Quasi-Weak Cost Automata: A New Variant of Weakness *

Denis Kuperberg[1] and Michael Vanden Boom[2]

1 **LIAFA/CNRS/Université Paris 7, Denis Diderot, France**
`denis.kuperberg@liafa.jussieu.fr`
2 **Department of Computer Science, University of Oxford**
Wolfson Building, Parks Road, Oxford OX1 3QD, England
`michael.vandenboom@cs.ox.ac.uk`

Abstract

Cost automata have a finite set of counters which can be manipulated on each transition but do not affect control flow. Based on the evolution of the counter values, these automata define functions from a domain like words or trees to $\mathbb{N} \cup \{\infty\}$, modulo an equivalence relation which ignores exact values but preserves boundedness properties. These automata have been studied by Colcombet et al. as part of a "theory of regular cost functions", an extension of the theory of regular languages which retains robust equivalences, closure properties, and decidability like the classical theory.

We extend this theory by introducing quasi-weak cost automata. Unlike traditional weak automata which have a hard-coded bound on the number of alternations between accepting and rejecting states, quasi-weak automata bound the alternations using the counter values (which can vary across runs). We show that these automata are strictly more expressive than weak cost automata over infinite trees. The main result is a Rabin-style characterization theorem: a function is quasi-weak definable if and only if it is definable using two dual forms of non-deterministic Büchi cost automata. This yields a new decidability result for cost functions over infinite trees.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Automata, infinite trees, cost functions, weak

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

Cost automata are finite-state machines enriched with counters which can be manipulated on each transition but cannot be used to affect control flow. Based on the evolution of the counter values, these automata define functions 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 boundedness properties. By only considering the functions up to $\approx$, the resulting "theory of regular cost functions" retains many of the equivalences, closure properties, and decidability results of the theory of regular languages [3]. It extends the classical theory 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 automata can count some behaviour within the input structure.

* The research leading to these results has received funding from ANR 2010 BLAN 0202 02 FREC and the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement 259454.

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 1–12

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The development of this theory was motivated by problems which can be reduced to questions of boundedness. For instance, Hashiguchi [7] and later Kirsten [8] used distance and nested-distance desert automata (special forms of cost automata) to prove the decidability of the "star-height problem": given a regular language L of finite words and a 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? Colcombet and Löding [4] have used a similar approach over finite trees. The theory of regular cost functions over finite words [3] and finite trees [6] can be viewed as a unifying framework for these problems.

It is desirable to extend the theory to infinite trees. For instance, the "parity-index problem" asks: given a regular language L of infinite trees and $i < j$, is there a parity automaton using only priorities $\{i, i+1, \ldots, j\}$? This is known to be decidable in some special cases (e.g. for deterministic languages [11]), but a general decision procedure is not known. However, Colcombet and Löding [5] have reduced the parity-index problem to another open problem, namely the decidability of $\approx$ for regular cost functions over infinite trees.

Weak cost automata (recently studied in [13]) are a natural starting point in this line of research on regular cost functions over infinite trees. In the classical setting, weak automata are a restricted form of alternating Büchi automata which have a fixed bound on the number of alternations between accepting and rejecting states across all runs. They were introduced in [10] to characterize the languages definable in weak monadic second-order logic (WMSO), a variant of MSO in which second-order quantifiers are interpreted over finite sets. Prior to this work, Rabin [12] had given an interesting characterization using non-deterministic automata, showing that a language is weakly definable if and only if the language and its complement are non-deterministic Büchi recognizable.

These notions of weakness have received considerable attention because the weakly definable languages are expressive (e.g. they capture CTL), but still admit efficient model-checking [9]. Indeed, in order to improve the efficiency in some model-checking scenarios, Kupferman and Vardi [9] adapted Rabin's work and provided a quadratic translation between non-deterministic Büchi automata and the corresponding weak automaton.

In [13], weak cost automata were shown equivalent to "cost WMSO", in analogy to the classical theory. However, the question of a Rabin-style characterization based on non-deterministic automata remained open and prompted this work.

1.1 Contributions

We continue the study of regular cost functions over infinite trees by introducing a variant of weakness which we call "quasi-weakness". Unlike traditional weak automata which have a hard-coded bound on the number of alternations between accepting and rejecting states, quasi-weak automata bound the alternations using counter values (which can vary across runs). We show that quasi-weak cost automata are strictly more expressive than weak cost automata over infinite trees.

Although there is no notion of complement for a function, there are two dual semantics (B and S) used to define cost functions. We show that quasi-weak B-automata can be simulated by non-deterministic B-Büchi and S-Büchi automata. Combined with results from [13], this implies the decidability of $f \approx g$ when f, g are cost functions defined by quasi-weak B-automata. Consequently, this work extends the class of cost functions over infinite trees for which $\approx$ is known to be decidable. We also provide a non-trivial extension of Kupferman and Vardi's construction [9] to translate equivalent non-deterministic B-Büchi and S-Büchi automata to an equivalent quasi-weak B-automaton (where the equivalence in each case is up to $\approx$). This provides a Rabin-style characterization of the functions definable

using quasi-weak B-automata and marks an interesting departure from the classical theory.

The construction relies on analyzing a composed run of a B-Büchi automaton and S-Büchi automaton. To aid in this analysis, we use BS-automata and introduce a corresponding BS-equivalence relation $\cong$ which can be used to compare cost automata which define not one but two functions (one based on the B-counters and one on the S-counters). Although the B- and S-counter actions in such a composed run can be independent, we show that it is possible to effectively construct an equivalent (up to $\cong$) BS-automaton in which the actions are more structured (namely, the counters are "hierarchical" so they can be totally ordered and manipulating a higher counter resets all lower counters). We believe this may be a useful technique in other situations which require both counter types.

1.2 Organization

We define cost automata on infinite trees in Sect. 2, with semantics based on two-player infinite games. We also introduce the new quasi-weak cost automata and compare to the more traditional weak cost automata. In Sect. 3, we consider automata with both counter types and show how they can be made hierarchical. Finally, in Sect. 4, we describe the other components of the main result, a Rabin-style characterization for quasi-weak cost automata. We conclude with some open questions in Sect. 5.

1.3 Notations

We write $\mathbb{N}$ for the set of non-negative integers and $\mathbb{N}_\infty$ for the set $\mathbb{N} \cup \{\infty\}$, ordered by $0 < 1 < \cdots < \infty$. If $i \leq j$ are integers, $[i, j]$ denotes the set $\{i, i+1, \ldots, j\}$. We fix a finite alphabet $\mathbb{A}$. The set of finite (respectively, infinite) words over $\mathbb{A}$ is $\mathbb{A}^*$ (respectively, $\mathbb{A}^\omega$) and the empty word is ϵ. For notational simplicity we work only with infinite binary trees. Let $\mathcal{T} = \{0,1\}^*$ be the unlabelled infinite binary tree. A *branch* in $\mathcal{T}$ is a word $\pi \in \{0,1\}^\omega$. The set $\mathcal{T}_\mathbb{A}$ of *complete $\mathbb{A}$-labelled binary trees* is composed of mappings $t : \mathcal{T} \to \mathbb{A}$.

Non-decreasing functions $\mathbb{N} \to \mathbb{N}$ will be denoted by letters $\alpha, \beta, \ldots$, and will be extended to $\mathbb{N}_\infty$ by $\alpha(\infty) = \infty$. We call these *correction functions*.

2 Cost Automata

2.1 Cost Functions

Let E be any set, and $\mathcal{F}_E$ be the set of functions $: E \to \mathbb{N}_\infty$. For $f, g \in \mathcal{F}_E$ and α a correction function, we write $f \preccurlyeq_\alpha g$ if $f \leq \alpha \circ g$ (or if we are comparing single values $n, m \in \mathbb{N}$, $n \preccurlyeq_\alpha m$ if $n \leq \alpha(m)$). We write $f \approx_\alpha g$ if $f \preccurlyeq_\alpha g$ and $g \preccurlyeq_\alpha f$. Finally, $f \approx g$ (respectively, $f \preccurlyeq g$) if $f \approx_\alpha g$ (respectively, $f \preccurlyeq_\alpha g$) for some α. The idea is that the *boundedness relation* $\approx$ does not pay attention to exact values, but does preserve the existence of bounds. Remark that $f \not\preccurlyeq g$ if and only if there exists a set $D \subseteq E$ such that g is bounded on D but f is unbounded on D.

A *cost function over E* is an equivalence class of $\mathcal{F}_E/\approx$. In practice, a cost function (denoted $f, g, \ldots$) will be represented by one of its elements in $\mathcal{F}_E$. In this paper, E will usually be $\mathcal{T}_\mathbb{A}$. The functions defined by automata will always be considered as cost functions, i.e. only considered up to $\approx$.

2.2 B- and S-Valuations

Cost automata define functions from $\mathcal{T}_{\mathbb{A}}$ to $\mathbb{N}_\infty$. The valuation is based on both classical acceptance conditions (in this paper, Büchi acceptance) and a finite set of counters Γ.

A counter γ is initially assigned value 0 and can be *incremented* $\mathtt{i}$, *reset* $\mathtt{r}$ to 0, *checked* $\mathtt{c}$, or left unchanged ε. Given an infinite word u_γ over the alphabet $\{\varepsilon, \mathtt{i}, \mathtt{r}, \mathtt{c}\}$, we define a set $C(u_\gamma) \subseteq \mathbb{N}$ which collects the checked values of γ. In the case of a finite set of counters Γ and a word u over $\{\varepsilon, \mathtt{i}, \mathtt{r}, \mathtt{c}\}^\Gamma$, $C(u) := \bigcup_{\gamma\in\Gamma} C(pr_\gamma(u))$ ($pr_\gamma(u)$ is the γ-projection of u).

We will separate counters into two types: B-counters, which accept as atomic actions the set $\mathbb{B} = \{\varepsilon, \mathtt{ic}, \mathtt{r}\}$, and S-counters, with atomic actions $\mathbb{S} = \{\varepsilon, \mathtt{i}, \mathtt{r}, \mathtt{cr}\}$. Given B-counters Γ_B and $u \in (\mathbb{B}^{\Gamma_B})^\omega$, the *$B$-valuation* is $val_B(u) := \sup C(u)$; likewise, given S-counters Γ_S and $u \in (\mathbb{S}^{\Gamma_S})^\omega$, the *$S$-valuation* is $val_S(u) := \inf C(u)$. By convention, $\inf\emptyset = \infty$ and $\sup\emptyset = 0$. For instance $val_B((\mathtt{ic})^\omega) = \infty$, $val_B((\mathtt{icr})^\omega) = 1$, $val_S(\mathtt{i}^{100}\mathtt{cri}\varepsilon\mathtt{icr}(\mathtt{r})^\omega) = 2$, and $val_S(\mathtt{i}^\omega) = \infty$ because the counter is never checked.

In all cases, if the set of counters Γ is $[1, k]$, an action ν is called *hierarchical* if there is some $i \in [1, k]$ such the action ν performs ε on all counters $j > i$, and $\mathtt{r}$ on all counters $j < i$. It means that performing an increment or a reset on counter i resets all counters j below it.

Cost automata are named B-, S-, or BS-automata depending on the type(s) of counters used. They are *hierarchical* (written, e.g. hB-automata) if only hierarchical actions are used.

2.3 B- and S-Automata on Infinite Trees

An *alternating B-Büchi automaton* $\mathcal{A} = \langle Q, \mathbb{A}, q_0, F, \Gamma_B, \delta\rangle$ on infinite trees has a finite set of states Q, alphabet $\mathbb{A}$, initial state $q_0 \in Q$, accepting states F, finite set Γ_B of B-counters, and transition function $\delta : Q \times \mathbb{A} \to \mathcal{B}^+(\{0,1\} \times \mathbb{B}^{\Gamma_B} \times Q)$, where $\mathcal{B}^+(\{0,1\} \times \mathbb{B}^{\Gamma_B} \times Q)$ is the set of positive boolean combinations, written as a disjunction of conjunctions of elements $(d, \nu, q) \in \{0,1\} \times \mathbb{B}^{\Gamma_B} \times Q$. *Alternating S-Büchi automata* are defined in the same way, replacing B-counters by S-counters and $\mathbb{B}$ with $\mathbb{S}$.

We view running a B-automaton (resp. S-automaton) $\mathcal{A}$ on an input tree t as a game $(\mathcal{A}, t)$ between two players : Eve is in charge of the disjunctive choices and tries to minimize (resp. maximize) counter values while satisfying the Büchi condition, and Adam is in charge of the conjunctive choices and tries to maximize (resp. minimize) counter values or show the Büchi condition is not satisfied. Because the transition function is given as a disjunction of conjunctions, we can consider that at each position, Eve first chooses a disjunct, and then Adam chooses a single tuple (d, ν, q) in this disjunct.

A *play* of $\mathcal{A}$ on input t is a sequence $q_0, (d_1, \nu_1, q_1), (d_2, \nu_2, q_2), \ldots$ compatible with t and δ, i.e. q_0 is initial, and for all $i \in \mathbb{N}$, $(d_{i+1}, \nu_{i+1}, q_{i+1})$ appears in $\delta(q_i, t(d_1 \ldots d_i))$.

A *strategy* for Eve (resp. Adam) in the game $(\mathcal{A}, t)$ is a function that fixes the next choice of Eve (resp. Adam), based on the history of the play (resp. the history of the play and Eve's choice of disjunct). A strategy is *finite-memory* if the number of memory states needed for the player to choose the next move is finite. A strategy is *positional* if no memory at all is needed: the player only needs to know the current position. Notice that choosing a strategy for Eve and a strategy for Adam fixes a play in $(\mathcal{A}, t)$. We say a play π is *compatible* with a strategy σ for Eve if there is some strategy σ' for Adam such that σ and σ' yield the play π.

A play π is *accepting* if there is $q \in F$ appearing infinitely often in π (i.e. π satisfies the Büchi acceptance condition). Given a play π from a B-automaton $\mathcal{A}$, the value of π is $val(\pi) := val_B(h_B(\pi))$ if π is accepting, and $val(\pi) = \infty$ otherwise (where h_B is the projection of π to the B-actions). This yields the maximum checked counter value if

the play is accepting, and ∞ otherwise. We assign a value to a strategy σ for Eve by $val(\sigma) := \sup\{val(\pi) : \pi \text{ is compatible with } \sigma\}$. The value of $\mathcal{A}$ over a tree t is $[\![\mathcal{A}]\!]_B(t) := \inf\{val(\sigma) : \sigma \text{ is a strategy for Eve in the game } (\mathcal{A}, t)\}$.

Likewise, in an S-automaton $\mathcal{A}'$, we define $val(\pi) := val_S(h_S(\pi))$ if π is accepting, and 0 otherwise (where h_S is the projection to the S-actions). Once again, counter actions are only considered if the play is accepting (this time the minimum checked value is used), and 0 is assigned to rejecting plays. Then $val(\sigma) := \inf\{val(\pi) : \pi \text{ is compatible with } \sigma\}$, and $[\![\mathcal{A}']\!]_S(t) := \sup\{val(\sigma) : \sigma \text{ is a strategy for Eve in the game } (\mathcal{A}', t)\}$.

We consider $[\![\mathcal{A}]\!]_B$ and $[\![\mathcal{A}']\!]_S$ as cost functions, so we always work modulo the cost function equivalence $\approx$. If it is clear what semantic the automaton uses we will omit the subscript and write just $[\![\mathcal{A}]\!]$ or $[\![\mathcal{A}']\!]$. If $f \approx [\![\mathcal{A}]\!]$ then we say $\mathcal{A}$ *recognizes* the cost function f.

If for all $(q, a) \in Q \times \mathbb{A}$, $\delta(q, a)$ is of the form $\bigvee_i (0, \nu_i, q_i) \wedge (1, \nu'_i, q'_i)$, then we say the automaton is *non-deterministic*. We define a *run* to be the set of possible plays compatible with some fixed strategy of Eve. Since the only choices of Adam are in the branching, a run labels the entire binary tree with states, and choosing a branch yields a unique play of the automaton. A run is accepting if all of its plays are accepting (that is, if it is accepting on all branches). A value is assigned to a run of a B-automaton (resp. S-automaton) by taking the supremum (resp. infimum) of the values across all branches.

Finally, a cost automaton $\mathcal{A} = \langle Q, \mathbb{A}, q_0, F, \Gamma, \delta\rangle$ is *weak* if the state-set Q can be partitioned into $Q_1, \dots, Q_k$ satisfying:

- for all i and for all $q, q' \in Q_i$, $q \in F$ if and only if $q' \in F$;
- if some (d, ν, q) appears in some $\delta(p, a)$ with $p \in Q_i$ and $q \in Q_j$, then $j \leq i$.

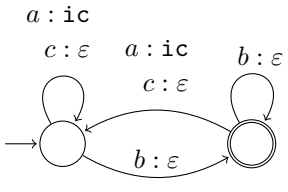
This means there is a fixed bound k on the number of alternations between accepting and rejecting states, so any accepting play must stabilize in an accepting partition.

2.3.1 Examples

Let $\mathbb{A} = \{a, b, c\}$ and let f be the cost function over $\mathbb{A}$-labelled trees where $f(t) = \infty$ if there is a branch with only finitely many b's, and $f(t) = \sup\{|\pi|_a : \pi \text{ is a branch of } t\}$ otherwise, where $|\pi|_a$ denotes the number of a's in π.

We define a non-deterministic B-Büchi automaton $\mathcal{U}$ and a non-deterministic S-Büchi automaton $\mathcal{U}'$, together with a weak automaton $\mathcal{B}$, such that $f \approx [\![\mathcal{U}]\!] \approx [\![\mathcal{U}']\!] \approx [\![\mathcal{B}]\!]$.

The principle of $\mathcal{U}$ is to simultaneously count a's and check for infinitely many b's by running the following deterministic B-automaton on every branch. We write $a : \nu$ to denote that on input a, the counter action is ν; accepting states are denoted by double circles.

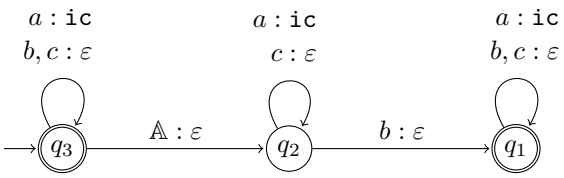


On the other hand, $\mathcal{U}' = \langle \{p_a, p_b, q_b, \top\}, \mathbb{A}, \{p_a, p_b\}, \{q_b, \top\}, \{\gamma\}, \delta\rangle$ tries to find a branch π with either a lot of a's (state p_a), or only finitely many b's (state p_b), in order to witness a high value for f (∞ in the second case). For simplicity, we allow here two initial states, but this does not add expressive power to the model. The state q_b is used when Eve has guessed the position of the last b, and still needs to prove that there are no more b on π, and $\top$ is used when the remainder of the branch does not matter.

The transition table δ for $\mathcal{U}'$ follows. Remark that $\mathcal{U}'$ is in fact a non-deterministic weak S-automaton.

δ	p_a	p_b	q_b	$\top$
a	$((0,\mathtt{i},p_a)\wedge(1,\varepsilon,\top))$ $\vee((0,\varepsilon,\top)\wedge(1,\mathtt{i},p_a))$ $\vee((0,\mathtt{cr},\top)\wedge(1,\varepsilon,\top))$ $\vee((0,\varepsilon,\top)\wedge(1,\mathtt{cr},\top))$	$((0,p_b)\wedge(1,\varepsilon,\top))$ $\vee((0,\varepsilon,\top)\wedge(1,\varepsilon,p_b))$ $\vee((0,\varepsilon,q_b)\wedge(1,\varepsilon,\top))$ $\vee((0,\varepsilon,\top)\wedge(1,\varepsilon,q_b))$	$((0,\varepsilon,q_b)\wedge(1,\varepsilon,\top))$ $\vee((0,\varepsilon,\top)\wedge(1,\varepsilon,q_b))$	$(0,\varepsilon,\top)\wedge(1,\varepsilon,\top)$
b	$((0,\varepsilon,p_a)\wedge(1,\varepsilon,\top))$ $\vee((0,\varepsilon,\top)\wedge(1,\varepsilon,p_a))$ $\vee((0,\mathtt{cr},\top)\wedge(1,\varepsilon,\top))$ $\vee((0,\varepsilon,\top)\wedge(1,\mathtt{cr},\top))$	$=\delta(p_b,a)$	*false* (empty disjunction)	$(0,\varepsilon,\top)\wedge(1,\varepsilon,\top)$
c	$=\delta(p_a,b)$	$=\delta(p_b,a)$	$=\delta(q_b,a)$	$(0,\varepsilon,\top)\wedge(1,\varepsilon,\top)$

Finally, $\mathcal{B}$ is designed such that Adam controls all of the choices: Adam selects a single branch, and runs the following automaton on this branch (he controls the non-determinism):



If there is a branch π with finitely many b's, Adam can select π and stabilize in rejecting state q_2 by moving from q_1 to q_2 after the last b. This witnesses value ∞ for f. Otherwise, Adam tries to select a branch which maximizes the number of a's. The state-set can be partitioned such that $Q_i = \{q_i\}$ for $i \in [1,3]$.

2.4 Quasi-Weak B-Automata

We want to define an extension of weak B-automata, which preserves the property that accepting plays must stabilize in accepting states. The idea of weak automata is to bound the number of alternations between accepting and rejecting states by a hard bound.

Here we have another available tool to bound the number of such alternations: the counters. We know that in a B-automaton, an accepting play of finite value n does at most n increments between resets, but this number is not known a priori by the automaton. Thus, if we guarantee there is correction function α such that in any play π of value n, $\alpha(n)$ is greater than the number of alternations between accepting and rejecting states in π, then we know that any play of finite value must stabilize in accepting states. Otherwise, infinitely many alternations would give value ∞ to the cost function computed by the automaton.

Thus we define quasi-weak automata in the following way:

▶ **Definition 1.** An alternating B-Buchi automaton is *quasi-weak* if there is a correction function α such that in any play of $\mathcal{A}$ of value $n < \infty$, the number of alternations between accepting and rejecting states is smaller than $\alpha(n)$.

In particular, any weak automaton $\mathcal{A}$ is quasi-weak since we can take $\alpha(n) = k$ for all n, where k is the number of partitions of $\mathcal{A}$. We can also give a structural characterization.

▶ **Proposition 2.** *An alternating B-Büchi automaton is quasi-weak if and only if in any reachable cycle containing both accepting and rejecting states, some counter is incremented but not reset.*

We say a cost function is *quasi-weak* if it is recognized by some quasi-weak B-automaton.

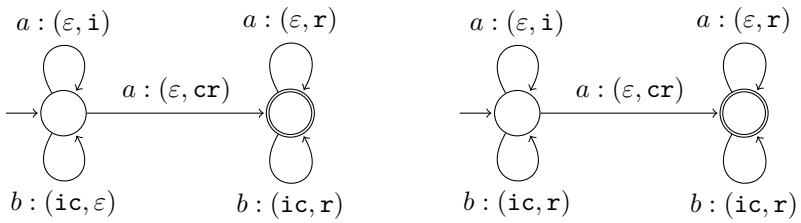
▶ **Proposition 3.** *There exists a cost function over infinite trees which is recognized by a non-deterministic quasi-weak B-automaton, but not by any weak B-automaton. Consequently, quasi-weak B-automata are strictly more expressive than weak B-automata.*

Proof. (Sketch) The idea is to build an explicit cost function f, and for each $n \in \mathbb{N}$ an infinite tree t_n which includes labels that dictate which player controls each position in the game (this is inspired by [1]). These trees are designed such that any alternating B-automaton recognizing f is forced to do $\Theta(n)$ alternations between accepting and rejecting states on t_n. This shows f cannot be computed by a weak B-automaton. On the other hand, we give an explicit non-deterministic quasi-weak B-automaton for f. ◀

3 BS-Automata

We usually work with cost automata with only one type of counter, B or S. In the next section, however, we compose runs from B-Büchi and S-Büchi automata and consequently must work with both counter types simultaneously. We capture this in a *non-deterministic BS-Büchi automaton* $\mathcal{A} = \langle Q, \mathbb{A}, q_0, F_B, F_S, \Gamma_B, \Gamma_S, \Delta \rangle$. Such an automaton defines functions $[\![\mathcal{A}]\!]_B$ and $[\![\mathcal{A}]\!]_S$ as expected (by restricting to one of the counter types).

Let $\mathcal{A}$ and $\mathcal{A}'$ be the following non-deterministic BS-automata on infinite words over $\mathbb{A} := \{a, b, c\}$, each with one B- and one S-counter. We write $a : (d, d')$ if on input a, the output is action d (resp. d') for the B (resp. S) counter. We omit self-loops $c : (\varepsilon, \varepsilon)$.



These automata are very similar. For instance, $[\![\mathcal{A}]\!]_B = [\![\mathcal{A}']\!]_B = |\cdot|_b$. The key difference is $\mathcal{A}'$ is *hierarchical*, with the B-counter above the S-counter. Formally, the counters $\Gamma_B \uplus \Gamma_S$ are globally numbered $[1, k]$ (for $k = |\Gamma_B| + |\Gamma_S|$) and for any action on $\mathbb{B}^{\Gamma_B} \times \mathbb{S}^{\Gamma_S}$ there is some $i \in [1, k]$ such that ε is performed on all counters $j > i$ and $\mathtt{r}$ on all counters $j < i$.

Notice that we have $[\![\mathcal{A}]\!]_S \approx |\cdot|_a$ (if there are a finite number of a's, then the best run of $\mathcal{A}$ moves to the accepting state when reading the final a; otherwise, for every n, there is an accepting run of $\mathcal{A}$ such that the S-counter has value n). In $\mathcal{A}'$, however, the B-counter is higher than the S-counter so $\mathcal{A}'$ forces a reset of the S-counter when a b is read in the initial state. Since there is no a priori bound on the number of b's in the input, this means $[\![\mathcal{A}']\!]_S \not\approx [\![\mathcal{A}]\!]_S$. However, for any fixed m and any u such that $[\![\mathcal{A}]\!]_B(u) \leq m$, the S-value of $\mathcal{A}$ on u is $\approx_{\beta_m}$-equivalent to $\mathcal{A}'$ on u with $\beta_m(n) = n(m+1)$.

This motivates a new equivalence relation $\cong$ which we call *BS-equivalence*. We define $\mathcal{A} \cong \mathcal{A}'$ to hold if there is a correction function α such that (i) $[\![\mathcal{A}]\!]_B \approx_\alpha [\![\mathcal{A}']\!]_B$ and (ii) for any m, there is a correction function β_m such that the S-values of $\mathcal{A}$ and $\mathcal{A}'$ are $\approx_{\beta_m}$-equivalent when restricted to inputs with B-values at most $\alpha(m)$. Although it is technical, this definition captures the notion that two BS-Büchi automata behave in a similar fashion (as in the example above).

It turns out that given any BS-automaton like $\mathcal{A}$, there is an hBS-automaton $\mathcal{A}'$ satisfying $\mathcal{A} \cong \mathcal{A}'$. Moreover, this translation can be done effectively by transducers which

read an infinite word of non-hierarchical counter actions and output hierarchical counter actions. This is in analogy to the deterministic transducer which can be used to translate a Muller condition to a parity condition in the classical setting, or the transducer defined in [6] which translates B-actions to hierarchical B-actions. A similar idea is also used in [2] for automata with both B- and S-counters but in a setting where only boolean properties about boundedness and unboundedness are considered (unlike the quantitative setting here).

▶ **Theorem 4.** *For all sets Γ_B, Γ_S of counters, there exists effectively a history-deterministic hBS-automaton $\mathcal{H}(\Gamma_B, \Gamma_S)$ on infinite words over $\mathbb{B}^{|\Gamma_B|} \times \mathbb{S}^{|\Gamma_S|}$ with $\mathcal{H}(\Gamma_B, \Gamma_S) \cong \mathcal{G}(\Gamma_B, \Gamma_S)$ where $\mathcal{G}(\Gamma_B, \Gamma_S)$ is the BS-automaton which copies the counter actions from the input.*

The transducer $\mathcal{H}(\Gamma_B, \Gamma_S)$ has the same set of B counters, but extra copies of the S-counters. The principle of the automaton is to split the input word into sequences of S-actions from $\{\mathtt{i}, \varepsilon\}^*$ which are between resets of the B-counters. It uses one copy of the S-counter to count the number of S-increments within each sequence, and another copy to count the sequences with at least one S-increment. If the S-value is high compared to the B-value, then the transducer will also have a high S-value, obtained from one of the copies.

These transducers are *history-deterministic*, a weakening of traditional determinism [3]. The entire history of the input and the current state are required to determine the next transition (rather than just the current state and input letter). Because the choice of the transition depends only on the past, for any two input words the automaton can find good moves which do not conflict on any common prefix. This means these automata (like deterministic automata) compose well with alternating automata and games: they can be simulated on each play in a game while preserving the value up to $\approx$ or $\cong$ (see [3] for more information).

This means that we can use the transducers to transform arbitrary BS-automata over words or trees into hierarchical BS-automata which are easier to work with.

4 Characterization of Quasi-Weak Cost Automata

In this section we prove a Rabin-style characterization for quasi-weak B-automata:

▶ **Theorem 5.** *A cost function f over infinite trees is recognizable by some quasi-weak B-automaton $\mathcal{B}$ if and only if there is a non-deterministic B-Büchi automaton $\mathcal{U}$ and non-deterministic S-Büchi automaton $\mathcal{U}'$ such that $f \approx [\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S$.*

The first direction is described in Lemmas 6 and 7 in Section 4.1. The other direction is described in Sections 4.2–4.4, culminating in Theorem 10.

4.1 Simulation

We start by showing that a quasi-weak B-automaton (in fact, any alternating B-Büchi automaton) $\mathcal{A}$ can be simulated by a non-deterministic B-Büchi version.

▶ **Lemma 6.** *Given an alternating B-automaton $\mathcal{B}$, a non-deterministic B-Büchi automaton $\mathcal{U}$ can be effectively constructed such that $[\![\mathcal{B}]\!]_B \approx [\![\mathcal{U}]\!]_B$.*

Proof. (Sketch) In a B-Büchi game, the value of a strategy is the max over all plays compatible with it. Hence, we first show there is a history-deterministic B-Büchi automaton $\mathcal{D}_{\max}$ recognizing $\mathit{max\text{-}play}(w_\sigma^\tau) = \sup\{\mathit{val}(\pi) : \pi \text{ is compatible with } \sigma \text{ and stays on } \tau\}$ on words w_σ^τ which describe the set of plays from a strategy σ which stay on a branch τ.

On input t, the non-deterministic B-Büchi $\mathcal{U}$ guesses a tree t_σ (an annotated version of t over an extended alphabet), checks that the annotations describe a valid finite-memory strategy σ in $(\mathcal{B}, t)$, and simulates $\mathcal{D}_{\max}$ on each branch in t_σ in order to calculate the value of the strategy (possible since $\mathcal{D}_{\max}$ is history-deterministic). Because non-determinism resolves into taking an infimum, $\mathcal{U}$ calculates the infimum over the values of all finite-memory strategies in $(\mathcal{B}, t)$. Although finite-memory strategies might not achieve the optimal value, they do achieve an $\approx$-equivalent value in B-Büchi games by [13]. Hence, $[\![\mathcal{B}]\!] \approx [\![\mathcal{U}]\!]$. ◀

Proving that $\mathcal{B}$ can be simulated by a non-deterministic S-Büchi automaton $\mathcal{U}'$ is more technical and uses the fact that $\mathcal{B}$ is quasi-weak.

▶ **Lemma 7.** *Given a quasi-weak B-automaton $\mathcal{B}$, a non-deterministic S-Büchi automaton $\mathcal{U}'$ can be effectively constructed such that $[\![\mathcal{B}]\!]_B \approx [\![\mathcal{U}']\!]_S$.*

Proof. (Sketch) The automaton $\mathcal{U}'$ can no longer guess a strategy in $(\mathcal{B}, t)$, since the value of $(\mathcal{B}, t)$ is the infimum over all strategies and non-determinism in an S-automaton resolves into taking a supremum. Instead, we consider a dual game $\overline{(\mathcal{B}, t)}$ where the roles of the players are reversed so Eve tries to maximize the B-value across all strategies. We show there is a history-deterministic S-Büchi automaton $\mathcal{D}_{\min}$ which computes the minimum value of a set of plays from such a game, and show these games admit finite-memory strategies. The S-Büchi automaton $\mathcal{U}'$ guesses a finite-memory strategy in such a game and then simulates $\mathcal{D}_{\min}$ on each branch of the tree annotated with this strategy in order to compute its value. ◀

These simulation lemmas and [13, Lemma 1] imply a new decidability result (extending the class of cost functions over infinite trees for which decidability of $\approx$ is known).

▶ **Corollary 8.** *If f, g are cost functions over infinite trees which are given by quasi-weak B-automata then it is decidable whether or not $f \preccurlyeq g$.*

4.2 Construction from Kupferman and Vardi

We now turn to the other direction of Theorem 5. The corresponding classical result states that given non-deterministic Büchi automata $\mathcal{U}$ and $\mathcal{U}'$ such that $L(\mathcal{U})$ is the complement of $L(\mathcal{U}')$, there is a weak automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L(\mathcal{U})$ [9].

The proofs in [12, 9] begin with an analysis of composed runs of $\mathcal{U}$ and $\mathcal{U}'$. Let $m := |Q| \cdot |Q'|$. A *frontier* E is a set of nodes of t such that for any branch π of t, $E \cap \pi$ is a singleton. Kupferman and Vardi [9] define a *trap* for $\mathcal{U}$ and $\mathcal{U}'$ to be a strictly increasing sequence of frontiers $E_0 = \{\epsilon\}, E_1 \dots, E_m$ such that there exists a tree t, a run R of $\mathcal{U}$ on t, and a run R' of $\mathcal{U}'$ on t satisfying the following properties: for all $0 \leq i < m$ and for all branches π in t, there exists $x, x' \in [e_i^\pi, e_{i+1}^\pi)$ such that $R(x) \in F$ and $R'(x') \in F'$ where $e_0^\pi < \dots < e_m^\pi$ is the set of nodes from $E_0, \dots, E_m$ induced by π. The set of positions $[e_i^\pi, e_{i+1}^\pi)$ can be viewed as a block, and each block in a trap witnesses an accepting state from $\mathcal{U}$ and $\mathcal{U}'$.

This is called a trap because $L(\mathcal{U}')$ is the complement of $L(\mathcal{U})$, but a trap implies $L(\mathcal{U}) \cap L(\mathcal{U}') \neq \emptyset$ (using a pumping argument on blocks). The weak automaton $\mathcal{A}$ has Eve (resp. Adam) select a run of $\mathcal{U}$ (resp. $\mathcal{U}'$). The acceptance condition requires that any time an accepting state from $\mathcal{U}'$ is seen, an accepting state from $\mathcal{U}$ is eventually seen. Because of the trap condition, these accepting blocks only need to be counted up to m times (so $\mathcal{A}$ is weak).

4.3 Cost Traps

Now let $\mathcal{U} = \langle Q_{\mathcal{U}}, \mathbb{A}, q_0^{\mathcal{U}}, F_B^{\mathcal{U}}, \Gamma_B^{\mathcal{U}}, \Delta_{\mathcal{U}} \rangle$ (respectively, $\mathcal{U}' = \langle Q_{\mathcal{U}'}, \mathbb{A}, q_0^{\mathcal{U}'}, F_S^{\mathcal{U}'}, \Gamma_S^{\mathcal{U}'}, \Delta_{\mathcal{U}'} \rangle$) be a non-deterministic B-Büchi (respectively, S-Büchi) automaton such that $[\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S$. Our goal is to construct a quasi-weak B-automaton $\mathcal{B}$ which is equivalent to $\mathcal{U}$.

We want to extend the classical case to cost functions, so we seek a notion of "cost trap", which will imply a contradiction with $[\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S$. More specifically, we want a notion of blocks and traps which will witness a bounded B-value from $\mathcal{U}$ on some set of trees but an unbounded S-value for $\mathcal{U}'$ on the same set (showing $[\![\mathcal{U}']\!]_S \not\preccurlyeq [\![\mathcal{U}]\!]_B$). The definition of a block when using arbitrary B- and S-counter actions coming from $\mathcal{U}$ and $\mathcal{U}'$ would be very intricate because it would have to deal with the interaction of the B- and S-actions. In order to avoid this, we switch to working with a non-deterministic hBS-Büchi automaton $\mathcal{A} = \langle Q_{\mathcal{A}}, \mathbb{A}, q_0^{\mathcal{A}}, F_B, F_S, \Gamma_B, \Gamma_S, \delta_{\mathcal{A}} \rangle$ which is BS-equivalent to $\mathcal{U} \times \mathcal{U}' = \langle Q_{\mathcal{U}} \times Q_{\mathcal{U}'}, \mathbb{A}, (q_0^{\mathcal{U}}, q_0^{\mathcal{U}'}), F_B^{\mathcal{U}}, F_S^{\mathcal{U}'}, \Gamma_B^{\mathcal{U}}, \Gamma_S^{\mathcal{U}'}, \Delta_{\mathcal{U}\times\mathcal{U}'} \rangle$ but uses hierarchical counters.

A *block* based on hierarchical BS-actions from $\mathcal{A}$ has accepting states from both F_B and F_S (corresponding to accepting states for $\mathcal{U}$ and $\mathcal{U}'$), but it also has a reset for B-counter γ if γ is incremented in that block (in order to ensure pumping does not inflate the B-value). The number of blocks required is also increased to $m := (|Q_{\mathcal{A}}| + 2)^{|\Gamma_S|+1}$ for technical reasons.

A *cost trap* for $\mathcal{A}$ is a frontier E_m and for every branch π up to E_m a strictly increasing set of nodes $e_0^\pi < \cdots < e_m^\pi \in E_m$ such that there exists a tree t and a run R of $\mathcal{A}$ on t with $val_S(R) > |Q_{\mathcal{A}}|$ satisfying the following properties: for all $0 \leq i < m$ and for all branches π, $[e_i^\pi, e_{i+1}^\pi)$ is a block; if branches π_1 and π_2 share some prefix up to position y and $x < y$ is the first position with $e_i^{\pi_1} = x$ and $e_i^{\pi_2} \neq x$ then $e_i^{\pi_2} > y$ (i.e. pumping blocks from π_2 does not damage blocks from π_1).

A pumping argument shows a cost trap implies $\mathcal{U}$ and $\mathcal{U}'$ are not equivalent.

▶ **Proposition 9.** *Let $\mathcal{U}$ (respectively, $\mathcal{U}'$) be non-deterministic B-Büchi (respectively, S-Büchi). Let $\mathcal{A} \cong \mathcal{U} \times \mathcal{U}'$ be a non-deterministic hBS-automaton. If there exists a cost trap for $\mathcal{A}$, then $[\![\mathcal{U}']\!] \not\preccurlyeq [\![\mathcal{U}]\!]$.*

4.4 Construction of Quasi-Weak B-Automaton $\mathcal{B}$

Given $\mathcal{U}$ and $\mathcal{U}'$ with $[\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S$, we can effectively build a quasi-weak B-automaton $\mathcal{B}$ which on an input tree t,

- simulates in parallel $\mathcal{U}$ (driven by Eve) and $\mathcal{U}'$ (driven by Adam) over t;
- runs the hBS-transducer $\mathcal{H}(\Gamma_B^{\mathcal{U}}, \Gamma_S^{\mathcal{U}'})$ over the composed actions from $\mathcal{U}$ and $\mathcal{U}'$;
- analyzes the output of this transducer together with the accepting states of $\mathcal{U}$ and $\mathcal{U}'$, keeping track of blocks (see below);
- outputs the B-actions of $\mathcal{U}$.

The key difference from the classical case is in the block counting. In [9], the block number only increases and it suffices to count up to a fixed bound. Since each block contains at most 2 alternations between accepting and rejecting states, this results in a weak automaton.

Here, we also have to forbid in any block the presence of an increment for some counter γ without a reset for γ. However, it may be the case that on a branch of a run of $\mathcal{U}$ some counter is incremented but is never reset. So the automaton $\mathcal{B}$ may start counting blocks only to have to restart the counting if an increment is seen which does not have a later reset. But this means that any decrease in the block number corresponds to an increase in the

cost of the play. Hence, the bound on the number of alternations depends on the value of the automaton, which is exactly the property of a quasi-weak automaton.

The idea for the proof that $[\![\mathcal{B}]\!]_B \approx [\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S$ is that if $\mathcal{U}$ accepts some t with low value, then it gives Eve a strategy of the same value in $(\mathcal{B}, t)$. On the other hand, assuming (for the sake of contradiction) that Eve has a low-value strategy in $(\mathcal{B}, t)$ but $\mathcal{U}$ actually assigns t a high value results in a cost trap, which is absurd. Hence, we get the main result:

▶ **Theorem 10.** *If there is a non-deterministic B-Büchi automaton $\mathcal{U}$ and non-deterministic S-Büchi automaton $\mathcal{U}'$ such that $[\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S$, then we can effectively construct a quasi-weak alternating B-automaton $\mathcal{B}$ such that $[\![\mathcal{B}]\!]_B \approx [\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S$.*

We remark that when restricted to languages, this corresponds to the result from [9] since (i) if there are non-deterministic Büchi automata $\mathcal{U}$ and $\mathcal{U}'$ (without counters) recognizing a language and its complement, respectively, then $[\![\mathcal{U}]\!]_B = [\![\mathcal{U}']\!]_S$ and (ii) quasi-weak and weak automata coincide when the automata have no counters.

5 Conclusion

We have introduced quasi-weak cost automata as a variant of weak automata which uses the counters to bound the number of alternations between accepting and rejecting states. We have shown quasi-weak cost automata are strictly more expressive than weak cost automata over infinite trees. Moreover, it is the quasi-weak class of automata, rather than the more traditional weak cost automata, which admits a Rabin-style characterization with non-deterministic B-Büchi and S-Büchi automata. The question of a characterization for weak cost automata over infinite trees remains open (it would likely involve some further restrictions on the actions of the counters in the non-deterministic B-Büchi and S-Büchi automata).

Combined with results from [13], our Rabin-style characterization of quasi-weak automata implies the decidability of $f \preccurlyeq g$ and $f \approx g$ when f, g are defined by quasi-weak B-automata. Consequently, this work extends the class of cost functions over infinite trees for which $\approx$ is known to be decidable. Deciding $\preccurlyeq$ and $\approx$ for all regular cost functions over infinite trees remains a challenging open problem which would imply (by [5]) the decidability of the parity-index problem.

Finally, it was known from [13] that weak cost automata and cost WMSO are equivalent. The logic side of quasi-weak cost automata remains to be explored in future work.

Acknowledgements We are grateful to Thomas Colcombet for having made this joint work possible, and for many helpful discussions.

References

1 André Arnold and Damian Niwinski. Continuous separation of game languages. *Fundam. Inform.*, 81(1-3):19–28, 2007.

2 Mikolaj Bojanczyk and Thomas Colcombet. Bounds in ω-regularity. In *LICS*, pages 285–296. IEEE Computer Society, 2006.

3 Thomas Colcombet. The Theory of Stabilisation Monoids and Regular Cost Functions. In *ICALP (2)*, volume 5556 of *LNCS*, pages 139–150. Springer, 2009.

4 Thomas Colcombet and Christof Löding. The nesting-depth of disjunctive mu-calculus. In Michael Kaminski and Simone Martini, editors, *CSL*, volume 5213 of *LNCS*, pages 416–430. Springer, 2008.

5 Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In Luca Aceto, Ivan Damgard, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *LNCS*, pages 398–409. Springer, 2008.

6 Thomas Colcombet and Christof Löding. Regular cost functions over finite trees. In *LICS*, pages 70–79. IEEE Computer Society, 2010.

7 Kosaburo Hashiguchi. Limitedness theorem on finite automata with distance functions. *J. Comput. Syst. Sci.*, 24(2):233–244, 1982.

8 Daniel Kirsten. Distance desert automata and the star height problem. *RAIRO - Theoretical Informatics and Applications*, 39(3):455–509, 2005.

9 Orna Kupferman and Moshe Y. Vardi. The weakness of self-complementation. In Christoph Meinel and Sophie Tison, editors, *STACS*, volume 1563 of *LNCS*, pages 455–466. Springer, 1999.

10 David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Alternating automata. The weak monadic theory of the tree, and its complexity. In Laurent Kott, editor, *ICALP*, volume 226 of *LNCS*, pages 275–283. Springer, 1986.

11 Damian Niwinski and Igor Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Electr. Notes Theor. Comput. Sci.*, 123:195–208, 2005.

12 Michael O. Rabin. Weakly definable relations and special automata. In *Mathematical Logic and Foundations of Set Theory (Proc. Internat. Colloq., Jerusalem, 1968)*, pages 1–23. North-Holland, Amsterdam, 1970.

13 Michael Vanden Boom. Weak cost monadic logic over infinite trees. In Filip Murlak and Piotr Sankowski, editors, *MFCS*, volume 6907 of *Lecture Notes in Computer Science*, pages 580–591. Springer, 2011.