Weak Cost Monadic Logic Over Infinite Trees[⋆]

Michael Vanden Boom

Department of Computer Science, University of Oxford, UK
michael.vandenboom@cs.ox.ac.uk

Abstract. Cost monadic logic has been introduced recently as a quantitative extension to monadic second-order logic. A sentence in the logic defines a function from a set of structures to $\mathbb{N} \cup \{\infty\}$, modulo an equivalence relation which ignores exact values but preserves boundedness properties. The rich theory associated with these functions has already been studied over finite words and trees.

We extend the theory to infinite trees for the weak form of the logic (where second-order quantification is interpreted over finite sets). In particular, we show weak cost monadic logic is equivalent to weak cost automata, and finite-memory strategies suffice in the infinite two-player games derived from such automata. We use these results to provide a decision procedure for the logic and to show there is a function definable in cost monadic logic which is not definable in weak cost monadic logic.

1 Introduction

A fundamental result in the theory of regular languages is the equivalence between monadic second-order logic (MSO) and finite-state automata, which Büchi exploited in order to provide a decision procedure for the logic. Recently, Colcombet [3] has proposed a quantitative extension to MSO called cost monadic second-order logic (cost MSO). In this setting, a cost MSO sentence defines a function from some domain (like words or trees over a finite alphabet) to $\mathbb{N} \cup \{\infty\}$, modulo an equivalence relation $\approx$ which ignores exact values but preserves the existence of bounds over any subset of the domain. Mirroring the classical result, there is an equivalent automaton model called cost automata which can be used to help decide whether the functions definable by cost MSO sentences (over finite words [3] or finite trees [7]) are equivalent up to $\approx$.

This "theory of regular cost functions" is a strict extension to the theory of regular languages, which retains the equivalences, closure properties, and decidability of the classical theory. It captures the theory of regular languages since we can identify each language with its characteristic function mapping structures in the language to 0 and everything else to ∞; it is a strict extension since cost MSO can count some behaviour within the input structure (e.g. the number of positions labelled with some symbol). The theory has been studied over finite words [3, 4] and finite trees [5, 7]. This paper is an initial step in extending the theory to infinite trees when restricting to weak cost monadic logic (written cost WMSO) in which second-order quantifiers are interpreted over finite sets.

⋆ Supported by the French ANR 2007 JCJC 0051 JADE.

1.1 Motivation

The motivation for studying the logic and automata considered in this paper comes from problems that can be reduced to questions of boundedness. The most famous language-theoretic problem like this is the "star-height problem": given a regular language L and natural number n, is there some regular expression (using concatenation, union, and Kleene star) for L which uses at most n nestings of Kleene-star operations? Hashiguchi [8] first showed that this problem is decidable over finite words and Kirsten [9] gave an alternative proof. Colcombet and Löding have shown that this problem is also decidable over finite trees [5].

In each case, finite-state automata enriched with counting features were used (distance, nested distance-desert, and cost automata). The star-height problem was then reduced to a question of "limitedness": given some automaton with counting features, is the function it computes bounded over all accepted structures? Because limitedness is a special case of deciding $\approx$, this theory of regular cost functions is a useful framework for reasoning about this sort of problem.

Kirsten [9] and Colcombet [3] cite problems in areas as diverse as speech recognition, databases, and model theory which have been solved using a similar approach. One open problem is the "parity-index problem". It asks: given a regular language L of infinite trees and $i < j$, is there a parity automaton using only priorities $\{i, i+1, \ldots, j\}$? Colcombet and Löding [6] have shown that this problem is reducible to limitedness for a "cost-parity automaton" over infinite trees, but the decidability of limitedness for these automata remains open. Thus, understanding the theory of regular cost functions over infinite trees is desirable.

1.2 Contributions

We show that the results over finite trees from [7] can be lifted to infinite trees when restricting to cost WMSO. The main contribution is proving cost WMSO is effectively equivalent to alternating "weak cost automata" and that these automata can be simulated by a type of non-deterministic automata. This is used to prove the relation $\approx$ is decidable for functions definable in cost WMSO. Another consequence (similar to the classical theory) is a separation result showing there is a function definable in MSO which is not definable in cost WMSO.

The main difficulty compared to the finite-tree case in [7] is the simulation result for weak cost automata. It relies on the fact that certain "weak cost games" (games derived from weak cost automata) admit finite-memory strategies. Proving this result over infinite trees is more difficult because of the interplay between the traditional acceptance condition and the cost features of the game.

1.3 Organisation

In Sect. 2 we provide background on trees, cost WMSO, and cost functions. We then present in Sect. 3 the general framework for reasoning about cost automata acting on infinite trees using infinite two-player games. Using this framework, we prove the results mentioned above for cost WMSO and weak cost automata in Sect. 4. We conclude in Sect. 5.

2 Cost Monadic Logic

2.1 Trees

Let $\mathbb{N}$ be the set of non-negative integers and $\mathbb{N}_\infty := \mathbb{N} \cup \{\infty\}$, ordered such that $0 < 1 < \cdots < \infty$. For $i \leq j$, we write $[i, j]$ to denote $\{i, i+1, \ldots, j\}$. We fix a finite alphabet $\mathbb{A}$ which is *ranked*: each symbol $a \in \mathbb{A}$ has some *arity* $j \in \mathbb{N}$ denoted by $a \in \mathbb{A}_j$. Let r be the maximum arity of the labels in $\mathbb{A}$. The set $\mathcal{T}_{\mathbb{A}}$ of infinite $\mathbb{A}$-*labelled ranked trees* is the set of partial functions $t : [1, r]^* \to \mathbb{A}$ such that the domain of t (denoted $pos(t)$) is prefix-closed, $t(\epsilon) \in \mathbb{A}$ is the label at the root, and if $t(x) \in \mathbb{A}_j$ then $t(xi)$ is defined if and only if $i \leq j$.

2.2 Cost Monadic Logic

Cost monadic second-order logic, or cost MSO, is a quantitative extension to MSO (see e.g., [14] for an introduction to monadic second-order logic). As usual, the logic can be defined over any relational structure, but we describe here the logic over $\mathbb{A}$-labelled ranked trees. In addition to first-order variables which range over nodes and second-order monadic variables which range over sets of nodes, cost MSO uses a single additional variable N called the *bound variable* which ranges over $\mathbb{N}$. The atomic formulas in cost MSO are the usual atomic formulas from MSO (namely, the membership relation $x \in X$ and relations $a(x, x_1, \ldots, x_k)$ which assert that $a \in \mathbb{A}_k$ is the label at position x with children $x_1, \ldots, x_k$ from left to right), as well as new predicates $|X| \leq N$ where X is any second-order variable and N is the bound variable. Arbitrary cost MSO formulas can be built inductively in the usual way by applying boolean connectives or by quantifying (existentially or universally) over first- or second-order variables. We additionally require that predicates of the form $|X| \leq N$ appear positively in the formula (i.e. within the scope of an even number of negations).

If we fix a value n for N, then the semantic of $|X| \leq N$ is what one would expect: the predicate is satisfied if and only if the valuation of X has cardinality at most n. If this value for N is not specified, then a sentence φ in cost monadic logic defines a function $[\![\varphi]\!]$, from $\mathcal{T}_{\mathbb{A}}$ to $\mathbb{N}_\infty$ (the natural numbers extended with a special infinity symbol ∞). The function is

$$[\![\varphi]\!](t) := \inf\{n : t, n \models \varphi\}$$

where $t, n \models \varphi$ if t satisfies φ when all occurrences of N take value n. By convention, $\inf \emptyset = \infty$, so in case φ is a pure MSO sentence (with no instances of the predicates $|X| \leq N$), $[\![\varphi]\!](t)$ is 0 if t satisfies the sentence φ and ∞ otherwise.

In Sect. 4, we will focus our attention on *cost weak monadic second-order logic* (written weak cost monadic logic or cost WMSO) which restricts the second-order quantification to finite sets, as usual. WMSO (and consequently cost WMSO) is still a very expressive logic (e.g. CTL embeds into it) but as we will see in Sect. 4, it has some nice properties that make working with the corresponding automata and games easier than in the full logic. We pause to give an example of a typical function definable in cost WMSO.

Example 1. Let $\mathbb{A} = \mathbb{A}_2 = \{a, b, c\}$. We seek to define the function which, for trees with infinitely many a's, outputs the maximum number of b's along a single branch (and otherwise assigns value ∞). A suitable cost WMSO sentence φ is

$$\forall X.\exists x.\big(\neg(x \in X) \wedge a(x)\big) \wedge \forall Z.\big((\forall z.(z \in Z \to b(z)) \wedge \mathit{chain}(Z)) \to |Z| \leq N\big).$$

where $\mathit{chain}(Z)$ is a WMSO formula asserting Z is totally ordered (and hence the nodes are on the same branch). We write $a(x)$ for $\exists x_1.\exists x_2.a(x, x_1, x_2)$.

The first conjunct is a typical WMSO formula: "infinitely many a's" is expressed as "for all finite sets of nodes, there is an a-labelled node outside". The least n that can be substituted for N to satisfy the second conjunct is exactly the bound on the number of b's along a single branch (∞ if there is no bound).

2.3 Cost Functions

Given cost WMSO sentences φ and ψ, we would like to be able to decide for any $t \in \mathcal{T}_{\mathbb{A}}$ whether $[\![\varphi]\!](t) \leq [\![\psi]\!](t)$ or $[\![\varphi]\!](t) = [\![\psi]\!](t)$. This is undecidable even over words by [10], so we must relax the relation being used. Following [4], we define relations $\preccurlyeq$ and $\approx$. Given $f, g : \mathcal{T}_{\mathbb{A}} \to \mathbb{N}_\infty$, we say g *dominates* f (written $f \preccurlyeq g$) if and only if for all $U \subseteq \mathcal{T}_{\mathbb{A}}$, $g(U)$ bounded implies $f(U)$ bounded. We write $f \approx g$ if and only if $f \preccurlyeq g$ and $g \preccurlyeq f$. Thus, $\preccurlyeq$ and $\approx$ are weakenings of $\leq$ and $=$ which ignore exact values of f and g, but do preserve boundedness properties.

If we want to be more precise about the relationship between f and g, we can annotate $\preccurlyeq$ and $\approx$ with a *correction function* $\alpha : \mathbb{N} \to \mathbb{N}$, a non-decreasing function which satisfies $\alpha(n) \geq n$ for all n. We write $f \preccurlyeq_\alpha g$ if $f(t) \leq \alpha(g(t))$ for all $t \in \mathcal{T}_{\mathbb{A}}$ (with the convention that $\alpha(\infty) = \infty$). Thus, α describes how much we may need to "stretch" g such that it dominates f. As an example, the function $|\cdot|_a$ which counts the number of a's in an infinite $\{a, b\}$-labelled tree is not $\approx$-equivalent to $|\cdot|_b$; however, $|\cdot|_a \approx_\alpha 2|\cdot|_a$ for $\alpha(n) = 2n$. To compare single values $m, n \in \mathbb{N}$, we also write $m \preccurlyeq_\alpha n$ if $m \leq \alpha(n)$.

With these relations, we can formally define a *cost function* F to be an equivalence class of $\approx$, but we will blur the distinction between a particular function $f : \mathcal{T}_{\mathbb{A}} \to \mathbb{N}_\infty$ and its equivalence class F. The natural decision procedure in this setting is the following: given two cost functions f and g, is $f \preccurlyeq g$? We remark that the classical language inclusion problem and the limitedness problem mentioned in the introduction can be seen as a special cases of this procedure.

3 Cost Games

3.1 Objectives

We will use a game-theoretic approach to tree automata (see e.g., [14]).

An *objective* O is a tuple $\langle \mathbb{C}, f, \mathit{goal}\rangle$ where $\mathbb{C}$ is a finite alphabet of actions, $f : \mathbb{C}^\omega \to \mathbb{N}_\infty$ maps sequences of actions to a value, and $\mathit{goal} \in \{\min, \max\}$ describes how a player seeks to optimize f. Switching the goal (from min to max or max to min) yields the dual objective $\overline{O}$ representing the aim of the opponent.

These objectives take the place of classical winning conditions. For example, with a *parity condition*, Eve wins if the maximum priority occurring infinitely-often is even. This is described by the objective $\langle \Omega, P, \min\rangle$ where $\Omega \subseteq \mathbb{N}$ is a set of priorities and $P(u)$ is 0 if the maximum infinitely-occurring priority in u is even (∞ otherwise). Winning for Eve corresponds to minimizing P. The *Büchi condition* is a special case where $\Omega = [1, 2]$.

We want to enrich the classical objectives with counter actions such that values come from $\mathbb{N}_\infty$ (instead of only $\{0, \infty\}$). A counter γ is initially assigned value 0 and can be *incremented* $\mathtt{i}$, *reset* $\mathtt{r}$ to 0, *checked* $\mathtt{c}$, or left unchanged ε. We care about the value of the counter at the moment(s) when it is checked. Given a word u_γ over the alphabet $\{\varepsilon, \mathtt{i}, \mathtt{r}, \mathtt{c}\}$, we define a set $C(u_\gamma) \subseteq \mathbb{N}$ which collects all of the checked values of γ. For instance, $C(\mathtt{iri}\varepsilon\mathtt{iicriic}) = \{2, 3\}$ since the first time the counter is checked it has value 3 and the second time it has value 2. In the case of a finite set of counters Γ and a word u over the alphabet $\{\varepsilon, \mathtt{i}, \mathtt{r}, \mathtt{c}\}^\Gamma$, $C(u) := \bigcup_{\gamma\in\Gamma} C(u_\gamma)$ (u_γ is the γ-projection of u).

We will use three objectives which combine a classical parity condition with particular atomic counter actions and valuations. The *B-parity objective*[1] (over counters Γ and priorities Ω) is $\mathit{Cost}_B^{\Gamma,\Omega} := \langle \{\varepsilon, \mathtt{ic}, \mathtt{r}\}^\Gamma \times \Omega, \mathit{cost}_B^{\Gamma,\Omega}, \min\rangle$ where

$$\mathit{cost}_B^{\Gamma,\Omega}(u) := \sup(C(u) \cup \{P(u)\})$$

and $P(u)$ is the function described above which interprets the parity condition (on the projection of u to its last component). The atomic actions in this case are ε, $\mathtt{ic}$, and $\mathtt{r}$. For example, if $u = ((\mathtt{ic}, 2)(\mathtt{ic}, 2)(\varepsilon, 1)(\mathtt{r}, 2)(\mathtt{ic}, 1))^\omega$, then $\mathit{cost}_B^{\{1\},[1,2]}(u) = \sup(\{1, 2, 3\} \cup \{0\}) = 3$. The idea is that if the parity condition is satisfied (as in this example), then the value is the supremum of the checked counter values; otherwise, the counters are ignored and the value is ∞.

A useful variant of this B-objective is the *hB-parity objective*. In this case, the set of counters Γ is totally ordered and whenever a counter is incremented or reset, all lower counters are reset (we say the counters are *hierarchical*). Formally, we let $H_\Gamma := \{c \in \{\varepsilon, \mathtt{ic}, \mathtt{r}\}^\Gamma : c_\gamma \neq \varepsilon \text{ implies } c_{\gamma'} = \mathtt{r} \text{ for all } \gamma' < \gamma\}$ and then $\mathit{Cost}_{hB}^{\Gamma,\Omega} := \langle H_\Gamma \times \Omega, \mathit{cost}_B^{\Gamma,\Omega}, \min\rangle$.

The *S-parity objective* (over counters Γ and priorities Ω) has max as the goal and atomic actions ε, $\mathtt{i}$, $\mathtt{r}$, and $\mathtt{cr}$. It is $\mathit{Cost}_S^{\Gamma,\Omega} := \langle \{\varepsilon, \mathtt{i}, \mathtt{r}, \mathtt{cr}\}^\Gamma \times \Omega, \mathit{cost}_S^{\Gamma,\Omega}, \max\rangle$ where $\mathit{cost}_S^{\Gamma,\Omega}(u) := \inf(C(u) \cup \{\overline{P}(u)\})$ and $\overline{P}(u)$ is 0 (respectively, ∞) if $P(u)$ is ∞ (respectively, 0). In other words, if the parity condition is not satisfied then the counters are ignored and the value assigned is 0; otherwise, the minimum checked value is used (∞ if the counters are never checked).

3.2 Cost Games

A *cost game* $\mathcal{G} := \langle V, v_0, \delta, O\rangle$ consists of a set of positions V, an initial position $v_0 \in V$, an objective $O = \langle \mathbb{C}, f, \mathit{goal}\rangle$ for Eve, and a control function $\delta : V \to$

[1] This B and S notation was originally used in [2] to stand for counters which were $\underline{\text{b}}$ounded and $\underline{\text{s}}$trongly unbounded (whose limit tended towards infinity).

$\mathcal{B}^+(\mathbb{C}\times V)$ (where $\mathcal{B}^+(\mathbb{C}\times V)$ is the set of positive boolean combinations, written as a disjunction of conjunctions of elements from $\mathbb{C} \times V$).

A *play* π is an infinite word $(v_i, c_{i+1}, v_{i+1})_{i\in\mathbb{N}} \in (V \times \mathbb{C} \times V)^\omega$ such that v_0 is the initial position and (c_{i+1}, v_{i+1}) appears in $\delta(v_i)$ for all $i \in \mathbb{N}$. Given a set σ of plays, let $\mathit{pref}(\sigma)$ denote the set of prefixes of plays in σ. We say $(v_0, c_1, v_1), \dots, (v_j, c_{j+1}, v_{j+1}) \in \mathit{pref}(\sigma)$ is a partial play *ending* in v_{j+1} (by convention, we say $\epsilon \in \mathit{pref}(\sigma)$ ends in v_0). At a position $v \in V$, the positive boolean combination given by $\delta(v)$ can be viewed as a subgame in which Eve selects a disjunct in $\delta(v)$ and Adam selects a conjunct within this disjunct. For instance, if there is some partial play π ending in $v \in V$ with $\delta(v) = (c, v) \vee ((c', v') \wedge (c'', v''))$, then Eve can choose a disjunct, say $(c', v') \wedge (c'', v'')$, and Adam can choose one of the conjuncts in this disjunct, say (c'', v''). The play is then extended to $\pi \cdot (v, c'', v'')$ and c'' describes the cost for making this move.

A *strategy* σ for Eve is a set of plays σ such that if a partial play $\pi \in \mathit{pref}(\sigma)$ ends in position v, there must be some disjunct $(c'_1, v'_1) \wedge \ldots \wedge (c'_j, v'_j)$ in $\delta(v)$ such that for every conjunct (c'_i, v'_i) for $i \in [1, j]$, $\pi \cdot (v, c'_i, v'_i) \in \mathit{pref}(\sigma)$. We say σ is *positional* (or memoryless) if Eve's next move depends only on the current position rather than the history of the play (i.e. for all partial plays $\pi, \pi' \in \sigma$ ending in v, $\pi \cdot (v, c', v') \in \mathit{pref}(\sigma)$ if and only if $\pi' \cdot (v, c', v') \in \mathit{pref}(\sigma)$).

The objective $O = \langle \mathbb{C}, f, \mathit{goal} \rangle$ describes how to assign values. For a play $\pi = (v_i, c_{i+1}, v_{i+1})_{i\in\mathbb{N}}$, the value is $\mathit{val}(\pi) := f(\pi_{\mathbb{C}})$ where $\pi_{\mathbb{C}} := c_1 c_2 \ldots$. If *goal* is min, then the value of a strategy σ for Eve is $\mathit{val}(\sigma) := \sup\{\mathit{val}(\pi) : \pi \in \sigma\}$ and the value of the game is $\mathit{val}(\mathcal{G}) := \inf\{\mathit{val}(\sigma) : \sigma \text{ is a strategy for Eve in } \mathcal{G}\}$. In other words, Eve seeks to minimize over all strategies the maximum value of all plays compatible with the strategy. Dually, if *goal* is max, then $\mathit{val}(\sigma) := \inf\{\mathit{val}(\pi) : \pi \in \sigma\}$ and $\mathit{val}(\mathcal{G}) := \sup\{\mathit{val}(\sigma) : \sigma \text{ is a strategy for Eve in } \mathcal{G}\}$. We will refer to games by their objective (e.g. B-parity games).

The *dual* $\overline{\mathcal{G}}$ of a game $\mathcal{G}$ is obtained by switching disjunctions and conjunctions in the control function and using the dual objective (i.e. replacing min with max, and vice versa). This switches the roles of Adam and Eve.

3.3 Cost Automata

An *alternating cost automaton* $\mathcal{A} = \langle Q, \mathbb{A}, q_0, O, \delta \rangle$ has a finite set of states Q, a ranked alphabet $\mathbb{A}$, an initial state $q_0 \in Q$, an objective $O = \langle \mathbb{C}, f, \mathit{goal} \rangle$, and a transition function δ which maps $(q, a) \in Q \times \mathbb{A}_i$ to $\mathcal{B}^+([1, i] \times \mathbb{C} \times Q)$.

Given $t \in \mathcal{T}_{\mathbb{A}}$, we represent $\mathcal{A}$ acting on t via the cost game $\mathcal{A} \times t = \langle Q \times \mathit{pos}(t), (q_0, \epsilon), \delta', O \rangle$ where $\delta'((p, x)) = \delta(p, t(x))[(c, (q, xk))/(k, c, q)]$ and $\phi[s'/s]$ represents the formula ϕ with s' substituted for all occurrences of s. That is, a position in the game corresponds to a state of the automaton and a location in the input tree; the control function δ' modifies the transition function δ of the automaton to map to the appropriate positions in the game. We set $[\![\mathcal{A}]\!](t) := \mathit{val}(\mathcal{A} \times t)$, so $\mathcal{A}$ defines a function $[\![\mathcal{A}]\!] : \mathcal{T}_{\mathbb{A}} \to \mathbb{N}_\infty$. If there is a cost function g such that $[\![\mathcal{A}]\!] \approx g$ then we say that $\mathcal{A}$ *recognizes* g.

Notice that counter and priority actions occur on transitions. It is straightforward to translate between transition-labelled automata and the more common state-labelled automata (at the price of increasing the number of states).

In the simpler case that $\delta(q,a)$ for $a \in \mathbb{A}_i$ is a disjunction of statements of the form $\bigwedge_{j\in[1,i]}(j, c_j, q_j)$, then we say that the cost automaton is *non-deterministic.*

As with cost games, we will describe a cost automaton by its objective. A fundamental result, however, is that the B-, hB-, and S-objectives are equivalent.

Theorem 1. *It is effectively equivalent for a cost function f to be recognizable by a B-parity automaton, an hB-parity automaton, and an S-parity automaton.*

The proof requires results based on composing cost games with "history-deterministic" cost automata which translate between objectives (in analogy to the deterministic automata used in the translation between Muller and parity conditions). This technique was used for finite-duration cost games [7] and can be adapted to infinite cost games, but we will not develop this idea further here.

If the cost-parity automata are given as non-deterministic S-parity and B-parity automata, then it is also possible to decide $\preccurlyeq$.

Lemma 1. *The relation $f_1 \preccurlyeq f_2$ is decidable for cost functions f_1 and f_2 over infinite trees if f_1 is given by a non-deterministic S-parity automaton and f_2 is given by a non-deterministic B-parity automaton.*

The decision procedure is an adaptation of the proof in [7]. It uses ideas from the standard algorithm for deciding language inclusion for regular languages of infinite trees. We construct a product automaton combining f_1 and f_2 and allow Eve to "guess" a tree witnessing $f_1 \not\preccurlyeq f_2$ (this is possible since we are working with non-deterministic automata). In fact, through a series of translations, the decision of $f_1 \not\preccurlyeq f_2$ is reduced to solving a classical parity game (without costs).

4 Weak Cost Automata

We have defined the general framework for cost games and cost automata over infinite trees. In this section, we restrict our attention to a subclass of cost games which are derived from weak cost automata. We are able to show that this subclass defines the same cost functions as cost WMSO, and that weak B-games admit finite-memory strategies. Using these results, we show that it is decidable whether $[\![\varphi]\!] \preccurlyeq [\![\psi]\!]$ for cost WMSO sentences φ and ψ.

4.1 Weak Cost Automata

A *weak B-automaton* $\mathcal{A} = \langle Q, \mathbb{A}, q_0, Cost_B^{\Gamma,[1,2]}, \delta\rangle$ is an alternating B-Büchi automaton such that there is no cycle in δ using both priority 1 and 2. This corresponds to the standard definition (see e.g., [11]) but adapted to the case when priorities label transitions rather than states. A B-Büchi game such that every play stabilizes to moves using only a single priority is called a *weak B-game.* If $\mathcal{A}$ is a weak B-automaton, then $\mathcal{A} \times t$ is a weak B-game for all $t \in \mathcal{T}_{\mathbb{A}}$. Weak hB- and weak S-automata and games are defined by changing the objective.

Example 2. Let $\mathbb{A} = \mathbb{A}_2 = \{a, b, c\}$ and consider a 1-counter weak B-automaton $\mathcal{A} = \langle \{q_0, q_a, q_b, q_\top\}, \mathbb{A}, q_0, Cost_B^{\{1\},[1,2]}, \delta \rangle$.

We describe informally δ such that $\mathcal{A}$ computes the function from Example 1. Adam can either count the number of b's on some branch (incrementing and checking the counter while in state q_b), or prove there are only finitely many a's in the tree. If there are infinitely many a's then there is some branch τ such that an a-labelled node is reachable from each position on τ (but this a-labelled node does not need to be on τ itself). Eve picks out such a branch (marking it with q_0). At any point on this branch, Adam can move to state q_a and force Eve to witness a reachable a-labelled node. If she can, then the play stabilizes in $q_\top$.

The only transitions with priority 1 occur when in state q_a. The automaton can reach q_a only from q_0; once in q_a it can only move to $q_\top$. Thus, there is no cycle in the transition function which visits both priorities, so $\mathcal{A}$ is weak.

A useful notion from [12] is an *alternating chain*, a sequence of states $q_0 \ldots q_n$ such that there is some $p \in [1, 2]$ with q_1 reachable from q_0 using transitions of priority p, and for all $i \in [1, n-1]$, q_{i+1} reachable from q_i using transitions of priority p (respectively, $\overline{p}$) if i is even (respectively, odd) ($\overline{1} = 2$ and $\overline{2} = 1$). We say the *length* of $q_0 q_1 \ldots q_n$ is n. In the example, the automaton has alternating chains of length at most 2 ($q_0 q_a q_\top$). Since the length of these chains is bounded in weak cost automata, it can serve as an induction parameter in proofs.

4.2 Closure Properties

Instead of closure under union and intersection, weak cost automata are closed under min and max (the proof requires disjoint-union and product constructions as in the classical case). More interesting is closure under *weak* inf*-projection* and *weak* sup*-projection.* These operations correspond to finite projection in the classical setting. Let $h : \mathbb{A} \to \mathbb{B}$ be a map from ranked alphabets $\mathbb{A}$ and $\mathbb{B}$ such that $\mathbb{A} \supseteq \mathbb{B}$ and $h(b) = b$ for all $b \in \mathbb{B}$. We write $h(t') = t$ for the natural extension to trees which relabels each $\mathbb{A}$-labelled vertex of t' according to h. If t' contains only finitely many vertices labelled from $\mathbb{A} \setminus \mathbb{B}$, then we write $h_f(t') = t$. *Weak op-projection* of some cost function g over the alphabet $\mathbb{A}$ is the function $g_{op,h_f}(t) = op\,\{g(t') : h_f(t') = t\}$ over the alphabet $\mathbb{B}$ where op is inf or sup.

Generalizing [11, Lemma 1] and using results from [7], we can show weak B (respectively, weak S) automata are closed under weak inf-projection (respectively, weak sup-projection). Given a weak cost automaton for g and a tree t, we simulate it in "non-deterministic mode" on a finite prefix, then switch to "alternating mode" and run the original weak automaton on the remainder of t. While in non-deterministic mode, nodes labelled $b \in \mathbb{B}$ can be relabelled with some $a \in h^{-1}(b)$. By [7], this non-deterministic version on finite trees yields a value $\approx$-equivalent to the original alternating automaton. By the semantics of B (respectively, S) automata, the non-determinism is resolved into taking the infimum (respectively, supremum) of the values of $g(t')$ for t' satisfying $h_f(t') = t$.

Lemma 2. *Weak B-automata are closed under* min*,* max*, weak* inf*-projection. Weak S-automata are closed under* min*,* max*, weak* sup*-projection.*

4.3 Equivalence with Logic

It is no coincidence that we could express the cost WMSO sentence from Example 1 using the weak B-automaton in Example 2.

Theorem 2. *It is effectively equivalent for a cost function f to be definable in cost WMSO and recognizable by a weak cost automaton.*

Proof. (Sketch) To move from logic to automata, we use the standard technique of showing that each atomic formula is recognizable using a weak cost automaton, and then proving that these automata are closed under operations corresponding to other logical constructs (using Theorem 1 and Lemma 2).

To move from a weak cost automaton to a cost WMSO sentence, we do induction on the maximum length m of alternating chain. If $m = 0$, we can write a formula which assigns a value based strictly on the counters (it describes the existence of finite partial runs of a non-deterministic version of the automaton over finite trees, given by [7]). Otherwise, if $m > 0$, we find a finite partial run such that on each path, the automaton started with a transition of priority 1 but has passed through a transition of priority 2 and hence has reached a position with alternating chains strictly less than m (this may require converting between B- and S-versions of the automaton using Theorem 1). The inductive hypothesis yields formulas which correctly capture the value of the automaton on the continuations of the run, so we take the conjunction of these formulas and a formula describing the value on the initial partial run.

4.4 Shape of Strategies

A well-known result in the theory of infinite games is that parity games are positionally determined (see [14] for an introduction to this area). This means that from each position either Adam or Eve can win, and if Eve, say, has a winning strategy, then Eve has a positional strategy which is also winning.

In order to prove results like the decidability of $\preccurlyeq$, it becomes essential to have corresponding results about the strategies needed in cost games. Martin's theorem immediately implies that cost games are determined; in the cost setting, this means that the value of the original game and the dual game is the same.

Proposition 1. *For all cost games, $val(\mathcal{G}) = val(\overline{\mathcal{G}})$.*

There is no bound on the amount of memory Eve would need in order to achieve the optimal value in a cost game. However, in the cost setting, we just need to ensure that there is a positional strategy σ for Eve in $\mathcal{G}$ such that $val(\mathcal{G}) \approx_\alpha val(\sigma)$. If σ satisfies this condition, it means that by playing positionally according to σ, the error committed by Eve is bounded by α. It was shown that positional strategies suffice for finite-duration hB-games [5, 7]. We now prove a similar result for weak hB-games in which the underlying game graph has no cycles. This is a reasonable restriction since the cost games $\mathcal{A} \times t$ where $\mathcal{A}$ is a weak hB-automaton and t is an infinite tree satisfy this requirement. (The result no longer holds for infinite game graphs with cycles.)

We start with an arbitrary strategy τ that witnesses a bounded cost n in a weak hB-game $\mathcal{G}$ with an acyclic game graph, alternating chains of length at most m, and k hierarchical counters. Without loss of generality, we assume that all edges from a position v in the game have the same priority p (denoted $\Omega(v) = p$). We consider the corresponding *strategy tree* T, the tree of all plays consistent with τ. Let V (respectively, S) denote the set of positions in $\mathcal{G}$ (respectively, T). Let $h : S \to V$ denote the homomorphism which maps a position in the strategy tree to the corresponding game position. Using an optimal strategy τ, Eve's choice at a particular $v \in V$ may depend on the history of the play leading to v. Thus, there may be $s, s' \in h^{-1}(v)$ such that the moves possible from s are different than from s'; however, because the game graph is acyclic, s, s', and v must be at the same depth. A positional strategy σ can be viewed as a mapping from V to S which for each v selects a single element of $h^{-1}(v)$ for Eve to use (regardless of the history). To build this map, we use the notion of "signatures".[2]

We first define the components of the signature. For the B-condition, we let $\beta_j(s)$ for $1 \leq j \leq k$ be the number of times a path from s can increment counter j before it is reset. Note that $\beta_j(s) \leq n$ for all $s \in S$ since T has a bounded cost n. The strategy should try to minimize β_j in order to minimize the cost.

For the weak acceptance condition, let $\beta_{\text{alt}}(s)$ be the maximum length of alternating chain on a path from s in T. Just minimizing β_{alt} would not guarantee that the resulting positional strategy satisfies the weak acceptance condition. Thus, we also define inductively a strictly increasing sequence of depths $(d_i)_{i \in \mathbb{N}}$ and a function $\beta : S \to \{0, 1\}$. The depths $(d_i)_{i \in \mathbb{N}}$ "slice" T based on reachability of transitions of priority 2. Let $d_0 := 0$. Given d_i, the depth $d_{i+1} > d_i$ is chosen such that every path from every position in the set $S_i = \{s \in S : s \text{ is at depth } d_i \text{ and } \Omega(s) = 1\}$ visits priority 2 by depth d_{i+1}. This uniform choice is possible since (i) there are only finitely many nodes of priority 1 at a particular depth (because cost games have finite branching), and (ii) from a particular node s with $\Omega(s) = 1$, there is a bound on the length of paths of priority 1 from s (if not, König's Lemma would imply that there is an infinite path of priority 1 in T, which is impossible).

Let s be a node between d_i and d_{i+1}. We set $\beta(s) := 0$ if every path from s can reach priority 2 by d_{i+1}. Otherwise, if $\Omega(s) = 1$ and some path from s cannot reach a node s' with $\Omega(s') = 2$ by d_{i+1}, then $\beta(s) := 1$. The strategy should minimize β_{alt} and β in order to ensure that plays stabilize in priority 2.

The *signature* for $s \in S$ is $sig(s) := \langle \beta_{\text{alt}}(s), \beta(s), \beta_k(s), \beta_{k-1}(s), \ldots, \beta_1(s) \rangle$. Let $\sigma : V \to S$ be the positional strategy which maps v to the element in $h^{-1}(v)$ with the lexicographically-least signature. It turns out that minimizing this signature ensures the weak acceptance condition is satisfied and the value of the play is still bounded.

Theorem 3. *If $\mathcal{G}$ is a weak hB-game with an acyclic game graph, alternating chains of length at most m, and k hierarchical counters, then there is a positional strategy σ (defined above) such that $val(\mathcal{G}) \approx_\alpha val(\sigma)$ for $\alpha(n) = 2m(n+1)^k$.*

[2] We do not use infinite ordinals in the definition of these signatures so, in that sense, it is simpler than many of the classical proofs which use this approach.

It is possible to generalize this technique to show that weak B-games and and B-Büchi games admit finite-memory strategies.

Theorem 4. *For all $k \in \mathbb{N}$, there exists α_k such that for any $\mathcal{G}$ which is a weak B-game (or its dual) or a B-Büchi game, if $\mathcal{G}$ has an acyclic game graph and k counters then there is a finite-memory strategy σ such that $val(\mathcal{G}) \approx_{\alpha_k} val(\sigma)$.*

4.5 Results

Taking advantage of Theorems 3 and 4, we can show that it is possible to simulate a weak cost automaton with a non-deterministic automaton. The idea is that the non-deterministic version guesses a finite-memory strategy in the weak cost game corresponding to the original weak automaton, and then computes its value.

Theorem 5. *If a cost function f is recognizable by a weak cost automaton $\mathcal{A}$ then a non-deterministic B-Büchi automaton $\mathcal{B}$ and non-deterministic S-Büchi automaton $\mathcal{S}$ can be effectively constructed from $\mathcal{A}$ such that $f \approx [\![\mathcal{B}]\!] \approx [\![\mathcal{S}]\!]$.*

The decidability for cost WMSO follows from Theorems 2, 5, and Lemma 1.

Corollary 1. *The relation $[\![\varphi]\!] \preccurlyeq [\![\psi]\!]$ is decidable for any cost WMSO sentences φ and ψ over infinite trees.*

Another nice consequence of Theorem 5 is a separation result. Rabin [13] showed there is a language of infinite trees definable in MSO not definable in WMSO. The separating language L consists of infinite trees over the alphabet $\{a, b\}$ on which every branch has finitely many b's. A similar result holds in the cost setting (and the separating function is the characteristic function of L).

Proposition 2. *There is a cost function over infinite trees definable in cost MSO (in fact, in pure MSO) which is not definable in cost WMSO.*

5 Conclusion

We have extended the framework of cost games and cost automata to infinite trees, building on the work from the finite-tree case [7]. In doing so, we were able to prove that the relations $\preccurlyeq$ and $\approx$ are decidable for cost WMSO, an expressive fragment of cost MSO. The proof relies crucially on the fact that finite-memory strategies suffice in the cost games derived from weak cost automata.

The natural extension of this work would be to show that full cost MSO is decidable over infinite trees. This paper and the work by Colcombet and Löding in [7] have already set the stage for such a result. The missing link is a proof that finite-memory strategies suffice in cost-parity games. As explained in [6], this is a challenging open problem because of the complex interplay between an arbitrary parity condition and the actions of the counters in the cost game.

Another interesting direction would be to compare cost WMSO with the extension of WMSO with the unboundedness operator $\mathbb{U}$, where $\mathbb{U}X.\varphi(X)$ expresses "there is no bound on the size of sets X satisfying φ". This logic has been studied over infinite words in [1] where an equivalent automaton model called deterministic max automata are introduced. These automata lack non-determinism but allow an explicit max operation on counters. It would be interesting to find a similar deterministic model for cost automata over infinite words.

Acknowledgments. I would like to thank Thomas Colcombet, Denis Kuperberg, and the referees for their helpful comments and support.

References

1. Bojanczyk, M.: Weak MSO with the unbounding quantifier. In: Albers, S., Marion, J.Y. (eds.) STACS. LIPIcs, vol. 3, pp. 159–170. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2009)
2. Bojanczyk, M., Colcombet, T.: Bounds in ω-regularity. In: LICS. pp. 285–296. IEEE Computer Society (2006)
3. Colcombet, T.: Regular cost functions over words (2009), manuscript online
4. Colcombet, T.: The theory of stabilisation monoids and regular cost functions. In: ICALP (2). LNCS, vol. 5556, pp. 139–150. Springer (2009)
5. Colcombet, T., Löding, C.: The nesting-depth of disjunctive mu-calculus. In: Kaminski, M., Martini, S. (eds.) CSL. LNCS, vol. 5213, pp. 416–430. Springer (2008)
6. Colcombet, T., Löding, C.: The non-deterministic Mostowski hierarchy and distance-parity automata. In: Aceto, L., Damgard, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP (2). LNCS, vol. 5126, pp. 398–409. Springer (2008)
7. Colcombet, T., Löding, C.: Regular cost functions over finite trees. In: LICS. pp. 70–79. IEEE Computer Society (2010)
8. Hashiguchi, K.: Relative star height, star height and finite automata with distance functions. In: Pin, J.E. (ed.) Formal Properties of Finite Automata and Applications. LNCS, vol. 386, pp. 74–88. Springer (1988)
9. Kirsten, D.: Distance desert automata and the star height problem. RAIRO - Theoretical Informatics and Applications 39(3), 455–509 (2005)
10. Krob, D.: The equality problem for rational series with multiplicities in the tropical semiring is undecidable. International Journal of Algebra and Computation 4, 405–425 (1994)
11. Muller, D.E., Saoudi, A., Schupp, P.E.: Alternating automata. the weak monadic theory of the tree, and its complexity. In: Kott, L. (ed.) ICALP. LNCS, vol. 226, pp. 275–283. Springer (1986)
12. Parigot, M.: Automata, games, and positive monadic theories of trees. In: Nori, K.V. (ed.) FSTTCS. LNCS, vol. 287, pp. 44–57. Springer (1987)
13. Rabin, M.O.: Weakly definable relations and special automata. In: Mathematical Logic and Foundations of Set Theory (Proc. Internat. Colloq., Jerusalem, 1968), pp. 1–23. North-Holland, Amsterdam (1970)
14. Thomas, W.: Languages, automata, and logic. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3: Beyond Words. pp. 389–455. Springer (1997)