

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 $\{i, i + 1, \dots, j\}$? 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 $0 < 1 < \dots < \infty$. For $i \leq j$, we write $[i, j]$ to denote $\{i, i+1, \dots, j\}$. 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 $t : [1, r]^* \rightarrow \mathbb{A}$ such that the domain of t (denoted $\text{pos}(t)$) 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 $a(x, x_1, \dots, x_k)$ which assert that $a \in \mathbb{A}_k$ is the label at position x with children x_1, \dots, 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 $\llbracket \varphi \rrbracket$, from $\mathcal{T}_{\mathbb{A}}$ to \mathbb{N}_∞ (the natural numbers extended with a special infinity symbol ∞). The function is

$$\llbracket \varphi \rrbracket(t) := \inf\{n : t, n \models \varphi\}$$

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 $|X| \leq N$), $\llbracket \varphi \rrbracket(t)$ 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

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

where $\text{chain}(Z)$ 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 $\llbracket \varphi \rrbracket(t) \leq \llbracket \psi \rrbracket(t)$ or $\llbracket \varphi \rrbracket(t) = \llbracket \psi \rrbracket(t)$. This is undecidable even over words by [10], so we must relax the relation being used. Following [4], we define relations \preceq and \approx . Given $f, g : \mathcal{T}_{\mathbb{A}} \rightarrow \mathbb{N}_{\infty}$, we say g *dominates* f (written $f \preceq g$) 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 $f \preceq g$ and $g \preceq f$. Thus, \preceq 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 \preceq and \approx with a *correction function* $\alpha : \mathbb{N} \rightarrow \mathbb{N}$, a non-decreasing function which satisfies $\alpha(n) \geq n$ for all n . We write $f \preceq_{\alpha} g$ 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 $m \preceq_{\alpha} n$ 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 $f : \mathcal{T}_{\mathbb{A}} \rightarrow \mathbb{N}_{\infty}$ and its equivalence class F . The natural decision procedure in this setting is the following: given two cost functions f and g , is $f \preceq g$? 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 $\langle \mathbb{C}, f, \text{goal} \rangle$ where \mathbb{C} is a finite alphabet of actions, $f : \mathbb{C}^{\omega} \rightarrow \mathbb{N}_{\infty}$ maps sequences of actions to a value, and $\text{goal} \in \{\min, \max\}$ describes how a player seeks to optimize f . Switching the goal (from min to max or max to min) yields the dual objective \bar{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}_∞ (instead of only $\{0, \infty\}$). A counter γ is initially assigned value 0 and can be *incremented* **i**, *reset* **r** to 0, *checked* **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 $\{\varepsilon, \mathbf{i}, \mathbf{r}, \mathbf{c}\}$, we define a set $C(u_\gamma) \subseteq \mathbb{N}$ which collects all of the checked values of γ . For instance, $C(\mathbf{i}\mathbf{r}\varepsilon\mathbf{i}\mathbf{i}\mathbf{c}\mathbf{r}\mathbf{i}\mathbf{i}\mathbf{c}) = \{2, 3\}$ 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 $\{\varepsilon, \mathbf{i}, \mathbf{r}, \mathbf{c}\}^\Gamma$, $C(u) := \bigcup_{\gamma \in \Gamma} C(u_\gamma)$ (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 $\text{Cost}_B^{\Gamma, \Omega} := \langle \{\varepsilon, \mathbf{i}, \mathbf{c}, \mathbf{r}\}^\Gamma \times \Omega, \text{cost}_B^{\Gamma, \Omega}, \min \rangle$ where

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

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 , **ic**, and **r**. For example, if $u = ((\mathbf{i}, \mathbf{c}, 2)(\mathbf{i}, \mathbf{c}, 2)(\varepsilon, 1)(\mathbf{r}, 2)(\mathbf{i}, \mathbf{c}, 1))^\omega$, then $\text{cost}_B^{\{1\}, [1, 2]}(u) = \sup(\{1, 2, 3\} \cup \{0\}) = 3$. 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 $H_\Gamma := \{c \in \{\varepsilon, \mathbf{i}, \mathbf{c}, \mathbf{r}\}^\Gamma : c_\gamma \neq \varepsilon \text{ implies } c_{\gamma'} = \mathbf{r} \text{ for all } \gamma' < \gamma\}$ and then $\text{Cost}_{hB}^{\Gamma, \Omega} := \langle H_\Gamma \times \Omega, \text{cost}_B^{\Gamma, \Omega}, \min \rangle$.

The *S-parity objective* (over counters Γ and priorities Ω) has max as the goal and atomic actions ε , **i**, **r**, and **cr**. It is $\text{Cost}_S^{\Gamma, \Omega} := \langle \{\varepsilon, \mathbf{i}, \mathbf{r}, \mathbf{c}\}^\Gamma \times \Omega, \text{cost}_S^{\Gamma, \Omega}, \max \rangle$ where $\text{cost}_S^{\Gamma, \Omega}(u) := \inf(C(u) \cup \{\bar{P}(u)\})$ and $\bar{P}(u)$ 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 $O = \langle \mathbb{C}, f, \text{goal} \rangle$ for Eve, and a control function $\delta : V \rightarrow$

¹ This *B* and *S* notation was originally used in [2] to stand for counters which were bounded and strongly 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 $\text{pref}(\sigma)$ denote the set of prefixes of plays in σ . We say $(v_0, c_1, v_1), \dots, (v_j, c_{j+1}, v_{j+1}) \in \text{pref}(\sigma)$ is a partial play *ending* in v_{j+1} (by convention, we say $\epsilon \in \text{pref}(\sigma)$ 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 $\pi \in \text{pref}(\sigma)$ ends in position v , there must be some disjunct $(c'_1, v'_1) \wedge \dots \wedge (c'_j, v'_j)$ in $\delta(v)$ such that for every conjunct (c'_i, v'_i) for $i \in [1, j]$, $\pi \cdot (v, c'_i, v'_i) \in \text{pref}(\sigma)$. 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 , $\pi \cdot (v, c', v') \in \text{pref}(\sigma)$ if and only if $\pi' \cdot (v, c', v') \in \text{pref}(\sigma)$).

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

The *dual* $\bar{\mathcal{G}}$ 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 $O = \langle \mathbb{C}, f, \text{goal} \rangle$, 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 $\mathcal{A} \times t = \langle Q \times \text{pos}(t), (q_0, \epsilon), \delta', O \rangle$ 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 $\llbracket \mathcal{A} \rrbracket(t) := \text{val}(\mathcal{A} \times t)$, so \mathcal{A} defines a function $\llbracket \mathcal{A} \rrbracket : \mathcal{T}_{\mathbb{A}} \rightarrow \mathbb{N}_\infty$. If there is a cost function g such that $\llbracket \mathcal{A} \rrbracket \approx g$ 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 \preceq .

Lemma 1. *The relation $f_1 \preceq f_2$ 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 $f_1 \not\preceq f_2$ (this is possible since we are working with non-deterministic automata). In fact, through a series of translations, the decision of $f_1 \not\preceq f_2$ 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 $\llbracket \varphi \rrbracket \preceq \llbracket \psi \rrbracket$ 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 $\mathcal{A} = \langle \{q_0, q_a, q_b, q_\top\}, \mathbb{A}, q_0, \text{Cost}_B^{\{1\}, [1, 2]}, \delta \rangle$.

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 $q_0 \dots q_n$ 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, \bar{p}) if i is even (respectively, odd) ($\bar{1} = 2$ and $\bar{2} = 1$). We say the *length* of $q_0 q_1 \dots q_n$ 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 $h : \mathbb{A} \rightarrow \mathbb{B}$ 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 \preceq , 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 $h : S \rightarrow V$ 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 $\beta : S \rightarrow \{0, 1\}$. 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 $\text{sig}(s) := \langle \beta_{\text{alt}}(s), \beta(s), \beta_k(s), \beta_{k-1}(s), \dots, \beta_1(s) \rangle$. Let $\sigma : V \rightarrow S$ 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 $\text{val}(\mathcal{G}) \approx_\alpha \text{val}(\sigma)$ 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 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 $\text{val}(\mathcal{G}) \approx_{\alpha_k} \text{val}(\sigma)$.*

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 $f \approx \llbracket \mathcal{B} \rrbracket \approx \llbracket \mathcal{S} \rrbracket$.*

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

Corollary 1. *The relation $\llbracket \varphi \rrbracket \preceq \llbracket \psi \rrbracket$ 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 \preceq 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., Ingólfssdóttir, 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)

A Appendix

A.1 Cost Automata over Infinite Words

We introduce some material on cost automata over infinite words which is needed in the subsequent proofs. This is based on [4] (on cost automata over finite words) as well as unpublished work by Colcombet extending the results to infinite words.

A non-deterministic cost automaton over infinite words is a tuple $\mathcal{A} = \langle Q, \mathbb{A}, q_0, \langle \mathbb{C}, g, \text{goal} \rangle, \Delta \rangle$. It can be viewed as a non-deterministic cost automaton over trees (as defined in Sect. 3) but where the alphabet is restricted to only symbols of arity 1 ($\mathbb{A} = \mathbb{A}_1$). The semantics are defined accordingly, i.e. for a B -automaton (resp. S -automaton), the value of a run is the maximum (resp. minimum) counter value achieved on the run, and the value of the word is the infimum (resp. supremum) over the values of the accepting runs.

These non-deterministic automata satisfy the following closure properties.

Lemma 3. *The following closure properties hold for non-deterministic cost automata over infinite words:*

- *B-parity: closed under min, max, and inf-projection;*
- *hB-parity: closed under min and inf-projection;*
- *S-parity: closed under min, max, and sup-projection.*

The corresponding results hold for B-Büchi, hB-Büchi, and S-Büchi automata.

Note that the inf-projection of some cost function g over $h : \mathbb{A} \rightarrow \mathbb{B}$ is defined to be $\inf\{g(u') : h(u') = u\}$ (unlike weak inf-projection, h can recolour infinitely many positions); sup-projection is defined accordingly.

History-determinism. In general, cost automaton cannot be determinized. However, they can be made “history-deterministic”. Although this property is not as strong as determinism, history-deterministic cost automata can be composed with cost games in such a way that the objective is changed but the value is preserved (so in this way they behave like deterministic automata).

Let $\mathcal{A} = \langle Q, \mathbb{A}, q_0, \langle \mathbb{C}, g, \text{goal} \rangle, \Delta \rangle$ be a non-deterministic cost automaton over infinite words. A *translation strategy* $\delta : \mathbb{A}^* \times \mathbb{A} \rightarrow \Delta$ for \mathcal{A} describes how to deterministically construct a run of \mathcal{A} over a word $u \in \mathbb{A}^\omega$. We can transform this into a mapping $\tilde{\delta} : \mathbb{A}^\omega \rightarrow \Delta^\omega$ defined such that $\tilde{\delta}(u) = \delta(\epsilon, a_0)\delta(a_0, a_1)\delta(a_0a_1, a_2)\dots$ for $u = a_0a_1a_2\dots \in \mathbb{A}^\omega$. We say δ *drives the run* $\tilde{\delta}(u) \in \Delta^\omega$ if $\tilde{\delta}(u)$ is a valid run of \mathcal{A} over u .

We say \mathcal{A} is α -*history-deterministic* iff for all n there is a translation strategy δ_n such that for all $u \in \mathbb{A}^\omega$, if *goal* is max and $\llbracket \mathcal{A} \rrbracket(u) \geq \alpha(n)$ then $\text{val}(\tilde{\delta}_n(u)) \geq n$, and if *goal* is min and $\llbracket \mathcal{A} \rrbracket(u) \leq n$ then $\text{val}(\tilde{\delta}_n(u)) \leq \alpha(n)$. Thus, by playing according to the translation strategy, the value is \approx_α -equivalent to the value from an arbitrary strategy. This is weaker than the normal notion of determinism since the desired value n , the history of the play, and a letter uniquely determine

the next transition (rather than the current state and a letter). Moreover, an automaton may not be able to actually implement this deterministic choice.

Please see Lemma 5 for examples of history-deterministic automata.

Although cost automata cannot be determinized in general, the automata can be made history-deterministic, and the usual objectives are all equivalent (this is shown for finite words in [4] and for infinite words in unpublished work by Colcombet).

Theorem 6. *It is equivalent for a cost function over words to be recognizable by a B-parity automaton, S-parity automaton, and history-deterministic B-parity automaton (as well as their hierarchical versions).*

Such a cost function is called a *regular cost function*. Over trees, we have seen that two-player games are the primary tool for proofs. Over words, it is algebra that plays the central role in proving results like this (see [4]).

As mentioned earlier, the reason history-deterministic automata are useful is that they can be composed with cost games to yield games with equivalent values but different objectives. Given a cost game $\mathcal{G} = \langle V, \delta, \langle \mathbb{A}, f, goal \rangle \rangle$ and a history-deterministic automaton $\mathcal{A} = \langle Q, \mathbb{A}, q_0, \langle \mathbb{C}, g, goal \rangle, \Delta \rangle$, the *composition of \mathcal{A} and \mathcal{G}* is the cost game $\mathcal{A} \times \mathcal{G} = \langle Q \times V, \delta', \langle \mathbb{C}, g, goal \rangle \rangle$ where

$$\delta'((q, v)) := \delta(v) \left[\bigvee \{ (c, (q', v')) : (q, a, c, q') \in \Delta \} / (a, v') \right].$$

The idea is that a play in the original game \mathcal{G} outputs a word over \mathbb{A} representing the actions on the transitions, and the cost automaton \mathcal{A} (over words) reads this sequence of actions and outputs a new sequence over \mathbb{C} (Eve is given control of any non-determinism in \mathcal{A}). If \mathcal{A} is history-deterministic, this composition results in a game with a different objective but the same value (up to \approx).

Lemma 4. *Let $\mathcal{A} = \langle Q, \mathbb{A}, q_0, \langle \mathbb{C}, g, goal \rangle, \Delta \rangle$ be an α -history-deterministic non-deterministic cost automaton (over infinite words) and $\mathcal{G} = \langle V, \delta, \langle \mathbb{A}, \llbracket \mathcal{A} \rrbracket, goal \rangle \rangle$ be a cost game. Then $val(\mathcal{G}) \approx_\alpha val(\mathcal{A} \times \mathcal{G})$.*

Proof. This is adapted from [7] to the case of infinite trees. Assume that *goal* is max. We first show that $val(\mathcal{A} \times \mathcal{G}) \leq val(\mathcal{G})$. Let σ be a strategy in $\mathcal{A} \times \mathcal{G}$. Every play $\pi \in \sigma$ can be written in the form $((q_0, v_0), c_1, (q_1, v_1))(q_1, v_1), c_2, (q_2, v_2)) \dots$ and can be transformed into a play $\pi_{\mathcal{G}} := (v_0, a_1, v_1)(v_1, a_2, v_2) \dots$ in \mathcal{G} where for each $i > 0$, $(q_{i-1}, a_i, c_i, q_i) \in \Delta$. Note that $val(\pi) = g(c_1 c_2 \dots) \leq \llbracket \mathcal{A} \rrbracket(a_1 a_2 \dots) = val(\pi_{\mathcal{G}})$ since $g(c_1 c_2 \dots)$ is the value of a particular run of \mathcal{A} on $a_1 a_2 \dots$ and $\llbracket \mathcal{A} \rrbracket$ is defined as the maximum value of g applied across all runs. Since the value of a strategy is the infimum over all plays consistent with the strategy, we have $val(\sigma) \leq val(\sigma_{\mathcal{G}})$ where $\sigma_{\mathcal{G}}$ is the result of applying the transformation above to every play in σ ; since this is true for all strategies σ , $val(\mathcal{A} \times \mathcal{G}) \leq val(\mathcal{G})$.

Now we need to show that $val(\mathcal{G}) \preceq_\alpha val(\mathcal{A} \times \mathcal{G})$. Let $\sigma_{\mathcal{G}}$ be a strategy in \mathcal{G} such that $val(\sigma_{\mathcal{G}}) \geq \alpha(n)$ and let δ_n be a translation strategy for the history-deterministic \mathcal{A} . Given a play $\pi_{\mathcal{G}} \in \sigma_{\mathcal{G}}$ of the form $(v_0, a_1, v_1)(v_1, a_2, v_2) \dots$, we know that $\tilde{\delta}_n(a_1 a_2 \dots)$ is of the form $(q_0, a_1, c_1, q_1)(q_1, a_2, c_2, q_2) \dots$. Define

$\pi := ((q_0, v_0), c_1, (q_1, v_1))((q_1, v_1), c_2, (q_2, v_2)) \dots$ which clearly is a play in $\mathcal{A} \times \mathcal{G}$. We know that $\llbracket \mathcal{A} \rrbracket(a_1 a_2 \dots) \geq \alpha(n)$ for if not $\text{val}(\sigma_{\mathcal{G}})$ would be less than $\alpha(n)$. Hence, by the definition of δ_n , $g(c_1 c_2 \dots) \geq n$ (i.e. the run driven by δ_n must have value at least n). Defining σ to be the strategy consisting of all such plays π for $\pi_{\mathcal{G}} \in \sigma_{\mathcal{G}}$, we see that $\text{val}(\sigma) \geq n$ and consequently, $\text{val}(\mathcal{G}) \preceq_{\alpha} \text{val}(\mathcal{A} \times \mathcal{G})$.

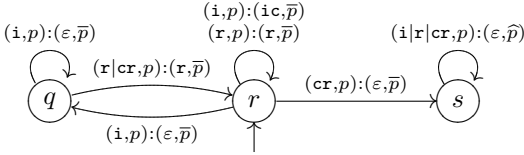
A.2 Proof of Theorem 1 (Duality)

We show that alternating B -parity can be converted to an equivalent alternating S -parity automaton. Moving from alternating S -parity to B -parity is similar. We first describe some history-deterministic cost automata over infinite words which will be needed for the conversion.

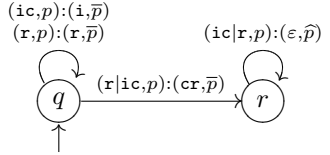
Lemma 5. *The function $\text{cost}_S^{1,\Omega}$ (respectively, $\text{cost}_B^{1,\Omega}$) is recognized by an id-history-deterministic B -automaton (respectively, S -automaton) where id is the identity function.*

We first define cost automata recognizing $\text{cost}_B^{1,\Omega}$ and $\text{cost}_S^{1,\Omega}$ which utilize only one counter. We label a transition with $a : (c, p)$ if on input a , the counter action output is c and the priority output is p . We write, e.g., $\mathbf{r}|\mathbf{cr}$ if that component can be either \mathbf{r} or \mathbf{cr} .

B -automaton $\mathcal{A}_{\text{cost}_S^{1,\Omega}}$:



S -automaton $\mathcal{A}_{\text{cost}_B^{1,\Omega}}$:



where \bar{p} is $p + 1$, and \hat{p} is $p + 1$ if p odd and $p + 2$ if p even.

We describe the operation of the S -automaton $\mathcal{A}_{\text{cost}_B^{1,\Omega}}$. Let $u \in (\{\mathbf{ic}, \mathbf{r}\} \times \Omega)^\omega$. Assume $P(u) = \infty$ because the maximum parity occurring infinitely often is odd, say $2p + 1$. Then the run which stays forever in state q has maximum priority $\overline{2p + 1} = 2p + 2$, so there is an accepting run which never checks the value of the counter and consequently is assigned value ∞ . Because the value of an S -automaton is defined to be the maximum value across all accepting runs, this means that $\llbracket \mathcal{A}_{\text{cost}_B^{1,\Omega}} \rrbracket(u) = \text{cost}_B^{1,\Omega}(u) = \infty$ as desired.

Otherwise, $P(u) = 0$ so any accepting run of $\mathcal{A}_{\text{cost}_B^{1,\Omega}}$ on u must move from state q to r (once in r , the run is guaranteed to be accepting). While in q any \mathbf{ic} is converted to \mathbf{i} for the S -counter (and any \mathbf{r} remains \mathbf{r}), and on the transition from q to r this S -counter value is checked. There is no way for $\mathcal{A}_{\text{cost}_B^{1,\Omega}}$ to achieve a value higher than the value described by the actions in u . Moreover, for every counter value n achieved on u , there is an accepting run of $\mathcal{A}_{\text{cost}_B^{1,\Omega}}$ on u with value n . Again, by taking the maximum value across all accepting runs we get that $\llbracket \mathcal{A}_{\text{cost}_B^{1,\Omega}} \rrbracket(u) = \text{cost}_B^{1,\Omega}(u)$.

The history-determinism is witnessed by the translation strategies δ_n which remain in state q until value n is reached, and then take the transition to state r (and remain there for the remainder of the run). The run $\tilde{\delta}_n(u)$ driven by δ_n satisfies $\text{val}(\tilde{\delta}_n(u)) \geq n$ iff u is rejecting for the parity condition or u is accepting for the parity condition and it achieves a value at least n . The supremum over all n such that δ_n drives $\tilde{\delta}_n(u)$ with $\text{val}(\tilde{\delta}_n(u)) \geq n$ is exactly the value of $\llbracket \mathcal{A}_{\text{cost}_B^{1,\Omega}} \rrbracket(u)$ (so we have that $\mathcal{A}_{\text{cost}_B^{1,\Omega}}$ is *id*-history-deterministic).

In order to define $\mathcal{A}_{\text{cost}_B^{\Gamma,\Omega}}$ for $|\Gamma| > 1$, we take the product of $|\Gamma|$ -many copies of $\mathcal{A}_{\text{cost}_B^{1,\Omega}}$ where each copy is responsible for processing counter actions from a particular counter $\gamma \in \Gamma$. That is, there is a transition from $(s_1, \dots, s_{|\Gamma|})$ to $(s_1, \dots, s'_\gamma, \dots, s_{|\Gamma|})$ labelled $(a, p) : (a', p'')$ if a is a counter action for $\gamma \in \Gamma$ and there is a transition from s_γ to s'_γ with output (a', p') . The value of p'' depends on the states in all of the copies. We set p'' to be p' if $s_\zeta = q_\zeta$ for all $\zeta \in \Gamma$ (i.e. all copies are still in state q). Otherwise there is some copy in state r , and $p'' := \widehat{p}'$ (so the run is guaranteed to be accepting). The idea is that we can conclude that the input has a large value across all counters if just one copy is able to prove (by moving to state r) that the counter has reached a large value.

Duality. Assume we are starting with an alternating B -parity automaton \mathcal{B} with objective $\langle \{\varepsilon, \text{ic}, \mathbf{r}\}^\Gamma \times \Omega, \text{cost}_B^{\Gamma,\Omega}, \min \rangle$. We want to construct an equivalent alternating S -parity automaton \mathcal{S} . Let $\mathcal{A}_{\text{cost}_B^{\Gamma,\Omega}}$ be the *id*-history-deterministic S -automaton (over infinite words) described above which recognizes $\text{cost}_B^{\Gamma,\Omega}$.

Fix some tree t . Then the cost game $\overline{\mathcal{B} \times t}$ is like the game $\mathcal{B} \times t$ but with the roles of the players reversed (conjunctions and disjunctions in the control function switched) and the goal of the objective changed to max. Notice that in this new game, the value for Eve is now the maximum over all strategies σ of the minimum value of all plays $\pi \in \sigma$. This is like an S -automaton except for the fact that the values of the plays π are computed using $\text{cost}_B^{\Gamma,\Omega}$. Roughly speaking, composing with the S -automaton $\mathcal{A}_{\text{cost}_B^{\Gamma,\Omega}}$ results in a cost game with S -actions whose values can be computed using $\text{cost}_S^{\Gamma,\Omega}$ instead.

More formally, let \mathcal{S} be an alternating S -parity automaton such that $\mathcal{S} \times t$ is isomorphic to $\mathcal{A}_{\text{cost}_B^{\Gamma,\Omega}} \times (\overline{\mathcal{B} \times t})$ for all t . Then $\llbracket \mathcal{S} \rrbracket(t) = \text{val}(\mathcal{S} \times t) = \text{val}(\mathcal{A}_{\text{cost}_B^{\Gamma,\Omega}} \times (\overline{\mathcal{B} \times t})) = \text{val}(\overline{\mathcal{B} \times t})$ by Lemma 5. Moreover, by Proposition 1, $\text{val}(\overline{\mathcal{B} \times t}) = \text{val}(\mathcal{B} \times t) = \llbracket \mathcal{B} \rrbracket(t)$. Hence, $\llbracket \mathcal{S} \rrbracket(t) = \llbracket \mathcal{B} \rrbracket(t)$ for all t as desired. Going from alternating B -parity to S -parity is similar.

Note that if the original automaton was a weak cost automaton, then the dual will still be a weak cost automaton (composing with the automata given in Lemma 5 does not introduce cycles which would use both priorities).

Moving from alternating B -parity to alternating hB -parity uses similar ideas, but also incorporates the notion of “latest appearance record” which has been used for the translation between Muller and parity acceptance conditions in the classical setting (see [14] and [7]).

A.3 Proof of Lemma 1 (Decidability)

Let $\mathcal{A}_1 = \langle Q_1, \mathbb{A}, q_0^1, O_1, \Delta_1 \rangle$ be a non-deterministic S -parity automaton (so $O_1 = \text{Cost}_S^{\Gamma_1, \Omega_1}$) and let $\mathcal{A}_2 = \langle Q_2, \mathbb{A}, q_0^2, O_2, \Delta_2 \rangle$ be a non-deterministic B -parity automaton (so $O_2 = \text{Cost}_B^{\Gamma_2, \Omega_2}$). We describe how to decide $\llbracket \mathcal{A}_1 \rrbracket \not\leq \llbracket \mathcal{A}_2 \rrbracket$ using a straightforward adaptation of Colcombet and Löding's proof of the corresponding result for finite trees in [7].

We first apply a standard product construction to combine \mathcal{A}_1 and \mathcal{A}_2 into a single object $\mathcal{A} = \langle Q_1 \times Q_2, (q_0^1, q_0^2), O_1, O_2, \Delta \rangle$ where

$$((p, q), a, ((c_1, d_1), (p_1, q_1)), \dots, ((c_k, d_k), (p_k, q_k))) \in \Delta$$

iff $(p, a, (c_1, p_1), \dots, (c_k, p_k)) \in \Delta_1$ and $(q, a, (d_1, q_1), \dots, (d_k, q_k)) \in \Delta_2$. Strictly speaking, this is not a cost automaton (since there are operations which involve both B counters and S counters and there are objectives of both types). However, we can adapt the definition of a cost game appropriately such that it now has two objectives, and a strategy σ (for Eve) yields two values (denoted $\text{val}_1(\sigma)$ and $\text{val}_2(\sigma)$) based on each objective.

We describe succinctly the initial stages of the construction; please see [7] for further details. Let \mathcal{G} be the cost game in which Eve both selects a tree t and plays her usual role in the game $\mathcal{A} \times t$ (i.e. \mathcal{G} is the cost game $\mathcal{A} \times t$ where positions of t are removed and Eve is controls labels in t). This means each strategy τ for Eve in \mathcal{G} corresponds to a tree t and a strategy σ for Eve in $\mathcal{A} \times t$.

Let $\tilde{\mathcal{G}}$ be the game $\overline{\mathcal{A}_{\text{cost}_S^{\Gamma_1, \Omega_1}} \times \tilde{\mathcal{G}}}$ where $\mathcal{A}_{\text{cost}_S^{\Gamma_1, \Omega_1}}$ is from Lemma 5 and acts only on the first component of the game (i.e. the output related to the S -automaton \mathcal{A}_1). This transformed game now has a new \overline{B} -objective O'_1 as well as the original B -objective O_2 from \mathcal{G} (the \overline{B} -objective O'_1 is the normal B -objective but with min replaced by max; it is easier to translate into an ω -regular winning condition than the original S -objective O_1). By Proposition 1 and Lemma 4, the values of \mathcal{G} and $\tilde{\mathcal{G}}$ are equivalent, and we have the following characterization of the decidability of \preceq .

Lemma 6. $\llbracket \mathcal{A}_1 \rrbracket \not\leq \llbracket \mathcal{A}_2 \rrbracket$ iff there exists $n \in \mathbb{N}$ and a family $(\tau_j)_{j \in \mathbb{N}}$ of strategies for Eve in $\tilde{\mathcal{G}}$ with $\text{val}_1(\tau_j) \geq j$ and $\text{val}_2(\tau_j) \leq n$ for all j .

We now view $\tilde{\mathcal{G}}$ as an ω -regular game. Consider the ω -regular winning condition \mathcal{F} which is the conjunction of:

- the parity condition from \mathcal{A}_1 ;
- the parity condition from \mathcal{A}_2 ;
- every counter in Γ_2 is incremented finitely often or reset infinitely often.

All three of these conditions can be expressed as Muller conditions, and the conjunction of Muller conditions is again a Muller condition. Using standard techniques (see e.g., [14]), we can compute the winning region for Eve in the Muller game $\tilde{\mathcal{G}}$ with winning condition \mathcal{F} . Let $\tilde{\mathcal{G}}'$ be the cost game $\tilde{\mathcal{G}}$ restricted to just this winning region for Eve and let σ be a finite-memory winning strategy for Eve in this game.

Finally, we define another winning condition \mathcal{F}' which is the conjunction of:

- at least one counter in Γ_1 is incremented infinitely often and reset finitely often;
- every counter in Γ_2 is incremented finitely often or reset infinitely often.

Note that the first condition is a Rabin condition and the second condition is a Streett condition. Once again, the conjunction of these conditions can be represented as a Muller condition \mathcal{F}' . We now construct the family of strategies which witness $\llbracket \mathcal{A}_1 \rrbracket \not\leq \llbracket \mathcal{A}_2 \rrbracket$ using the winning strategies in these Muller games.

Lemma 7. *There exists $n \in \mathbb{N}$ and a family $(\tau_j)_{j \in \mathbb{N}}$ of strategies for Eve in $\tilde{\mathcal{G}}$ with $val_1(\tau_j) \geq j$ and $val_2(\tau_j) \leq n$ for all j iff Eve has a winning strategy in the Muller game $\tilde{\mathcal{G}}'$ with winning condition \mathcal{F}' .*

Proof. Assume that Eve has a winning strategy in the Muller game $\tilde{\mathcal{G}}'$ with winning condition \mathcal{F}' . Let σ' be a finite-memory winning strategy for Eve (this exists by finite-memory determinacy of Muller games, see [14]).

The strategy τ_j is defined as the strategy which plays like σ' until a counter from Γ_1 reaches a value greater than j (such a position must exist since the winning condition guarantees at least one counter in Γ_1 is incremented infinitely often and reset only finitely often). After this point, τ_j uses the strategy σ from the Muller game $\tilde{\mathcal{G}}$. This is possible since all of the positions in $\tilde{\mathcal{G}}'$ are in the winning region for $\tilde{\mathcal{G}}$.

We first show that τ_j satisfies the desired properties. The fact that σ is winning for \mathcal{F} means that the parity conditions for both \mathcal{A}_1 and \mathcal{A}_2 are satisfied, so $val_1(\tau_j)$ and $val_2(\tau_j)$ depend on the counters. By construction, $val_1(\tau_j) \geq j$. It remains to show that we can bound $val_2(\tau_j)$. Because σ and σ' are finite-memory strategies, there cannot be an infinite play with a looping segment (a segment which begins and ends at the same position and with the same memory content) which witnesses an increment for a counter from Γ_2 but no reset: if there were such a looping segment, we could pump a play which is consistent with σ or σ' and has infinitely many increments for some counter in Γ_2 but only finitely many resets, a contradiction of σ and σ' winning in the games $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{G}}'$, respectively. Thus, there must be some bounds m and m' for the maximum number of increments before a reset for plays consistent with σ and σ' . Moreover, these values do not depend on the play, but only on the memory of σ and σ' , and the size of the arenas in $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{G}}'$. Overall, the strategy τ_j which uses both σ' and σ must have a bound $n := m' + m$ on the number of increments before a reset for counters in Γ_2 . Hence, there exists n such that $val_2(\tau_j) \leq n$.

Now assume for contradiction that there is an n and a family $(\tau_j)_{j \in \mathbb{N}}$ of strategies for Eve such that $val_1(\tau_j) \geq j$ and $val_2(\tau_j) \leq n$ for all j , but Eve does not have a winning strategy in $\tilde{\mathcal{G}}'$ with winning condition \mathcal{F}' . Finite-memory determinacy of Muller games implies that Adam must consequently have a finite-memory winning strategy $\bar{\sigma}$. Let π_j be a play consistent with Eve's strategy τ_j and Adam's strategy $\bar{\sigma}$. It must be the case that π_j satisfies the parity conditions for \mathcal{A}_1 and \mathcal{A}_2 , otherwise it would be impossible for $val_1(\tau_j) \geq j$ and $val_2(\tau_j) \leq n$. Moreover, we can assume the following looping property (\star) of any looping segment in π_j :

- if a counter from Γ_1 is incremented, then it is reset; or
- there is a counter from Γ_2 which is incremented and not reset.

(If the negation of these properties held on a looping segment, then we could pump a winning play for Eve in $\tilde{\mathcal{G}}'$, contradicting the fact that Adam was using a winning strategy $\bar{\sigma}$.) As described in [7, Lemma 19], we can derive a contradiction using property (\star) as follows.

Choose $j \gg n \cdot k$ where k is the product of the size of memory for $\bar{\sigma}$ and the size of the arena $\tilde{\mathcal{G}}'$. Let β be a segment on which counter $\gamma \in \Gamma_1$ is incremented j times without being reset. Since we are assuming j very large, there is a looping segment β' on which counter γ is incremented $j' = \frac{j}{k}$. Moreover, by property (\star) , there is some counter $\zeta \in \Gamma_2$ which is incremented and not reset. But ζ cannot have been incremented more than n times (by the property of τ_j), hence there is a subsegment β'' which has $j'' = \frac{j}{nk}$ increments of γ but no increments of ζ . Now we repeat this process, starting with β'' , and find a subsegment has at least $\frac{j}{(nk)^2}$ increments of γ but no increments of two counters from Γ_2 . If we continue this process $|\Gamma_2|$ times we will have found a segment which has $\frac{j}{(nk)^{|\Gamma_2|}}$ increments of γ but no increment of *any* counter from Γ_2 . This contradicts (\star) .

A.4 Proof of Lemma 2 (Closure Properties)

We prove the closure of weak B -automata under weak inf-projection. The other constructions are straightforward (and are explained for finite trees in [7]).

The following lemma will allow us to run a non-deterministic automaton on some finite prefix of an infinite tree and then switch seamlessly back to the original alternating automaton, all the while preserving \approx .

Lemma 8. *Let \mathcal{A} be a weak B -automaton (respectively, weak S -automaton) with state-set Q . Then we can effectively construct a non-deterministic B -automaton (respectively, non-deterministic S -automaton) \mathcal{A}_{nd} with state-set Q_{nd} and mapping $\eta : Q_{\text{nd}} \rightarrow \mathcal{P}(Q)$ such that:*

- $\llbracket \mathcal{A} \rrbracket \approx_{\alpha_{\text{nd}}} \llbracket \mathcal{A}_{\text{nd}} \rrbracket$ over finite trees;
- for every partial run R of \mathcal{A}_{nd} on t there is a strategy σ for Eve in the cost game $\mathcal{A} \times t$ such that there is a play $\pi \in \sigma$ with a prefix ending in position (r, x) iff $r \in \eta(R(x))$.

Proof. The proof is a slight modification of the simulation theorem [7, Theorem 12] which makes explicit the states of the copies of the alternating automaton in the state of the non-deterministic version.

Lemma 9. *Weak B -automata (resp., weak S -automata) are closed under weak inf-projection (resp., weak sup-projection).*

Proof. Fix some weak B -automaton $\mathcal{A} = \langle Q_{\mathcal{A}}, \mathbb{A}, q_{\mathcal{A}}^0, \text{Cost}_B^{\Gamma, \Omega}, \delta_{\mathcal{A}} \rangle$ and some translation $h : \mathbb{A} \rightarrow \mathbb{B}$ where $\mathbb{B} \subseteq \mathbb{A}$. Our goal is to construct a weak B -automaton \mathcal{B} recognizing $\llbracket \mathcal{A} \rrbracket_{\text{inf}, h}$.

We first use Lemma 8 to construct $\mathcal{A}_{\text{nd}} = \langle Q_{\text{nd}}, \mathbb{A}, q_{\text{nd}}^0, \text{Cost}_B^T, \Delta_{\text{nd}} \rangle$, a non-deterministic B -automaton over finite trees with a mapping $\eta : Q_{\text{nd}} \rightarrow \mathcal{P}(Q_{\mathcal{A}})$ tracking the states of the copies of \mathcal{A} which would be running at a particular node. The desired automaton \mathcal{B} uses states from $Q_{\text{nd}} \uplus Q_{\mathcal{A}} \uplus \{q_0\}$. The computation proceeds as follows, using the ideas from Lemma 1 in [11].

Given an infinite tree t over the alphabet \mathbb{B} , \mathcal{B} begins in “non-deterministic mode”. In this mode, it can non-deterministically replace a label b with an element of $h^{-1}(b)$ and simulate the non-deterministic B -automaton \mathcal{A}_{nd} on this \mathbb{A} -labelled finite prefix. At any point along a given branch in the computation, \mathcal{B} can switch to “alternating mode” and use the original weak B -automaton \mathcal{A} from that point onwards. Formally, this means that the initial state is $q_{\mathcal{B}}^0 := q_{\text{nd}}^0$ and for $q \in Q_{\text{nd}}$, $\delta_{\mathcal{B}}(q, b)$ is a disjunction of statements of the form

$$\bigwedge_{k \in [1, n]} (k, (c_k, 1), q_k) \vee \bigwedge_{k \in [1, n]} \bigwedge_{r \in (\eta(q_k) \cup q_0)} (k, (c_k, 1), r),$$

for each transition $(q, b, (c_1, q_1), \dots, (c_n, q_n)) \in \Delta_{\text{nd}}$ and each $a \in h^{-1}(b) \subseteq \mathbb{A}_n$. Note that if $\gamma(q_k) = \emptyset$ (no copies of the automaton were sent in direction k), then the automaton enters state q_0 and $\delta_{\mathcal{B}}(q_0, b) = \bigwedge_{k=1}^n (k, (\varepsilon, 2), q_0)$. For $r \in Q_{\mathcal{A}}$, we have $\delta_{\mathcal{B}}(r, b) = \delta_{\mathcal{A}}(r, b)$.

It is clear that this automaton is still an alternating B -automaton. In order for it to be a weak B -automaton, however, we must ensure that there is no cycle in the transition function using both priorities 1 and 2. Note that the automaton uses only priority 1 in the initial non-deterministic mode and then uses the priorities given by \mathcal{A} once it switches to the alternating mode (and it cannot switch back to the non-deterministic mode). Since there is no cycle with both priorities in \mathcal{A} , there is no cycle with both priorities possible in \mathcal{B} .

Now fix a strategy σ for Eve in the cost game $\mathcal{B} \times t$ which satisfies the weak acceptance condition. Such a strategy (if it exists) determines an \mathbb{A} -labelled tree t' , a strategy $\sigma_{\mathcal{A}}$ in $\mathcal{A} \times t'$, and a strategy $\sigma_{\mathcal{A}_{\text{nd}}}$ in $\mathcal{A}_{\text{nd}} \times t'|_{\text{nd}}$ where $t'|_{\text{nd}}$ (respectively, t'_{alt}) is t' restricted to positions where the automaton \mathcal{B} is in “non-deterministic mode” (resp. “alternating mode”) when using strategy σ . We show that $h_f(t') = t$ and $\text{val}(\sigma) \approx \llbracket \mathcal{A} \rrbracket(t')$.

It is clear that $h(t') = t$. However, assume by contradiction that the labels of t and t' differ in infinitely many positions. Then there is some path on which \mathcal{B} never leaves its “non-deterministic mode” (since we can only change symbols of t in this mode). Hence, there is some path which stabilizes on transitions labelled with priority 1, which contradicts the fact that σ satisfies the weak acceptance condition. Thus, $h_f(t') = t$. Now consider all of the plays in σ on a given path τ through t . Restricting to the moves while in non-deterministic mode (finite by the argument above), this induces a single finite play in $\mathcal{A}_{\text{nd}} \times t'|_{\text{nd}}$ on the path $\tau|_{\text{nd}}$ as well as infinite continuations of this play along $\tau|_{\text{alt}}$ in $\mathcal{A} \times t'$. As described above, we can replace the part of the play in $\mathcal{A}_{\text{nd}} \times t'|_{\text{nd}}$ with the corresponding plays in $\mathcal{A} \times t$ while still retaining \approx -equivalence on this finite prefix of τ (we do not need to consider the weak acceptance condition since this is only a finite prefix of the run, and the weak acceptance condition only deals

with infinitary behaviour). Since the plays are identical when \mathcal{B} is in alternating mode, this means that the supremum of the values of the plays in $\mathcal{A} \times t'$ along τ are \approx -equivalent to plays in $\sigma|_\tau$. Hence, $val(\sigma) \approx \llbracket \mathcal{A} \rrbracket(t')$ as desired.

Similarly, we can show that a tree t' accepted by \mathcal{A} (viewed as just a weak tree automaton) and with $h_f(t') = t$ induces a strategy σ in $\mathcal{B} \times t$ satisfying the weak acceptance condition and $\llbracket \mathcal{A} \rrbracket(t') \approx val(\sigma)$.

We have a weak B -automaton \mathcal{B} recognizing the weak inf-projection of \mathcal{A} :

$$\begin{aligned} \llbracket \mathcal{A} \rrbracket_{\text{inf}, h_f}(t) &= \inf\{\llbracket \mathcal{A} \rrbracket(t') : h_f(t') = t\} \\ &= \inf\{\llbracket \mathcal{A} \rrbracket(t') : h_f(t') = t \text{ and } \mathcal{A} \text{ accepts } t'\} \\ &\approx \inf\{val(\sigma) : \sigma \text{ is strategy in } \mathcal{B} \times t \text{ satisfying acc. condition}\} \\ &= \inf\{val(\sigma) : \sigma \text{ is strategy in } \mathcal{B} \times t\} = \llbracket \mathcal{B} \rrbracket(t). \end{aligned}$$

A.5 Proof of Theorem 2 (Automata-Logic Equivalence)

Logic to Automata. We use an equivalent variant of the logic in which only monadic variables are allowed (so the inclusion relation $X \subseteq Y$ is used instead of the membership relation, and each relation $a(x, x_1, \dots, x_k)$ over first-order variables is raised to the relation $a(X, X_1, \dots, X_k)$ over monadic variables which holds if X, X_1, \dots, X_k are singleton sets, the letter at the position given by the singleton set X is a , and its children from left to right are the nodes given by the singleton sets X_1, \dots, X_k). Pushing negation to the leaves, it means that a cost WMSO formula φ can take the form

$$R(X_1, \dots, X_k) \mid \neg R(X_1, \dots, X_k) \mid |X| \leq N \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \exists X. \varphi \mid \forall X. \varphi$$

where φ and ψ are formulas, X, X_1, \dots, X_k are monadic second-order variables, N is the bound variable, and R is some relation of arity k (the inclusion relation \subseteq of arity 2, or the relation $a(X, X_1, \dots, X_k)$ of arity $k + 1$ for $a \in \mathbb{A}_k$).

If $\varphi(X_1, \dots, X_k)$ is a formula with free variables X_1, \dots, X_k (excluding the bound variable), then a *valuation* ν for φ is a mapping from each free variable X_i to a set $V_i \subseteq pos(t)$ of positions. If ν is a valuation, then $\nu[X \mapsto V]$ denotes the new valuation which maps X to V and all other variables X_i to $\nu(X_i)$. We write $t, \nu, n \models \varphi$ if t satisfies φ when free variables are evaluated according to ν and when n takes value N , and $\llbracket \varphi(X_1, \dots, X_k) \rrbracket(t, \nu) := \inf\{n : t, \nu, n \models \varphi\}$. Examining this definition for the particular constructs in the logic we have the following results.

Lemma 10.

$$\llbracket R(X_1, \dots, X_k) \rrbracket(t, \nu) = \begin{cases} 0 & \text{if } t \models R(\nu(X_1), \dots, \nu(X_k)) \\ \infty & \text{otherwise} \end{cases} \quad (1)$$

$$\llbracket \neg R(X_1, \dots, X_k) \rrbracket(t, \nu) = \begin{cases} \infty & \text{if } t \models R(\nu(X_1), \dots, \nu(X_k)) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$\llbracket |X| \leq N \rrbracket(t, \nu) = |\nu(X)| \quad (3)$$

$$\llbracket \varphi \vee \psi \rrbracket(t, \nu) = \min(\llbracket \varphi \rrbracket(t, \nu), \llbracket \psi \rrbracket(t, \nu)) \quad (4)$$

$$\llbracket \varphi \wedge \psi \rrbracket(t, \nu) = \max(\llbracket \varphi \rrbracket(t, \nu), \llbracket \psi \rrbracket(t, \nu)) \quad (5)$$

$$\llbracket \exists X. \varphi \rrbracket(t, \nu) = \inf\{\llbracket \varphi \rrbracket(t, \nu[X \mapsto V]) : V \subseteq \text{pos}(t) \text{ is finite}\} \quad (6)$$

$$\llbracket \forall X. \varphi \rrbracket(t, \nu) = \sup\{\llbracket \varphi \rrbracket(t, \nu[X \mapsto V]) : V \subseteq \text{pos}(t) \text{ is finite}\} \quad (7)$$

The strategy for the translation between logic and automata is standard. We must show that the functions corresponding to the atomic formulas are definable using a weak cost automaton, and prove that weak cost automata are closed under operations corresponding to the other logical constructs.

The languages corresponding to the usual predicates of WMSO over infinite trees given in (1) and (2) are regular and hence their characteristic functions are recognizable by weak cost automata (without any counters) using standard techniques. The predicate $|X| \leq N$ corresponds to the function which computes the cardinality of (the valuation) of some set X of nodes (3). The cost automaton recognizing such a function (up to \approx) is described in [7]. In the case of a binary tree where the set X of nodes is labelled with symbol a , the informal idea for such a cost automaton is that on each branch the automaton computes the maximum of the number of a -labelled nodes and the number of “special nodes” x which have a -labelled nodes in the subtrees starting at x_i in both directions i . This yields the number of a 's in the tree, up to a correction function of $\alpha(n) = 2^n$.

Disjunction (4) and conjunction (5) in the logic correspond to min and max, respectively; second-order existential quantification (6) (respectively, second-order universal quantification (7)) corresponds to weak inf-projection (respectively, weak sup-projection). Lemma 2 and Theorem 1 imply that both weak B -automata and weak S -automata are closed under these desired operations.

Automata to Logic. Fix a weak cost automaton $\mathcal{A} = \langle Q, \mathbb{A}, q_0, O, \delta \rangle$ with counters Γ . We assume that all transitions leading from a given state $q \in Q$ are labelled with the same priority (it is straightforward to convert an arbitrary cost-parity automaton to one satisfying this property). In this case, we write $\Omega(q) = p$ to mean that all transitions leading from q are labelled with priority p . We write \mathcal{A}_q for the automaton \mathcal{A} starting from state q . We want to construct a cost WMSO formula defining an equivalent (up to \approx) function. We first consider a *trivial* acceptance condition (the underlying weak automaton either accepts all trees or rejects all trees).

Lemma 11. *Let \mathcal{A} be a weak cost automaton with trivial acceptance condition. Then for all $q \in Q$, there is a formula $\varphi_q(x)$ of weak cost monadic logic such that $\llbracket \mathcal{A}_q \rrbracket(t_x) \approx_{\alpha_{\text{nd}}} \llbracket \varphi_q(x) \rrbracket(t)$ for all trees t and $x \in \text{pos}(t)$.*

Proof. We assume \mathcal{A} has objective $\text{Cost}_B^{r,[1,2]}$ (using Theorem 1 if necessary). If $\Omega(q) = 1$ for all $q \in Q$, then $\varphi_q(x) := \text{false}$. This means that $\llbracket \varphi_q(x) \rrbracket(t) = \infty$ for all trees t and positions x within t , since we will never be able to satisfy the weak acceptance condition. Otherwise, $\Omega(q) = 2$ for all $q \in Q$, and all runs must satisfy the weak acceptance condition, so we can focus strictly on the counters when defining φ_q . We utilize the non-deterministic cost automaton on finite trees \mathcal{A}_{nd} given by Lemma 8 (which is $\approx_{\alpha_{\text{nd}}}$ -equivalent to \mathcal{A} on finite trees).

Formula φ_q needs to assert that “there exists a bound N such that for every finite prefix of the input tree, there is a run of \mathcal{A}_{nd} on this prefix starting in state q such that the maximum counter value is at most N ”. Formally, $\varphi_q(x)$ is

$$\forall X. \exists \bar{Y}. \exists \bar{Z}. \forall Z'. (Run_B(x, X, \bar{Y}, \bar{Z}) \wedge (Check_B(Z', \bar{Z}) \rightarrow |Z'| \leq N))$$

where \bar{Y} is a tuple of $|Q_{\text{nd}}|$ sets where each set represents the nodes in a particular state, \bar{Z} is a tuple of sets where each set represents nodes where a particular counter operation was performed, and $Run_B(x, X, \bar{Y}, \bar{Z})$ expresses the fact that \bar{Y} and \bar{Z} actually describe a finite partial run of \mathcal{A}_{nd} (starting in state q) on a finite prefix X of the tree t_x . Finally, $Check_B(Z', \bar{Z})$ ensures that Z' is a series of positions along a single branch in the run where a particular counter is incremented and checked, and there are no intermediate positions where the counter is reset. It is straightforward to see that Run_B and $Check_B$ are expressible in WMSO. Moreover, we have $\llbracket \varphi_q(x) \rrbracket(t) = \inf\{n : t, n \models \varphi_q(x)\} \approx_{\alpha_{\text{nd}}} \llbracket \mathcal{A}_q \rrbracket(t_x)$ since $\llbracket \mathcal{A}_{\text{nd}} \rrbracket \approx_{\alpha_{\text{nd}}} \llbracket \mathcal{A} \rrbracket$ by Lemma 8. If there is no bound on the counter values for some input tree t , then there will be no n and no finite sets X, \bar{Y}, \bar{Z} which satisfy the formula and the value ∞ will be correctly assigned (recall $\inf \emptyset = \infty$).

We now need to define a formula when the acceptance condition is not trivial.

Lemma 12. *Let \mathcal{A} be a weak cost automaton with alternating chains of length at most m . Then there is a weak cost monadic formula φ such that $\llbracket \mathcal{A} \rrbracket \approx_{\beta} \llbracket \varphi \rrbracket$ where $\beta(n) := m \cdot \alpha_{\text{nd}}(n)$.*

Proof. We prove a slightly stronger result: for any weak cost automaton \mathcal{A} and for any state $q \in Q$, there is a formula $\varphi_q^{\mathcal{A}}(x)$ such that for all trees t and all positions x in t , $\llbracket \varphi_q^{\mathcal{A}}(x) \rrbracket(t) \approx_{\beta_m} \llbracket \mathcal{A}_q \rrbracket(t_x)$ where t_x denotes the subtree of t starting at position x , \mathcal{A}_q denotes the automaton \mathcal{A} starting from state q , and $\beta_m(n) := m \cdot \alpha_{\text{nd}}(n)$ with m the maximum length of alternating chains in \mathcal{A}_q . The desired formula is then $\varphi := \varphi_{q_0}^{\mathcal{A}}(\epsilon)$. The proof of the stronger result is by induction on the maximum length of alternating chains of \mathcal{A}_q . Without loss of generality, we assume that \mathcal{A} is a simple, hierarchical automaton.

If the maximum length of alternating chains is 1, then \mathcal{A}_q satisfies the conditions for Lemma 11 and we are done. We now assume the maximum length

is m , and the induction hypothesis holds for automata with alternating chains of length less than m . We also assume that $\Omega(q) = 1$ (this is possible since the conversion between B - and S -automata results in the priorities output on the transitions being switched; see Theorem 1 and Lemma 5) and that there are non-trivial counter operations involved (if there are no counters, or the counters are never used, then the classical result [12] can be used to obtain the formula).

If $\Omega(q) = 1$ and \mathcal{A} has a B -objective, then to find the value of \mathcal{A} on t_x , φ_q^A must specify that there exists N and a partial run R_0 of \mathcal{A}_{nd} on a finite prefix of t_x such that the following conditions hold for all y on the frontier of R_0 :

- (i) for each $r \in \eta(R_0(y))$ there is no alternating chain of length m in \mathcal{A}_r ;
- (ii) the maximum counter value checked on the path to y is at most N ;
- (iii) $\bigwedge_{r \in \eta(R_0(y))} \varphi_r^A(xy)$.

It is straightforward to see that (i) and (ii) are expressible in cost WMSO. Because of requirement (i), the formulas $\varphi_r^A(xy)$ in (iii) are well-defined by the inductive hypothesis.

Fix some tree t and position x . We first prove that $\llbracket \varphi_q^A(x) \rrbracket(t) \preceq_{\beta_m} \llbracket \mathcal{A}_q \rrbracket(t_x)$. Assume $\llbracket \mathcal{A}_q \rrbracket(t_x)$ has bounded value n . There must be a run R of \mathcal{A}_q on t_x such that every path satisfies the weak acceptance condition and has checked counter value at most n . On every branch of t_x , there must be a position y such that every copy of \mathcal{A}_q running at y according to R has passed through a transition of priority 2 (if not, then we could construct an infinite path in R which never stabilizes to priority 2, a contradiction). Thus there is a finite prefix of t_x and a partial run R_0 of \mathcal{A}_{nd} on this finite prefix which satisfies condition (i) above. The checked counter values on this prefix may not be bounded by n , but must be bounded by $N := \alpha_{\text{nd}}(n)$ (given by Lemma 8). Moreover, on the extensions of this run from positions y along the frontier of R_0 , the inductive hypothesis implies $\llbracket \varphi_r^A(xy) \rrbracket(t) \approx_{\beta_{m-1}} \llbracket \mathcal{A}_r \rrbracket(t_{xy})$. Therefore, overall, $\llbracket \varphi_q^A(x) \rrbracket(t) \preceq_{\beta_m} \llbracket \mathcal{A}_q \rrbracket(t_x)$.

Next we must show $\llbracket \mathcal{A}_q \rrbracket(t_x) \preceq_{\beta_m} \llbracket \varphi_q^A(x) \rrbracket(t)$. Assume $\llbracket \varphi_q^A(x) \rrbracket(t) = n$. Then there is a run R_0 of \mathcal{A}_{nd} on t_x satisfying (i)-(iii) above. In particular, property (iii) means that the value of the formulas φ_r^A must be bounded for all $r \in \eta(R_0(y))$, so by the inductive hypothesis \mathcal{A}_r is bounded on t_{xy} . This implies that the weak acceptance condition is satisfied on these extensions of the run, and hence the run of \mathcal{A}_q on t_x must also satisfy the weak acceptance condition (since the formula ensures that there is a run which reaches states $r \in \eta(R_0(y))$ at position y). In terms of the counters, it could happen that the maximum counter value in \mathcal{A}_q exceeds the value n given by φ_q^A . This could occur because the formula refers to the non-deterministic automaton \mathcal{A}_{nd} (rather than the original alternating automaton). It could also occur, however, because of the way we are calculating the values on the extensions of the run: the formulas φ_r^A underestimate the maximum counter value if some counter has a non-zero value (in R_0) when it reaches position y along the frontier. Hence the value of \mathcal{A}_q on t_x could be at worst the sum of the value on R_0 ($\alpha_{\text{nd}}(n)$) and the maximum value on the extensions ($\beta_{m-1}(n)$ by the induction hypothesis). Since $\beta_m = \alpha(n) + \beta_{m-1}(n)$, the correction function accounts for these sources of error and we have $\llbracket \mathcal{A}_q \rrbracket(t_x) \preceq_{\beta_m} \llbracket \varphi_q^A(x) \rrbracket(t)$ as desired.

If $\Omega(q) = 1$ and \mathcal{A} has an S -objective then the formula must assign a finite value if for all runs R either there is some path which does not satisfy the weak acceptance condition or there is some path which witnesses a “low” counter value. Because we are using a weak logic, the formula cannot reference the entire run. Instead, the formula $\varphi_q^{\mathcal{A}}(x)$ asserts that there exists a bound N such that for all partial runs R_0 of \mathcal{A}_{nd} on a finite prefix of t_x , either there is some y on the frontier of R_0 such that $\neg\varphi_r^{\mathcal{A}'}(xy)$ for $r \in \gamma(R_0(y))$, or there is some partial run R_1 extending R_0 and there is some y' on the frontier of R_1 such that there is a counter with checked value at most N on the path to y' , where \mathcal{A}' is \mathcal{A} with all counters and counter operations removed. Note that $\varphi_r^{\mathcal{A}'}$ is well-defined since \mathcal{A}' has no counters (so we can use the classical result converting weak automata to weak monadic logic to obtain this formula).

For this case, we can show that $\llbracket \varphi_q^{\mathcal{A}}(x) \rrbracket(t) \approx_{\alpha_{\text{nd}}} \llbracket \mathcal{A}_q \rrbracket(t_x)$ for the correction function α_{nd} given by Lemma 8 (a correction function of α_{nd} is sufficient since the inductive hypothesis is not used). As explained above, \mathcal{A}_q is bounded on t_x iff all runs of \mathcal{A}_q have a path which does not satisfy the weak acceptance condition or witnesses a “low” counter value. The formula takes these two cases into account, and also captures the convention that value ∞ is assigned to paths where no counters are checked. For paths on which the weak acceptance condition is not satisfied, the formula and the automaton assign the same value (0); otherwise, the formula and the automaton may differ by at most α_{nd} (since \mathcal{A}_{nd} is used).

A.6 Proof of Theorem 3 (Positional Strategies)

We use the positional strategy σ as defined in Sect. 4.4 using the signature $\text{sig}(s) := \langle \beta_{\text{alt}}(s), \beta(s), \beta_k(s), \beta_{k-1}(s), \dots, \beta_1(s) \rangle$. We write $s < s'$ if $\text{sig}(s)$ is lexicographically less than $\text{sig}(s')$, and write $v < v'$ if $\sigma(v) < \sigma(v')$.

The formal inductive definition for $(d_i)_{i \in \omega}$ and β is as follows. Let $d_0 := 0$ and $\beta(s_0) := \Omega(s_0) \bmod 2$, for the root s_0 . If all s_i at depth d_i satisfy $\beta(s_i) = 0$, then $d_{i+1} := d_i + 1$. Otherwise, let S_i be the (non-empty) set of nodes s_i at d_i with $\beta(s_i) = 1$, and define d_{i+1} to be the least d such that for every $s_i \in S_i$ and every path from s_i , there is a node s'_i on this path at depth most d and with $\Omega(s'_i) = 2$. For s at $d_i < d \leq d_{i+1}$, let $\beta(s) := 0$ if every path from s reaches some s' satisfying $\Omega(s') = 2$ by d_{i+1} , otherwise $\beta(s) := 1$.

For $\Gamma = [1, k]$ we write IC_j (respectively, \mathbf{R}_j) to denote the hierarchical counter operation c such that $c(j) = \mathbf{ic}$ (respectively, $c(j) = \mathbf{r}$), $c(j') = \varepsilon$ for all $j' > j$ and $c(j') = \mathbf{r}$ for all $j' < j$.

First, we have the following straightforward properties of β_{alt} , β , and β_j .

Lemma 13. *Let T be the strategy tree corresponding to a strategy τ witnessing some bounded value n for a weak hB-game \mathcal{G} with an acyclic game graph and alternating chains of length at most m . Then for any edge $(s, (c, p), s')$ in T ,*

- (a) $\beta_{\text{alt}}(s') \leq \beta_{\text{alt}}(s) \leq m$, and if $\beta_{\text{alt}}(s) = \beta_{\text{alt}}(s')$ then $\beta(s') \leq \beta(s)$;
- (b) $\beta_j(s) \leq n$ for all $1 \leq j \leq k$;
- (c) if $c = \varepsilon$ then $\beta_{j'}(s') \leq \beta_{j'}(s)$ for all $1 \leq j' \leq k$;

- (d) if $c = \mathbf{R}_j$ (for some $1 \leq j \leq k$) then $\beta_{j'}(s') \leq \beta_{j'}(s)$ for all $j < j' \leq k$;
- (e) if $c = \mathbf{IC}_j$ (for some $1 \leq j \leq k$) then $\beta_{j'}(s') \leq \beta_{j'}(s)$ for all $j < j' \leq k$ and $\beta_{j''}(s') < \beta_{j''}(s)$ for some $j < j'' \leq k$.

Playing according to σ ensures that the weak acceptance condition is satisfied.

Lemma 14. *Every play consistent with σ stabilizes in priority 2.*

Proof. Assume not. Then there is some play $(v_0, c_1, v_1)(v_1, c_1, v_2) \dots$ which does not stabilize in priority 2. Consider the point v in the play after which all moves are labelled with priority 1 and have the same length of alternating chains (possible since the length of alternating chains is bounded in weak cost games). This node v must satisfy $\Omega(v) = 1$. Let s be the lexicographically-least element in $h^{-1}(v)$ selected by σ , and assume that s is at depth $d_i \leq d < d_{i+1}$ (where $(d_i)_{i \in \mathbb{N}}$ is the strictly increasing sequence of depths used to define β above).

First assume that $\beta(s) = 0$, and we proceed by induction on $d_{i+1} - d$. By definition of $\beta(s) = 0$, any path from s must witness a transition labelled priority 2 by depth d_{i+1} . If $d_{i+1} - d = 1$, then since σ is playing according to s , this means that the next move must be to a state of priority 2, a contradiction. If $d_{i+1} - d > 1$, then playing according to σ must move to some position s' of priority 1 corresponding to v' in \mathcal{G} . Since $\beta_{\text{alt}}(s) = \beta_{\text{alt}}(s')$, by Lemma 13, we have $\beta(s') \leq \beta(s)$. Thus, the choice at v' may not be s' , but must be some node s'' which also satisfies $\beta(s'') = 0$. Because we are assuming that the game graph does not have cycles, s'' must be at the same depth as v' and s' . Thus, we can apply the inductive hypothesis from s'' to get the desired contradiction.

Now assume $\beta(s) = 1$. It suffices to show we can reach a node s' which satisfies $\beta(s') = 0$ using σ (and then we can apply the reasoning above to derive a contradiction). We do induction on $d_{i+1} - d$. If $d_{i+1} - d = 1$, then the next move must be to state s' which satisfies $\beta(s') = 0$ because d_{i+2} is defined such that all paths from s have seen a transition labelled with priority 2 by that point. In the inductive case, playing according to σ must move to some s' corresponding to v' . If the node s'' selected at v' satisfies $\beta(s'') = 0$, then we are done; otherwise, $\beta(s'') = 1$ and we can apply the inductive hypothesis to reach s''' with $\beta(s''') = 0$.

We are now ready to prove that this positional strategy σ still guarantees a bounded cost in \mathcal{G} . It is clear that $\text{val}(\mathcal{G}) \leq \text{val}(\sigma)$ (since σ is one of the strategies considered in the cost game \mathcal{G}). Thus, it remains to show that the positional strategy σ satisfies $\text{val}(\sigma) \preceq_{\alpha} \text{val}(\mathcal{G}) = \text{val}(\tau)$ where $\alpha(n) := 2m(n+1)^k$.

Suppose for contradiction that $\text{val}(\sigma) > \alpha(\text{val}(\tau))$. By Lemma 14, every play consistent with σ satisfies the acceptance condition, so there must be a play which has a ‘high’ counter value, i.e. there is some counter j (for $1 \leq j \leq k$) which is incremented more than $2m(n+1)^k$ times on a play $(v_0, c_0, v_1)(v_1, c_1, v_2) \dots$. Consider a segment of the play starting with $(v_{i_0}, I_j, v_{i_0+1})$ witnessing more than $2m(n+1)^k$ increments I_j at positions $I = \{i_0, i_1, \dots, i_{2m(n+1)^k}, \dots\}$ without any intermediate resets for counter j or any operations for counters $j' > j$. By Lemma 13, the maximum signature at v_{i_0} is $\langle m, 1, n, \dots, n \rangle$.

If $i \in I$ and $\beta_{\text{alt}}(\sigma(v_i)) = \beta_{\text{alt}}(\sigma(v_{i+1}))$, then $v_{i+1} < v_i$ (and one of β_k, \dots, β_j witnesses this strict decrease in the signature). If $i \notin I$ and $\beta_{\text{alt}}(\sigma(v_i)) = \beta_{\text{alt}}(\sigma(v_{i+1}))$, then $\beta, \beta_k, \dots, \beta_j$ must remain the same or decrease between v_i and v_{i+1} . This means the maximum counter value on a segment where β_{alt} is constant is $2(n+1)^{k-j+1}$. But $\beta_{\text{alt}}(\sigma(v_{i_0})) \leq m$ and β_{alt} can only decrease by Lemma 13. Hence, there can be at most $2m(n+1)^{k-j+1}$ increments in total (with $1 \leq j \leq k$), contradicting the fact that it has more than $2m(n+1)^k$ increments.

A.7 Proof of Theorem 4 (Finite Memory Strategies)

The proof of this result uses the idea of “slicing” the strategy tree. Although the ideas are similar to the proof in the body of the paper, we use a different presentation here (suggested by T. Colcombet) which makes use of the composition of history-deterministic cost automata with cost games as described in Lemma 4.

Fix an arbitrary strategy τ that witnesses a bounded cost n for the hB -Büchi game $\mathcal{G} = \langle V, v_0, \delta, O \rangle$ with $O = \text{Cost}_{hB}^{[1,k],[1,2]}$ and consider the corresponding strategy tree T . We assume that \mathcal{G} has an acyclic game graph, but unlike a weak cost game, there may be plays in the game which visit infinitely-often both priorities. We also assume that \mathcal{G} uses k hierarchical counters. Let \mathbb{C} denote the alphabet of counter actions (i.e. $\mathbb{C} := H_{[1,k]} \times [1,2]$). We define inductively a strictly increasing sequence of depths $(d_i)_{i \in \mathbb{N}}$ where $d_0 = 0$ and d_{i+1} is the least d such that all paths in T have at least one transition labelled with priority 2 between depths d_i and d_{i+1} . This is well-defined because of the finite-branching inherent in hB -Büchi games (for assume that there were no bound on the depth at which a priority 2 transition is reached; then König’s Lemma would imply that there is a path in T with infinitely many priority 1 transitions, a contradiction.)

A play in \mathcal{G} induces an output word $w = c_1 c_2 \dots \in \mathbb{C}^\omega$ (as usual). Consider a modified version w' of this word with a special output symbol, say $\$$, interspersed at exactly the depths d_i for $i > 0$: $w' := c_1 \dots c_{d_1} \$ c_{d_1+1} \dots c_{d_2} \$ c_{d_2+1} \dots$. Consider the function over the alphabet $\mathbb{C}' := \mathbb{C} \cdot \{\epsilon, \$\}$ which reads such a word and outputs ∞ if there is some subword $u = c_{d_i+1} c_{d_i+2} \dots c_{d_{i+1}} \in \mathbb{C}^*$ (between two signals $\$$) such that priority 2 is not in u ; otherwise it assigns the usual counter value from the run. It is not hard to see that there is a B -Büchi automaton \mathcal{D} over the input alphabet \mathbb{C}' and with objective $\langle \mathbb{C}, \text{cost}_B^{[1,k],[1,2]}, \min \rangle$ recognizing this function (it uses the state to remember whether priority 2 has been seen between signals $\$$, but otherwise outputs actions from \mathbb{C} unchanged). By Theorem 6, we can assume \mathcal{D} is history-deterministic.

Next we transform \mathcal{G} into $\mathcal{G}_\tau = \langle V, v_0, \delta', O' \rangle$. If v is not at a depth d_i for any $i \in \mathbb{N}$, then $\delta'(v) := \delta(v)$. Otherwise, for v at depth d_i for some $i \in \mathbb{N}$, we need to update the output to produce the appropriate signals described above: $\delta'(v) := \delta(v)[(c\$, v')/(c, v')]$. In order to assign a value to plays, the objective is $O' = \langle \mathbb{C} \cdot \{\epsilon, \$\}, \llbracket \mathcal{D} \rrbracket, \min \rangle$. This new game has the same value as the original.

Lemma 15. $\text{val}(\mathcal{G}) = \text{val}(\mathcal{G}_\tau)$.

Proof. Recall that the strategy τ for \mathcal{G} witnesses $\text{val}(\mathcal{G}) = \text{val}(\tau) = n$. Playing according to this strategy in \mathcal{G}_τ shows that $\text{val}(\mathcal{G}_\tau) \leq \text{val}(\mathcal{G})$. Now assume by

contradiction that there is a strategy σ in \mathcal{G}_τ such that $\text{val}(\mathcal{G}_\tau) < \text{val}(\mathcal{G})$. By the definition of \mathcal{D} , this strategy σ must satisfy the Büchi acceptance condition; in fact, it must satisfy a stronger version of this condition which requires priority 2 to occur at least once between depths d_i and d_{i+1} for all $i \in \mathbb{N}$. Hence σ played in \mathcal{G} will have the same value as in \mathcal{G}_τ , contradicting the optimality of τ in \mathcal{G} .

The composition $\mathcal{D} \times \mathcal{G}_\tau$ (which is an hB -Büchi game since \mathcal{D} translates \mathcal{G}_τ into an hB -Büchi objective) admits positional strategies.

Lemma 16. *There exists a positional strategy σ such that $\text{val}(\mathcal{D} \times \mathcal{G}_\tau) \approx_\alpha \text{val}(\sigma)$ for $\alpha(n) = (n + 1)^k$.*

Proof. Fix an arbitrary strategy τ' witnessing a bounded cost n in $\mathcal{G}' := \mathcal{D} \times \mathcal{G}_\tau$. For each node s in the corresponding strategy tree T' , we define $\text{sig}(s) := \langle \beta_k(s), \beta_{k-1}(s), \dots, \beta_1(s) \rangle$ where $\beta_j(s)$ is the number of times a path from s can increment counter j before a reset. We use this to construct a positional strategy $\sigma' : V \rightarrow S$ where $\sigma'(v)$ selects the node $s \in h^{-1}(v)$ with the lexicographically-least signature (please refer to Sect. 4.4 for a reminder of this notation).

We must show that σ' satisfies $\text{val}(\sigma') \preceq_\alpha \text{val}(\tau')$ where $\alpha(n) := (n + 1)^k$. Suppose for contradiction that $\text{val}(\sigma') > \alpha(\text{val}(\tau'))$. The first thing to notice is that each play consistent with σ' must satisfy the Büchi condition. Assume not. Then there is some position after which no transitions of priority 2 occur, so there is some i such that no transitions of priority 2 occur between d_i and d_{i+1} . Let v be the position in the play at d_i . We proceed by induction on the length of the play between d_i and d_{i+1} which does not witness transitions of priority 2. If $d = 1$ (i.e. $d_{i+1} - d_i = 1$), then the fact that no transition of priority 2 occurred between d_i and d_{i+1} means that there is some node $s \in h^{-1}(v)$ at depth d_i in T' such that there is a move from s which is not labelled with priority 2. This contradicts the fact that T' is a strategy tree for τ' witnessing value n . Otherwise, let v be the position at depth d_i and let v' be the next position in the play using strategy σ' . We can assume the transition between v and v' is labelled with priority 1 (otherwise we immediately have a contradiction). The important point is that the positions in \mathcal{G}' record the state of \mathcal{D} . This means that any $s' \in h^{-1}(v')$ with parent s in T' (note that $h(s)$ is not necessarily v) must also have an edge that is labelled with priority 1. Because the game graph is acyclic, we can apply the inductive hypothesis from v' to get the desired contradiction.

The only other way $\text{val}(\sigma') > \alpha(\text{val}(\tau'))$ is if there is some counter j (for $1 \leq j \leq k$) which is incremented more than $(n+1)^k$ times on $(v_0, c_0, v_1)(v_1, c_1, v_2) \dots$. Consider a segment of the play starting with $(v_{i_0}, I_j, v_{i_0+1})$ witnessing more than $(n+1)^k$ increments I_j at positions indexed by $I = \{i_0, i_1, \dots, i_{(n+1)^k}, \dots\}$ without any intermediate resets for counter j or any operations for counters $j' > j$. The maximum signature at v_{i_0} is $\langle n, \dots, n \rangle$. If $i \in I$ then $v_{i+1} < v_i$ (and one of the first $k - j + 1$ coordinates witnesses this strict decrease). Otherwise the first $k - j + 1$ coordinates must remain the same or decrease between v_i and v_{i+1} so the maximum counter value is $(n+1)^{k-j+1}$. This means there must be less than $(n+1)^{k-j+1}$ increments on this segment, contradicting the fact that this segment starting at v_{i_0} has more than $(n+1)^k$ increments.

By Lemma 4, $val(\mathcal{G}_\tau) = val(\mathcal{D} \times \mathcal{G}_\tau)$, so putting this together with Lemma 15 we see that $val(\mathcal{D} \times \mathcal{G}_\tau) = val(\mathcal{G})$. Moreover, since there was a positional strategy σ_τ in $\mathcal{D} \times \mathcal{G}_\tau$ such that $val(\sigma_\tau) \approx val(\mathcal{G}_\tau)$, there must be a finite-memory strategy σ_τ in \mathcal{G} (where the memory of the strategy depends on the history-deterministic hB -Büchi automaton \mathcal{D} , which is independent of \mathcal{G}).

Similarly, there is history-deterministic hB -Büchi automaton \mathcal{D}' with $[[\mathcal{D}']] \approx cost_B^{\Gamma, [1, 2]}$ (see [7]). This means that given B -Büchi game \mathcal{G} (with non-hierarchical counters), the game $\mathcal{D} \times (\mathcal{D}' \times \mathcal{G})$ admits positional strategies by Lemma 16, so the original game admits finite-memory strategies (where the memory depends on the number of counters, but otherwise does not depend on \mathcal{G}). Thus, finite memory strategies suffice in B -Büchi games (which include weak B -games).

A.8 Proof of Theorem 5 (Simulation)

The goal in this section is to simulate a weak hB -automaton \mathcal{A} with some non-deterministic B -parity automaton \mathcal{A}_{nd} (simulating with a non-deterministic S -parity automaton is similar). We follow the format in [7, Theorem 12].

Given an input tree t and a positional strategy σ for Eve in the weak hB -game $\mathcal{A} \times t$, we consider the tree t_σ which is annotated with this strategy. In other words, t_σ is a tree over the extended alphabet $\mathbb{A}' = \mathbb{A} \times \mathcal{P}(Q \times \mathbb{C} \times Q \times [1, r])$ where r is the maximum arity of any symbol in \mathbb{A} . Moreover, $t_\sigma(x) := (t(x), S_\sigma(x))$ where $S_\sigma(x)$ is $\{(s, c, s', k) : (c, (s', xk)) \text{ is a possible move from } (p, x) \text{ via } \sigma\}$.

Let τ be a path in t . Given t_σ and τ , we can define $w_\sigma^\tau := (a_0, k_0)(a_1, k_1) \dots$ such that $\tau = k_0 k_1 \dots$ (so $k_i \in [1, r]$) and $a_j = t_\sigma(k_0 k_1 \dots k_j) \in \mathbb{A}'$. We first show that there is a history-deterministic hB -Büchi automaton \mathcal{D} which reads a word w_σ^τ and computes $max\text{-}play(w_\sigma^\tau)$, the supremum of the cost of all plays in σ which stay on τ .

Given $w := w_\sigma^\tau$, $max\text{-}play$ is equal to $\max\{val_{\text{Büchi}}(w), val_{\text{counters}}(w)\}$ where $val_{\text{Büchi}}(w)$ is ∞ if there is some play π described by w which does not stabilize in an accepting partition and 0 otherwise, and $val_{\text{counters}}(w)$ is the maximum counter value achieved on any play π described by w . Since history-deterministic hB -Büchi automata are closed under \max , to get the desired \mathcal{D} it suffices to show $val_{\text{Büchi}}$ and val_{counters} are recognizable by history-deterministic hB -Büchi automata.

We first describe informally a deterministic Büchi automaton $\mathcal{D}_{\text{Büchi}}$ recognizing $val_{\text{Büchi}}$ (this automaton is based on a construction from [11]). The idea is that the state of $\mathcal{D}_{\text{Büchi}}$ includes a “testing set” $E \subseteq Q$ (representing a subset of possible states the automaton could be in based on the word w that is being read). We write E_i for the testing set at position i in the word. The set E_0 is the set of rejecting states described in $w(0)$. If $E_i = \emptyset$, then E_{i+1} is the set of rejecting states described in $w(i+1)$. Otherwise, the testing set E_{i+1} is updated to include all rejecting states q_{i+1} reachable from rejecting states $q_i \in E_i$ by the moves described in $w(i)$ (note that rejecting states reachable from accepting states in $w(i)$ are not added). The Büchi acceptance condition is used to ensure that the testing set is \emptyset infinitely often, i.e. there is no play described in w which stabilizes in rejecting states.

Next, we show that there is a history-deterministic hB -Büchi automaton $\mathcal{D}_{\text{counters}}$ recognizing $\text{val}_{\text{counters}}$. Since $\text{val}_{\text{counters}}$ can ignore the Büchi condition, we can actually obtain the maximum value of counters by looking at larger and larger prefixes of the plays. Given a partial play π , there is an S -automaton which recognizes its B -value (by [7, Lemma 4]). We can then construct a new S -automaton which when reading a prefix of w , non-deterministically selects a partial play described by it, checks that it is a valid partial play in the game, and then computes its B -value using the previous automaton. Given some prefix v of w , this automaton recognizes the maximum of val_B over the partial plays described in v . But by [4, Theorem 1], there is a history-deterministic hB -automaton recognizing the same function. This automaton, now viewed as a history-deterministic hB -Büchi automaton over infinite words, is the desired $\mathcal{D}_{\text{counters}}$. As mentioned above, this is enough to conclude that there is a history-deterministic hB -Büchi automaton \mathcal{D} for max-play .

Let $\mathcal{G}(t_\sigma)$ be the game $\langle \text{pos}(t_\sigma), \epsilon, \delta, O \rangle$. The control function is defined such that $\delta(x) = \bigwedge_{k \in [1, n]} ((a', k), xk)$ where $t_\sigma(x) = a' \in \mathbb{A}'_n$. This means that Adam simply chooses a direction in the tree (and Eve has no choices). The output when going in direction k is exactly the label on the current and the direction k taken. So there is only one strategy for Eve, and any play in $\mathcal{G}(t_\sigma)$ over which Adam chooses a path τ yields an output word which is exactly w_σ^τ . In order to give a value to such a word, the objective is $\langle \mathbb{A}' \times [1, r], \llbracket \mathcal{D} \rrbracket, \min \rangle$. Overall, $\text{val}(\mathcal{G}(t_\sigma))$ is the supremum over all τ of $\llbracket \mathcal{D} \rrbracket(w_\sigma^\tau)$, which is \approx -equivalent to $\text{val}(\sigma)$.

It is straightforward to construct an hB -parity automaton \mathcal{B} on infinite trees such that $\mathcal{B} \times t_\sigma$ is isomorphic to $\mathcal{D} \times \mathcal{G}(t_\sigma)$. But by Lemma 4, $\text{val}(\mathcal{G}(t_\sigma)) \approx \text{val}(\mathcal{D} \times \mathcal{G}(t_\sigma)) = \text{val}(\mathcal{B} \times t_\sigma)$, so $\llbracket \mathcal{B} \rrbracket(t_\sigma) \approx \text{val}(\sigma)$.

By Theorem 3, positional strategies suffice in weak hB -games. This means $\llbracket \mathcal{A} \rrbracket(t) \approx \inf \{ \text{val}(\sigma) : \sigma \text{ is a positional strategy for Eve in } \mathcal{A} \times t \}$. Since \mathcal{B} computes $\text{val}(\sigma)$ for positional strategies σ , the desired non-deterministic hB -parity automaton \mathcal{A}_{nd} is the (\inf, h_σ) -projection of \mathcal{B} , where h_σ removes the annotations (it is straightforward to show non-deterministic hB -parity automata are closed under \inf -projection).

A.9 Proof of Proposition 2 (Separation)

Consider the following languages of infinite trees over the alphabet $\mathbb{A} = \{a, b\}$:

$$\begin{aligned} L &= \{t : \text{some branch in } t \text{ has infinitely many } b\text{'s}\}, \\ L' &= \{t : \text{every branch in } t \text{ has finitely many } b\text{'s}\}. \end{aligned}$$

These languages are used in [13] and [14] to demonstrate the separation of WMSO and MSO over infinite trees. Indeed, both languages are recognizable by Muller automata (and hence definable in MSO). Although there is a non-deterministic Büchi automaton accepting L , there is no non-deterministic Büchi automaton accepting its complement L' . By the characterization theorem of Rabin in [13], this is enough to conclude L and L' are not definable in WMSO.

We use a similar argument to show the separation of cost WMSO and MSO over infinite trees using the characteristic function $\chi_{L'}$ of language L' above (so $\chi_{L'}(t)$ is 0 if $t \in L'$ and ∞ otherwise). We follow the presentation in [14].

Lemma 17. *$\chi_{L'}$ is not recognizable by non-deterministic B -Büchi automata.*

Proof. We define a family $(t_j)_{j \in \mathbb{N}}$ of $\{a, b\}$ -labelled binary trees such that $t_j(x) = b$ iff $x \in (1^+0) \cup (1^+01^+0) \cup \dots \cup (1^+0)^j$. For all $j \in \mathbb{N}$, $t_j \in L'$.

Assume by contradiction that there is a non-deterministic B -Büchi automaton \mathcal{A} with state-set Q and counters Γ such that $\llbracket \mathcal{A} \rrbracket \approx \chi_{L'}$. Let $S = \{t_j : j \in \mathbb{N}\}$. Since $\chi_{L'}$ is bounded on S , $\llbracket \mathcal{A} \rrbracket$ must also be bounded on S ; i.e. there exists some m such that for all $j \in \mathbb{N}$, there must be a run R of \mathcal{A} on t_j that satisfies the Büchi condition and whose maximum counter value is m along every branch.

Now consider the tree $t_{M \times N}$ where $N := |Q|$ and $M := m \cdot |\Gamma|$. Consider an accepting run R of \mathcal{A} on $t_{M \times N}$. Because 1^ω must satisfy the Büchi condition, there must be some k_0 such that the transition leading to 1^{k_0} is accepting (labelled with priority 2). Similarly, $1^{k_0}01^\omega$ must satisfy the Büchi condition, so there must be some k_1 such that the transition leading to $1^{k_0}01^{k_1}$ is accepting. Reasoning in this way, there exists $k_0, \dots, k_{M \times N}$ such that the transitions leading to nodes in the set $K = \{1^{k_0}, 1^{k_0}01^{k_1}, \dots, 1^{k_0}01^{k_1}0 \dots 1^{k_{M \times N}}\}$ are accepting. Because $|K| > M \times N$ and the value of the counters can never exceed M , there must be two nodes $u, v \in K$ (with $u < v$) that in R share the same state and the same values for all $\gamma \in \Gamma$ (either there are no increments in between, or there are resets/increments that result in the counters achieving the same value).

We now construct a new tree t' which is identical to $t_{M \times N}$ except on the subtree beginning at u . At this point in the tree, we pump the segment between u and v (including the subtrees with roots along this segment) infinitely many times. We can also pump this portion of the run-tree to see that this new branch of t' still satisfies the Büchi condition and has counter values bounded by m . Thus, the tree as a whole satisfies the Büchi condition and achieves counter values bounded by m (this property for the other branches follows from $\llbracket \mathcal{A} \rrbracket(t_{M \times N}) \leq m$). Overall, we have $\llbracket \mathcal{A} \rrbracket(t') \leq m$. Because label b occurs infinitely often on the branch that we pumped, however, $\chi_{L'}(t') = \infty$, a contradiction of $\llbracket \mathcal{A} \rrbracket \approx \chi_{L'}$.