.** *The fixed initial credit problem for imperfect information critical prefix energy games is undecidable.*

The above result contrasts the fixed initial credit problem for imperfect information energy games which is decidable [22].

▶ **Theorem 8.** *The fixed initial credit problem for imperfect information critical prefix energy games is decidable if from each vertex Adam has a strategy to reach a critical vertex against observation based strategies. Moreover, finite-memory strategies are sufficient for Eve to win.*

Proof sketch. This problem is reduced to the fixed initial credit problem for imperfect information energy games which is decidable [22]. In classical energy games, Eve loses as soon as the energy goes below zero. The idea of the reduction is that if in the critical prefix energy game the initial credit is c_0, then in the classical energy game we start the game with an additional buffer, i.e., with $c_0 + B$, for some computable bound B. In the critical prefix energy game, if the energy level drops below $-B$ Adam can force to visit a critical vertex such that the energy level can rise by at most B, ensuring that a critical vertex is visited with energy level below zero. Thus, the additional buffer B suffices in the classical energy game. ◀

4 Synthesis problems

Here, we solve the quantitative synthesis problems defined in Section 2. Recall that weighted specifications are given by weighted automata that alternate between reading one input and one output symbol. In other words, we prove the decidability results of Table 1. We then show consequences of these results to quantitative synthesis problems over infinite words.

Threshold synthesis problems. Since weighted specifications S are given by weighted automata, the synthesis problem naturally reduces to a game played on the automaton. In order to solve threshold synthesis problems, in contrast to best-value and approximate synthesis problems, it is not necessary to compare the values of runs of the specification automaton that have the same input sequence. Hence, it is relatively straightforward to reduce threshold synthesis problems to critical prefix threshold games. An important point needs to be taken care of due to the fact the domain of S might be partial, and therefore lead Eve into the following bad situation ($\star$): Eve must choose her outputs in such a way that she does not go in a state of the automaton which is non-accepting, while the input word played by Adam so far is in the domain of S. Otherwise, the pair of input and output word formed would not even be in S, something which is required by the definition of synthesis problems. So, Eve has to monitor the domain, which is easy if the domain is total, but more involved if it is partial. Thanks to Theorem 2, this can be done in polynomial time. More precisely, we first run the algorithm of Theorem 2 which either returns that there is no Boolean realizer, or returns a domain-safe deterministic weighted automaton $\mathcal{A}'$ which has the same Boolean realizers as S. By the very definition of domain-safe automata, the bad situation ($\star$) described above cannot happen. Hence, Eve can freely play on $\mathcal{A}'$ without taking care of the domain constraint. Only the quantitative constraint matters, and it has to be enforced whenever Eve is in an accepting state of $\mathcal{A}'$ (this corresponds to the situation where Adam has chosen an input word in the domain of S). Hence, only accepting states of $\mathcal{A}'$ matter for the quantitative constraint and these are declared as critical. To conclude, by projecting away the symbols of $\mathcal{A}'$ and by declaring its accepting states to be critical, we obtain a critical prefix game. For the threshold synthesis problem, decidability follows directly from the decidability of the threshold problem for critical prefix games (Theorems 3 and 4). For Sum- and Avg-specifications, this can be done in NP $\cap$ coNP. We leave open whether it is solvable in Ptime and show that this would also solve the long standing open problem of whether mean-payoff games are solvable in Ptime.

▶ **Theorem 9.** *The threshold synthesis problem for a V-specification with $V \in \{\mathsf{Sum}, \mathsf{Avg}\}$ and a strict or non-strict threshold is decidable in NP $\cap$ coNP and PTIME-equivalent to mean-payoff games. The threshold synthesis problem for a* Dsum*-specification and a strict resp. non-strict threshold is decidable in NP resp. in NP $\cap$ coNP.*

Synthesis and regret determinization. Before we prove our results about best-value and approximate synthesis, we highlight the tight connection between the approximate synthesis problem and the so-called regret determinization problem for nondeterministic weighted automata[4]. This problem has for instance been studied in [26] for Sum-automata and in [29] for Dsum-automata. We formalize this connection here. Given $r \in \mathbb{Q}$ and $\triangleleft \in \{<, \leq\}$, a nondeterministic WFA $\mathcal{A} = (Q, \Sigma, q_i, \Delta, F, \gamma)$ is called *$r_\triangleleft$-regret determinizable* if there exists a finite set of memory states M and a deterministic WFA $\mathcal{A}_r = (Q \times M, \Sigma, q_i^r, \Delta_r, F_r, \gamma_r)$, where $q_i^r = (q_i, m)$ for some $m \in M$, $F_r \subseteq F \times M$, $\big((q,m),a,(q,m')\big) \in \Delta_r$ implies that $(q, a, q') \in \Delta$, and $\gamma_r\big(\big((q,m),a,(q,m')\big)\big) = \gamma((q,a,q'))$ for all $m, m' \in M$, such that $L(\mathcal{A}) = L(\mathcal{A}_r)$ and $\mathcal{A}(w) - \mathcal{A}_r(w) \triangleleft r$ for all $w \in \mathrm{dom}(L(\mathcal{A}))$. The *regret determinization problem* asks, given a nondeterministic weighted automaton $\mathcal{A}$, a threshold $r \in \mathbb{Q}$, and $\triangleleft \in \{<, \leq\}$, whether $\mathcal{A}$ is $r_\triangleleft$-regret determinizable.

[4] In contrast to deterministic weighted automata, there might be serveral accepting runs on an input and the value of the word is defined as the maximal value of its accepting runs [25, 28].

▶ **Lemma 10.** *The approx. synthesis problem for weighted specifications reduces in linear time to the regret determinization problem for nondet. weighted automata (with the same threshold). The converse is true (in linear time and with the same threshold) for* Sum*-automata.*

Lemma 10 is independent from any payoff function. Regarding the converse direction, when going from the regret determinization problem to the approximate synthesis problem, a transition (for an input symbol) must be translated into two transitions (adding an output symbol). This step can cause difficulties depending on the used payoff function, e.g., Dsum.

Best-value synthesis problems. Best-value synthesis is equivalent to zero-regret synthesis, which is, by Lemma 10, equivalent to zero-regret determinization of weighted automata. In [9], the authors showed that if a Sum-automaton is zero-regret determinizable, then no memory states are needed, i.e., a sub-automaton suffices. We give general sufficient conditions on weighted finite automata (which hold for Sum-, Avg- and Dsum-automata) under which the latter result can be generalized.

Let $V\colon \mathbb{Z}^* \to \mathbb{Q}$ be a payoff function. A V-automaton defining a V-specification, where V is applied to runs as usual, is called $\leq$*-stable* if for all runs ρ, ρ', ρ'' such that the end state of ρ is the beginning state of ρ' and ρ'', $w' = u \otimes v'$, and $w'' = u \otimes v''$ for some $u \in \Sigma_{\mathtt{i}}^*$ and $v', v'' \in \Sigma_{\mathtt{o}}^*$, where w' and w'' are the words associated to ρ' and ρ'', respectively, holds that if $V(\rho') \leq V(\rho'')$ then $V(\rho\rho') \leq V(\rho\rho'')$.

▶ **Lemma 11.** *Given a weighted specification S by a $\leq$-stable weighted automaton $\mathcal{A}$, if there exists a transducer that implements a best-value S-realization, then there exists a transducer that implements a best-value S-realization that is defined as a sub-automaton of $\mathcal{A}$.*

While the above lemma can be used to obtain our decidability results for best-value synthesis, we use other techniques to obtain the complexity results stated below.

▶ **Theorem 12.** *The best-value synthesis problem is decidable in* Ptime *for* Sum*-specifications and* Avg*-specifications, and in NP∩coNP for* Dsum*-specifications.*

Proof sketch. For Sum, the problem reduces to the zero-regret determinization problem for Sum-automata, see Lemma 10, aka the determinization by pruning problem for Sum-automata, known to be decidable in Ptime in [3]. For Avg, it easily reduces to Sum by interpreting the Avg-specification as a Sum-specification. For Dsum, we show that the problem reduces in Ptime to a critical prefix threshold game, for non-strict threshold, which is solvable in NP∩coNP by Theorem 4. ◀

Alternatively, decidability for Dsum can be obtained by reduction to the zero-regret determinization problem for Dsum-automata over infinite words which was shown to be decidable in NP in [29, Theorem 6]. However, our techniques allow us to get NP∩coNP.

Approximate synthesis problems. We now turn to the approximate synthesis problems and show its decidability for Sum and Avg. We leave the decidability status open for Dsum, but nevertheless show decidability for a large class, namely when the discount factor is of the form $\frac{1}{n}$ for $n \in \mathbb{N}$. Nondeterministic Dsum-automata in this class have been considered in [8] and shown to be determinizable.

▶ **Theorem 13.** *The approximate synthesis problem is*

- *ExpTime-complete for* Sum*-specifications and strict or non-strict thresholds;*
- *decidable and ExpTime-hard for* Avg*-specifications and strict or non-strict thresholds;*
- *in NExpTime (resp. ExpTime) for* Dsum*-specifications with a discount factor λ of the form $\frac{1}{n}$ with $n \in \mathbb{N}$ and strict (resp. non-strict) thresholds.*

Proof sketch. For Sum, we reduce the problem to r-regret determinization of Sum-automata, known to be ExpTime-complete, using the back-and-forth connection given by Lemma 10.

For an Avg-specifications S, it is worth noting that even though r-approximate synthesis reduces to r-approximate synthesis for Sum when $r = 0$, interpreting S as a Sum-specification, this reduction is wrong for $r > 0$ in general. It is because in an Avg-specification, Eve can deviate more and more from the best sum, while the average of this difference can stay low. We instead rely on a reduction to critical prefix energy games of imperfect information and fixed initial credit (which falls into the decidable subclass of Theorem 8). Intuitively, in this game, Adam constructs a run ρ on a pair of words (u, v) and Eve constructs a run ρ' on some (u, v'). She only sees u and not ρ. The energy level of such a play is set to $\mathsf{Sum}(\rho') + |uv| \cdot r - \mathsf{Sum}(\rho)$ and must be positive whenever Adam reaches an accepting state. ExpTime-hardness is perhaps the most technical result of the paper, and is a non-trivial adaptation of reduction from countdown games used to show ExpTime-hardness of the regret determinization of Sum-automata [26].

Finally, for Dsum, we use that by projecting away the output in the Dsum-automaton defining the specification, we obtain a nondeterministic weighted automaton which is determinizable by [8]. This allows us to reduce the problem to the threshold synthesis problem for Dsum, which is decidable by Theorem 9. To obtain the complexity results, we first analyze the determinization procedure. It yields an automaton whose states are exponential in the number of states and polynomial in the weights of the nondeterministic one. Its weights are polynomial in the weights of the nondeterministic one. For a strict threshold, the claimed complexity bound follows directly from Theorem 9. For a non-strict threshold, we use that critical prefix threshold games are reduced in polynomial time to discounted-sum games. Using value iteration [34] to solve discounted-sum games yields the claimed complexity bound, because it runs in polynomial time in the size of the arena, logarithmic in the absolute maximal weight of the arena, and exponential in the representation of the discount factor, i.e., polynomial in the discount factor. ◀

Infinite words and Church synthesis. An ω-specification is a subset $S \subseteq (\Sigma_{\mathtt{i}}.\Sigma_{\mathtt{o}})^\omega$. The (Church) synthesis problem asks to decide whether there exists a strategy to pick a correct output sequence given longer and longer prefixes of an infinite input sequence. Formally, an ω-specification S is said to be *realizable* if there exists a function $\lambda\colon \Sigma_{\mathtt{i}}^* \to \Sigma_{\mathtt{o}}$ such that for all $i_1 i_2 \cdots \in \Sigma_{\mathtt{i}}^\omega$, it holds that $i_1\lambda(i_1)i_2\lambda(i_1 i_2)i_3\lambda(i_1 i_2 i_3)\cdots \in S$.

Strategies of interest are those which can be represented by a finite-state machine, and in particular a Mealy machine, that is, roughly, a transducer running on ω-words and without acceptance condition. Formally, it is a tuple $M = (P, p_0, \delta)$ such that P is a finite set of states with initial state p_0, and $\delta\colon P \times \Sigma_{\mathtt{i}} \to \Sigma_{\mathtt{o}} \times P$ is a (total) transition function. The function δ can be extended to $\delta^*\colon P \times \Sigma_{\mathtt{i}}^+ \to \Sigma_{\mathtt{o}} \times P$ as usual. Then, M defines the strategy λ_M such that for all $u \in \Sigma_{\mathtt{i}}^*$, $\lambda_M(u) = \pi_1(\delta^*(p_0, u))$, where π_1 is the first projection. It is well-known that when S is ω-regular (given e.g. as a parity automaton), it is decidable whether S is realizable [14]. Moreover, realizability implies realizability by a Mealy machine.

Weighted safety specifications. In this paper, we go beyond ω-regular specifications, by considering safety ω-specifications induced by weighted specifications of finite words defined by deterministic weighted automata. Let $W\colon (\Sigma_{\mathtt{i}}\Sigma_{\mathtt{o}})^* \to \mathbb{Q} \cup \{-\infty\}$ be a weighted specification. For a threshold $t \in \mathbb{Q}$ and $\triangleright \in \{>, \geq\}$, we define the ω-specification $\mathsf{Thres}^{\triangleright t}(W) = \{i_1 o_1 \cdots \in (\Sigma_{\mathtt{i}}.\Sigma_{\mathtt{o}})^\omega \mid \forall k \geq 0, i_1 \dots i_k \in \mathrm{dom}(W) \to W(i_1 o_1 \dots i_k o_k) \triangleright t\}$. In words, an ω-word w is in $\mathsf{Thres}^{\triangleright t}(W)$ iff for all finite prefixes $u = i_1 o_1 \dots i_k o_k$ of w, either $i_1 \dots i_k \notin \mathrm{dom}(W)$ or

$W(u) \triangleright t$. So, the quantitative condition is checked only for prefixes whose input belongs to $\mathrm{dom}(W)$. The ω-specification $\mathsf{Thres}^{\triangleright t}(W)$ is a safety specification [5]. More generally, any set $S \subseteq (\Sigma_{\mathtt{i}}.\Sigma_{\mathtt{o}})^*$ induces a safety ω-specification $\mathsf{Safe}(S) = \{i_1o_1 \cdots \in (\Sigma_{\mathtt{i}}.\Sigma_{\mathtt{o}})^\omega \mid \forall k \geq 0, i_1 \dots i_k \in \mathrm{dom}(S) \to i_1o_1 \dots i_ko_k \in S\}$.

For example, we have the equality $\mathsf{Thres}^{\triangleright t}(W) = \mathsf{Safe}(\{u \in (\Sigma_{\mathtt{i}}.\Sigma_{\mathtt{o}})^* \mid W(u) \triangleright t\})$. Likewise, we define best-value and approximate safety ω-specifications. Formally, given a finite word $i_1 \dots i_k \in \Sigma_{\mathtt{i}}^*$ and $\triangleleft \in \{<, \leq\}$, we let $\mathsf{BestVal}(W) = \mathsf{Approx}^{\leq}(W, 0)$ where for all $r \in \mathbb{Q}_{\geq 0}$ we have $\mathsf{Approx}^{\triangleleft}(W, r) = \mathsf{Safe}(\{u = i_1o_1 \dots i_ko_k \mid \mathsf{bestVal}_W(i_1 \dots i_k) - W(u) \triangleleft r\}$. Note that the three notions of safety ω-specifications we have defined are not necessarily ω-regular, even if W is given by a deterministic weighted automaton. Nevertheless, an immediate consequence of the results we have obtained previously on finite words is that

▶ **Theorem 14.** *The synthesis problem for an ω-specification $O \subseteq (\Sigma_{\mathtt{i}}.\Sigma_{\mathtt{o}})^\omega$ is decidable when O is given by a deterministic V-automaton defining a weighted V-specification of finite words W s.t. $O \in \{\mathsf{Thres}^{>t}(W), \mathsf{Thres}^{\geq t}(W), \mathsf{BestVal}(W), \mathsf{Approx}^{<}(W, r), \mathsf{Approx}^{\leq}(W, r)\}$ and $V = \mathsf{Sum}$, $V = \mathsf{Avg}$ or $V = \mathsf{Dsum}$ with discount factor $1/n$ for $n \in \mathbb{N}$. Moreover, if O is realizable, it is realizable by a Mealy machine.*

5 Future work

In this paper, weighted specifications are defined by deterministic weighted automata. Nondeterministic, even unambiguous, weighted automata, are strictly more expressive than their deterministic variant in general, and in particular for Sum, Avg and Dsum. An interesting direction is to revisit our quantitative synthesis problems for specifications defined by nondeterministic weighted automata. Using similar ideas as the undecidability of critical prefix energy games of imperfect information, it can be shown that threshold synthesis becomes undecidable for unambiguous sum- and avg-specifications. The problem is open for best-value and approximate synthesis, and we plan to investigate it.

Two other directions seem interesting as future work, both in the setting of infinite words. First, natural measures in this setting are discounted-sum and mean-payoff. While the threshold synthesis problems directly reduce to known results and best-value/approximate synthesis for dsum has been studied in [29], nothing is known to the best of our knowledge about best-value/approximate synthesis for mean-payoff. We expect the techniques to be different because such a measure is prefix-independent, unlike our measures in the setting of finite words. As a second direction, we have seen how our results apply to synthesis on infinite words through weighted safety conditions. An interesting direction is to consider such weighted requirements in conjunction with ω-regular conditions such as parity, in the line of [17] that combines energy and parity objectives in games.

References

1 Shaull Almagor and Orna Kupferman. Good-enough synthesis. In *International Conference on Computer Aided Verification*, pages 541–563. Springer, 2020.

2 Shaull Almagor, Orna Kupferman, Jan Oliver Ringert, and Yaron Velner. Quantitative assume guarantee synthesis. In *International Conference on Computer Aided Verification*, pages 353–374. Springer, 2017.

[5] A language of ω-words S is a safety language if any ω-word w whose finite prefixes u are such that $uv_u \in S$ for some ω-word v_u, belongs to S [15].

3 Benjamin Aminof, Orna Kupferman, and Robby Lampert. Reasoning about online algorithms with weighted automata. *ACM Trans. Algorithms*, 6(2):28:1–28:36, 2010.

4 Daniel Andersson. An improved algorithm for discounted payoff games. In *ESSLLI Student Session*, pages 91–98, 2006.

5 Roderick Bloem, Krishnendu Chatterjee, Thomas A Henzinger, and Barbara Jobstmann. Better quality in synthesis through quantitative objectives. In *International Conference on Computer Aided Verification*, pages 140–156. Springer, 2009.

6 Roderick Bloem, Rüdiger Ehlers, and Robert Könighofer. Cooperative reactive synthesis. In Bernd Finkbeiner, Geguang Pu, and Lijun Zhang, editors, *Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Shanghai, China, October 12-15, 2015, Proceedings*, volume 9364 of *Lecture Notes in Computer Science*, pages 394–410. Springer, 2015.

7 Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3):911–938, 2012. doi:10.1016/j.jcss.2011.08.007.

8 Udi Boker and Thomas A. Henzinger. Exact and approximate determinization of discounted-sum automata. *Logical Methods in Computer Science*, 10(1), 2014.

9 Udi Boker, Thomas A. Henzinger, and Jan Otop. The target discounted-sum problem. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 750–761. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.74.

10 Patricia Bouyer, Uli Fahrenberg, Kim G Larsen, and Nicolas Markey. Timed automata with observers under energy constraints. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pages 61–70, 2010.

11 Tomáš Brázdil, Petr Jančar, and Antonín Kučera. Reachability games on extended vector addition systems with states. In *International Colloquium on Automata, Languages, and Programming*, pages 478–489. Springer, 2010.

12 Romain Brenguier, Lorenzo Clemente, Paul Hunter, Guillermo A Pérez, Mickael Randour, Jean-François Raskin, Ocan Sankur, and Mathieu Sassolas. Non-zero sum games for reactive synthesis. In *Language and Automata Theory and Applications*, pages 3–23. Springer, 2016.

13 Romain Brenguier, Jean-François Raskin, and Ocan Sankur. Assume-admissible synthesis. *Acta Informatica*, 54(1):41–83, 2017. doi:10.1007/s00236-016-0273-2.

14 J Richard Büchi and Lawrence H Landweber. Solving sequential conditions finite-state strategies. *Trans. Ameri. Math. Soc.*, 138:295–311, 1969.

15 Edward Chang, Zohar Manna, and Amir Pnueli. The safety-progress classification. In *Logic and Algebra of Specification*, pages 143–202. Springer, 1993.

16 Krishnendu Chatterjee and Laurent Doyen. Energy parity games. *Theoretical Computer Science*, 458, 2012.

17 Krishnendu Chatterjee and Laurent Doyen. Energy parity games. *Theor. Comput. Sci.*, 458:49–60, 2012. doi:10.1016/j.tcs.2012.07.038.

18 Krishnendu Chatterjee and Thomas A. Henzinger. Assume-guarantee synthesis. In Orna Grumberg and Michael Huth, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 13th International Conference, TACAS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007 Braga, Portugal, March 24 - April 1, 2007, Proceedings*, volume 4424 of *Lecture Notes in Computer Science*, pages 261–275. Springer, 2007.

19 A Church. Applications of recursive arithmetic to the problem of circuit synthesis–summaries of talks. *Institute for Symbolic Logic, Cornell University*, 1957.

20 Edmund M Clarke, Thomas A Henzinger, Helmut Veith, and Roderick Bloem. *Handbook of model checking*, volume 10. Springer, 2018.

21 Rodica Condurache, Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. The complexity of rational synthesis. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages,*

and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy, volume 55 of *LIPIcs*, pages 121:1–121:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

22 Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Toruńczyk. Energy and mean-payoff games with imperfect information. In *International Workshop on Computer Science Logic*, pages 260–274. Springer, 2010.

23 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.

24 Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8(2):109–113, 1979.

25 Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. Quantitative languages defined by functional automata. In *CONCUR*, volume 7454 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2012.

26 Emmanuel Filiot, Ismaël Jecker, Nathan Lhote, Guillermo A. Pérez, and Jean-François Raskin. On delay and regret determinization of max-plus automata. In *LICS*, pages 1–12. IEEE Computer Society, 2017.

27 Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. Antichains and compositional algorithms for LTL synthesis. *Formal Methods in System Design*, 39(3):261–296, 2011.

28 Axel Haddad and Benjamin Monmege. Why value iteration runs in pseudo-polynomial time for discounted-payoff games. *Technical note, Université libre de Bruxelles*, 2015.

29 Paul Hunter, Guillermo A. Pérez, and Jean-François Raskin. Minimizing regret in discounted-sum games. In *CSL*, volume 62 of *LIPIcs*, pages 30:1–30:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

30 Orna Kupferman, Giuseppe Perelli, and Moshe Y. Vardi. Synthesis with rational environments. *Ann. Math. Artif. Intell.*, 78(1):3–20, 2016.

31 Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. Safraless compositional synthesis. In *Computer Aided Verification, 18th International Conference, CAV 2006*, volume 4144 of *Lecture Notes in Computer Science*, pages 31–44. Springer, 2006.

32 Jianwen Li, Kristin Y. Rozier, Geguang Pu, Yueling Zhang, and Moshe Y. Vardi. Sat-based explicit ltlf satisfiability checking. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 2946–2953. AAAI Press, 2019.

33 Shufang Zhu, Lucas M. Tabajara, Jianwen Li, Geguang Pu, and Moshe Y. Vardi. Symbolic ltlf synthesis. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1362–1369. ijcai.org, 2017.

34 Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1-2):343–359, 1996.