Two-Way Cost Automata and Cost Logics over Infinite Trees [*]

Achim Blumensath [†]
TU Darmstadt
blumensath@mathematik.tu-darmstadt.de

Thomas Colcombet
Université Paris Diderot
thomas.colcombet@liafa.univ-paris-diderot.fr

Denis Kuperberg
University of Warsaw
denis.kuperberg@gmail.com

Paweł Parys [‡]
University of Warsaw
parys@mimuw.edu.pl

Michael Vanden Boom
University of Oxford
michael.vandenboom@cs.ox.ac.uk

Abstract

Regular cost functions provide a quantitative extension of regular languages that retains most of their important properties, such as expressive power and decidability, at least over finite and infinite words and over finite trees. Much less is known over infinite trees.

We consider cost functions over infinite trees defined by an extension of weak monadic second-order logic with a new fixed-point-like operator. We show this logic to be decidable, improving previously known decidability results for cost logics over infinite trees. The proof relies on an equivalence with a form of automata with counters called quasi-weak cost automata, as well as results about converting two-way alternating cost automata to one-way alternating cost automata.

Categories and Subject Descriptors Theory of computation [*Automata over infinite objects*]

1. Introduction

Boundedness is a central notion arising in both mathematics and computer science. Indeed, being able to determine the existence of bounds is an important form of quantitative reasoning in many contexts, ranging from verification to model theory.

Consider the following boundedness questions:

- Given a finite state automaton with some costly transitions, is there a bound n such that any finite word accepted by the automaton has an accepting run which takes these costly edges at most n times?
- Given some regular language L of finite words, is there a bound n such that L^* (consisting of any finite concatenation of words from L) is equal to $L^0 \cup L^1 \cup \cdots \cup L^n$?
- Given a monadic second-order formula $\varphi(X, x)$ positive in X, is there a bound n such that the least fixed point of φ over finite words is always reached within n iterations?

We could add to this list many other questions like the star height problem from language theory [1–4], the boundedness problem for fixed points of monadic second-order formulae in model theory [5, 6], the bounded repair problem in database theory [7], and the resource bounded reachability problem in verification [8]. We could also consider these problems over different classes of structures (like finite trees or infinite words). The *theory of regular cost functions* introduced in [9] (and developed in subsequent papers) is a powerful extension of the theory of regular languages that enables all of these boundedness questions to be decided in a uniform way.

In the framework of cost functions, the notion of a language is extended to the richer notion of a function from structures (like words or trees) to $\mathbb{N} \cup \{\infty\}$. The central decidability question is whether a function is bounded by some $n \in \mathbb{N}$ over its domain (or a regular subset of its domain). By identifying a language with its characteristic function mapping structures in the language to 0 and everything else to ∞, these boundedness questions subsume classical language decision problems like universality and emptiness.

Like regular languages, these regular cost functions can be described using a number of different formalisms. One way to define these regular cost functions is using *cost monadic logic*, an extension of monadic second-order logic with an operator $|X| \leq N$ which enables some limited reasoning about the cardinality of sets (enough to distinguish notions of "large" and "small" for the purposes of boundedness). These regular cost functions can also be defined in terms of *cost automata*, which are traditional automata enriched with a finite set of counters that can be incremented, reset, or left unchanged on each transition, and are used to assign a value from $\mathbb{N} \cup \{\infty\}$ to each input.

But the connections with regular languages do not end there. Since the works of Rabin, Scott, and Büchi [10–12], there have not been many attempts to extend the expressive power of the associated formalisms beyond regular languages, while at the same time keeping most of their good properties – closure, decidability, and the equivalent presentations in terms of regular expressions, algebra, automata, and logic. To the best of our knowledge, the theory of regular cost functions is the only extension that achieves faithful extensions of classical results (in terms of closure, decidability, and

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

[†] Work partially supported by DFG grant BL 1127/2-1.

[‡] Work supported by the fellowship of the Foundation for Polish Science, during the author's post-doc stay at Université Paris Diderot.

CSL-LICS 2014, July 14–18, 2014, Vienna, Austria.
Copyright © 2014 ACM 978-1-4503-2886-9...$15.00.
http://dx.doi.org/10.1145/2603088.2603104

equivalent presentations) over finite words [9, 13], infinite words [14], and finite trees [15]. This unique position makes it an attractive framework, since it subsumes many classical results while allowing further questions about boundedness to be answered.

Central open question

The central open question in the theory is whether it can also be faithfully extended to infinite trees. In particular the question "can we solve satisfiability of cost monadic logic over infinite trees?" – the counterpart of the theorem of Rabin for monadic second-order logic – is still open.

This paper can be seen as a step to close this gap, by showing that there is a robust subclass of decidable cost functions over infinite trees that have natural correspondences with cost logics and cost automata.

Showing decidability of the full logic would help answer some fundamental questions about language and automata theory that have been open for decades. One of the motivating open problems like this is the (nondeterministic) *Mostowski index problem*, or parity index problem, which asks, given a regular language of infinite trees and a set of priorities P, is there a nondeterministic parity automaton that accepts this language using only priorities P? It turns out that this problem can be reduced to a boundedness question for regular cost functions over infinite trees [16]. Being able to solve this problem is useful, since the number of priorities in a parity automaton (even more so than the number of states) reflects how complicated the automaton is. Indeed, minimizing the number of priorities is especially important in verification, because model checking against a property described by a parity automaton is essentially exponential in the number of priorities.

Thus, our desire to understand and solve cost logics over infinite trees comes from a desire to close the gap in the theory of regular cost functions, as well as a desire to solve some long-standing and important problems in automata theory.

Weak and quasi-weak cost functions

In trying to solve the full logic, the weak fragment of cost monadic logic (where second-order quantifiers range over finite sets only) was a natural starting point for investigation. In [17], weak cost monadic logic was shown to be decidable, and equivalent to a form of weak alternating automata with counters.

However, weakness as defined for regular languages is not canonical in the context of cost functions. Indeed, a new, richer form of weakness emerged naturally in [18], in the form of *quasi-weak cost automata,* that enjoy many of the good properties of weak cost automata, but are strictly more expressive.

At the same time, there was a branch of work [5, 6] that sought to study the boundedness problem for fixed point formulas, and it turned out that the natural class of automata for problems like this was the quasi-weak class.

Recently in [19], it was also shown that the quasi-weak class of cost functions is strong enough to solve a special case of the *weak definability problem*. The weak definability problem asks, given a regular language of infinite trees, whether or not there is a weak automaton (equivalently, weak monadic second-order logic formula) that captures the same language. Like the parity index problem, the general version of this problem is open, and is of interest in verification because the weakly definable languages constitute an expressive but computationally feasible class of properties.

Thus, the quasi-weak class of cost functions emerged from a variety of sources as an interesting subclass of regular cost functions over infinite trees, with links to concrete decidable results and natural logical questions. However, it was not known whether this class corresponded to a natural extension of monadic second-order logic. This paper addresses this question, and develops further tools for understanding and working with cost logics over infinite trees.

Contributions

In this paper, we demonstrate that the class of quasi-weak cost functions corresponds to natural cost logics.

- We characterize the class of quasi-weak cost functions in terms of an extension of monadic second-order logic with a new form of fixed points that allows only bounded unfolding.
- We describe an extension of the μ-calculus with this bounded unfolding operator, and explain that the alternation-free fragment of this logic also characterizes quasi-weak cost functions.
- We show how to translate a 2-way quasi-weak automaton (in fact, any 2-way cost parity automaton) into an equivalent 1-way automaton. This is the technical core of our contribution, which is central to showing the equivalence of quasi-weak cost automata and logic. Our translation differs from other similar constructions in the context of regular languages in the sense that the translation does not rely on a global positional determinacy result in the underlying game, and instead uses only local approximations of the two-way plays. Although the proof is technical, we believe the ideas behind this construction may be of independent interest, even outside of the theory of regular cost functions.

Related work

This work builds on the classical results due to Rabin, Scott and Büchi [10–12] about regularity for languages of words and trees, both finite and infinite. It is also indebted to the work of Hashiguchi, Leung, Simon and Kirsten [1, 3, 20, 21] on limitedness questions (mainly in the context of solving the star height problem), and work by Bojańczyk and Colcombet [22, 23] on MSO+$\mathbb{U}$ (another logic related to boundedness).

Structure of this document

The article is organized as follows. In Section 2, we introduce cost monadic logic, and its weak and quasi-weak fragments. We then describe 2-way and 1-way alternating quasi-weak automata in Section 3. In Section 4, our main results concerning equivalence and decidability are given, and the essential ideas of the proof are sketched. Finally, in Section 5 we mention some links with a cost form of the μ-calculus that provide additional insight into the quasi-weak class of cost functions.

Notation and conventions

$\mathbb{A}$ will denote a fixed finite alphabet. The set of finite and infinite words over $\mathbb{A}$ is $\mathbb{A}^*$ and $\mathbb{A}^\omega$, respectively. The empty word is ϵ. For simplicity we work primarily with infinite binary trees. Let $\mathcal{T} = \{0,1\}^*$ be the unlabeled infinite binary tree. The set $\mathcal{T}_\mathbb{A}$ of *complete* $\mathbb{A}$*-labeled binary trees* consists of mappings $t : \mathcal{T} \to \mathbb{A}$. A *branch* π is a word $\{0,1\}^\omega$.

2. Quasi-weak cost logic

2.1 Cost functions

We write $\mathbb{N}$ for the set of non-negative integers and $\mathbb{N}_\infty$ for the set $\mathbb{N} \cup \{\infty\}$ with the obvious ordering. Non-decreasing functions $\mathbb{N} \to \mathbb{N}$ are called *correction functions.* We denote them by $\alpha, \beta, \ldots$. Any such function is implicitly extended to $\mathbb{N}_\infty$ by $\alpha(\infty) = \infty$.

Let E be a set and $\mathcal{F}_E$ the set of functions $E \to \mathbb{N}_\infty$. For $f, g \in \mathcal{F}_E$ and a correction function α, 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. Note that $f \preccurlyeq g$ holds if, and only if, f is bounded over every set $X \subseteq E$ over which g is bounded. The idea is that the *boundedness relation* $\approx$ does not pay attention to exact values, but does preserve the existence of bounds over all subsets of the domain. 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 $\mathcal{T}_{\mathbb{A}}$ and functions defined by logics and automata will always be considered as cost functions, *i.e.*, up to $\approx$.

We can identify a language L with its characteristic function χ_L mapping structures in the language to 0 and everything else to ∞. Note that for languages K and L, $K \subseteq L$ iff $\chi_L \preccurlyeq \chi_K$, so deciding cost function equivalence subsumes language inclusion testing in the classical setting.

Given two mathematical expressions $E(\bar{x}), F(\bar{x})$ denoting elements of $\mathbb{N}_\infty$ and depending on variables $\bar{x}$ (ranging over an implicit domain clear from the context), we write $E \preccurlyeq^{\bar{x}} F$ to specify that $\lambda\bar{x}.E \preccurlyeq \lambda\bar{x}.F$ where $\lambda\bar{x}.E$ denotes the function mapping $\bar{x}$ to $E(\bar{x}) \in \mathbb{N}_\infty$. The relation $\approx^{\bar{x}}$ is defined accordingly.

2.2 Cost monadic logic

Cost monadic second-order logic (CMSO) was introduced in [24] as a quantitative extension of monadic second-order logic. As usual, the logic can be defined over any relational structure, but we restrict our attention to CMSO over trees. In addition to first-order variables ranging over nodes of the tree and set variables ranging over sets of nodes, CMSO uses a single additional variable N, called the *bound variable,* which ranges over $\mathbb{N}$.

The atomic formulae in CMSO are those from MSO (the membership relation $x \in X$ and relations $a(x, x_1, x_2)$ asserting that $a \in \mathbb{A}$ is the label at position x with children x_1, x_2 from left to right), as well as a new predicate $|X| \leq N$ where X is any set variable and N is the bound variable. Arbitrary CMSO formulae are built inductively by applying boolean connectives and by quantifying (existentially or universally) over first-order or set variables. We require that any 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, the semantics of $|X| \leq N$ is what one would expect: the predicate holds iff the value of X has cardinality at most n. If, however, no value for N is specified then a sentence φ in cost monadic logic defines a function $[\![\varphi]\!] : \mathcal{T}_{\mathbb{A}} \to \mathbb{N}_\infty$ by $[\![\varphi]\!](t) := \inf\{n : t, n \models \varphi\}$, where we write $t, n \models \varphi$ if t satisfies φ when all occurrences of N take value n. Notice that in case φ is a pure MSO-sentence (not containing the predicates $|X| \leq N$), $[\![\varphi]\!](t)$ is 0, if t satisfies the sentence φ, and ∞ otherwise.

The weak variant of CMSO (denoted WCMSO) restricts the second-order quantification to finite sets, and has been studied in [17].

2.3 Bounded expansion and quasi-weak cost monadic logic.

In this paper, we introduce a new logic called *quasi-weak cost monadic logic*, QWCMSO, which extends WCMSO by a *bounded expansion* operator μ^N. This operator takes a function F mapping sets of nodes to sets of nodes which is *monotonic* (*i.e.*, $X \subseteq Y$ implies $F(X) \subseteq F(Y)$), and it computes $F^{(N)}$, where $F^{(0)} = \emptyset$ and $F^{(\ell+1)} = F(F^{(\ell)})$. We denote this value as $\mu^N X.F(X)$ in order to make the name of the variable over which the construction is performed explicit.

To add this operator to WCMSO, we define the following new *bounded expansion* construct

$$x \in \mu^N Z.\{y : \varphi(y, Z)\}\ ,$$

where x, y are first order variables, Z is a set variable and $\varphi(y, Z)$ is a formula that *uses Z positively*, *i.e.*, every predicate of the form $z \in Z$ appears below an even number of negations. This operator binds the variables y and Z, while it leaves x free. We also require that any such bounded expansion construct appears positively in the formula. The semantics are as one would expect: $\{y : \varphi(y, Z)\}$ is a set which depends on Z, and is thus subject to the application of $\mu^N Z$. The variable N used here is the same bound variable used in the predicates $|X| \leq N$. As before, the semantics associate to a sentence the least n which can be substituted for all occurrences of N and make the sentence true.

Example 1. Let $\mathbb{A} = \{a, b\}$. We use the operator μ^N to define a formula counting the maximal number of consecutive a's on a branch starting at the root, where $\mathtt{root}(w)$ identifies the root of the tree:

$$\begin{aligned}\exists w\big[&\mathtt{root}(w) \wedge \\ &w \in \mu^N X.\big\{x : \exists yz[b(x, y, z) \vee \\ &\qquad\qquad (a(x, y, z) \wedge y \in X \wedge z \in X)]\big\}\big]\ .\end{aligned}$$

This is equivalent (in the sense of the $\approx$ relation) to the CMSO formula

$$\forall X.[\mathtt{downclose}(X) \wedge ((\forall x \in X)a(x)) \to |X| \leq N]\ ,$$

where $\mathtt{downclose}(X)$ asserts that X is closed under the ancestor relation and $a(x)$ is shorthand for $\exists y, z.a(x, y, z)$.

3. Cost automata and games

In this section we introduce the model of automata used in this paper: the alternating 1-way/2-way B-quasi-weak automata. We consider classical parity automata over trees equipped with a finite set of counters Γ that can be *incremented* $\mathtt{ic}$, *reset* $\mathtt{r}$, or left *unchanged* ε (but whose values do no affect the flow of the automaton). Let $\mathbb{C} := \{\mathtt{ic}, \mathtt{r}, \varepsilon\}$ be the alphabet of counter actions. Each counter starts with value zero, and the value of a sequence of actions is the supremum of the values achieved during this sequence. For instance $(\mathtt{ic})(\mathtt{ic})\mathtt{r}\varepsilon(\mathtt{ic})\varepsilon$ has value 2, $((\mathtt{ic})\mathtt{r})^\omega$ has value 1, and $(\mathtt{ic})\mathtt{r}(\mathtt{ic})^2\mathtt{r}(\mathtt{ic})^3\mathtt{r}\ldots$ has value ∞. The set $Act_P^\Gamma := \mathbb{C}^\Gamma \times P$ collects the counter actions for a finite set Γ of counters and some finite set of priorities P. To an infinite sequence over Act_P^Γ, we assign the value ∞ if the maximum priority occurring infinitely often in it is odd (*i.e.*, if it does not satisfy the parity condition); otherwise, the value is the supremum of the values achieved by the counters (in case of several counters, we take the counter with the maximal value).

Formally, an *(alternating) two-way B-parity automaton* over the alphabet $\mathbb{A}$ is a tuple $\langle Q, \mathbb{A}, q_0, \Gamma, P, \delta\rangle$ consisting of a finite set of states Q, an initial state $q_0 \in Q$, a finite set Γ of counters, a finite set P of priorities, and a transition function

$$\delta : Q \times \mathbb{A} \to \mathcal{B}^+(\{\uparrow, \swarrow, \searrow, \circlearrowleft\} \times Act_P^\Gamma \times Q)$$

mapping a state and a letter to a positive boolean combination of triples of the form (d, c, q). Such a triple encodes the instruction to send the automaton to state q in direction d while performing action c. The directions $\swarrow$ and $\searrow$ move to the left or right child, $\uparrow$ moves to the parent, and $\circlearrowleft$ stays in place. We assume that $\delta(q, a)$ is written in disjunctive normal form for all q and a. Without loss of generality, we assume that the automaton never proceeds in direction $\uparrow$ from the root of the tree.

Acceptance of an input tree t by a B-automaton $\mathcal{A}$ is defined in terms of a game $(\mathcal{A}, t)$ between two players: Eve is in charge of the disjunctive choices. She tries to minimize counter values and to satisfy the parity condition. Adam, on the other hand, is in charge of the conjunctive choices and tries to maximize counter values or to sabotage the parity condition. As the transition function is given in disjunctive normal form, each turn of the game consists of Eve choosing a disjunct and Adam then selecting a single tuple (d, c, q) from it. We assume that each disjunction is nonempty, and each disjunct contains a tuple with direction other than $\uparrow$; in other words, from every position there is some move (i.e., the automaton cannot get stuck at the root).

A *play* of $\mathcal{A}$ on the tree t is a sequence

$$q_0, (d_1, c_1, q_1), (d_2, c_2, q_2), \ldots$$

compatible with t and δ, *i.e.,* q_0 is initial, and for all $i \in \mathbb{N}$, $(d_{i+1}, c_{i+1}, q_{i+1})$ appears in $\delta(q_i, t(x_i))$ where x_i is the node of t after following the directions $d_1 d_2 \ldots d_i$ starting from the root. The value $val(\pi)$ of a play π is the value of the sequence $c_1 c_2 \ldots$ as defined above. We will say that π is *n-winning* (for Eve) if $val(\pi) \leq n$.

A *strategy* for one of the players in the game $(\mathcal{A}, t)$ is a function that returns the next choice given the history of the play. If this function depends only on the current position in the game (rather than the full history), then it is *positional*. Note that choosing a strategy for Eve and a strategy for Adam fixes a play in $(\mathcal{A}, t)$. We say that a play π is *compatible* with a strategy σ if there is some strategy σ' for the other player such that σ and σ' together yield the play π. A strategy for Eve is *n-winning* if every play compatible with it is n-winning. We say that Eve *n-wins the game* if there is some n-winning strategy for Eve. An automaton *n-accepts* a tree t if Eve n-wins the game $(\mathcal{A}, t)$. We denote by $[\![\mathcal{A}]\!] : \mathcal{T}_{\mathbb{A}} \to \mathbb{N}_\infty$ the function given by $[\![\mathcal{A}]\!](t) := \inf \{n : \mathcal{A}\ n\text{-accepts } t\}$.

We will sometimes use automata that start from some position other than the root. We will call such automata *localized*. For localized automata, we use the notation $[\![\mathcal{A}]\!]_v(t)$ to specify the node v the automaton starts at. We can also consider cost automata with other well-known acceptance conditions. For instance, a B-Büchi automaton is a B-parity automaton using priorities $\{1, 2\}$. In this case, we often assume that priorities label states rather than edges (always possible by adding intermediate states), and refer to *Büchi states* (priority 2) and *non-Büchi states* (priority 1).

If every $\delta(q, a)$ uses only directions $\swarrow$ and $\searrow$, then we call $\mathcal{A}$ *one-way*. Moreover, if every $\delta(q, a)$ is of the form

$$\bigvee_i (\swarrow, c_i, q_i) \wedge (\searrow, c'_i, q'_i),$$

$\mathcal{A}$ is *nondeterministic*. For such a one-way nondeterministic automaton, 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 it is accepting on all branches, and the value assigned to a run of a B-automaton is the supremum of the values across all branches. For nondeterministic automata, the choices of Eve and Adam in the game described above can be viewed as Eve picking a transition $(\swarrow, c_i, q_i) \wedge (\searrow, c'_i, q'_i)$, and Adam choosing a direction (which uniquely determines which atom Adam picks from the conjunction). Unless otherwise indicated, we assume automata to be alternating.

Variants of weakness for cost automata

Weak alternating automata are Büchi automata with the restriction that no cycle of the automaton visits both Büchi and non-Büchi states (this is equivalent to the original definition in [25]). We consider two variants of this classical notion of weakness for cost automata. An alternating B-Büchi automaton is called

B-weak if in all cycles, either all states are Büchi, or no state is Büchi,

B-quasi-weak if in all cycles that contain both a Büchi and a non-Büchi state, there is a counter that is incremented and never reset in this cycle.

The weakness property implies that every play in the game associated with this automaton has to eventually stabilize, either in a strongly connected component where all states are Büchi (so the play is winning for Eve), or in a strongly connected component where no state is Büchi (so the play is winning for Adam). In fact, this stabilization occurs after at most $|Q|$-many *changes of mode* between Büchi states and non-Büchi states (where Q is the set of states of the automaton). These automata were studied in [17], and have the same expressive power as WCMSO.

Similarly, the quasi-weakness property implies that any play that does not stabilize after kn-changes of mode (for some constant k depending on the automaton) has a counter with value greater than n. Such plays cannot be n-winning for Eve, independent of any other consideration. Quasi-weak automata were introduced in [18]. They are strictly more expressive than weak automata (*i.e.,* WCMSO), but not as expressive as general cost automata (and in particular not as expressive as full CMSO).

Thus, the difference between these models is in the number of allowed mode changes: unrestricted for B-Büchi automata; bounded by some function of n for B-quasi-weak automata; and bounded by some constant for B-weak automata.

4. Equivalences

4.1 The main theorem and the general approach

Our goal is to show the equivalence of the various models of quasi-weak logics and automata we have introduced, as stated by the following main theorem.

Theorem 2. *Let f be a cost function over infinite trees. Then the following statements are equivalent:*

- *f is recognizable by a 1-way B-quasi-weak automaton;*
- *f is recognizable by a 2-way B-quasi-weak automaton;*
- *f is definable in QWCMSO.*

Moreover, the translations between these formalisms are effective. We call such a cost function f a quasi-weak cost function.

Since $f \preccurlyeq g$ is decidable for functions f, g defined by 1-way B-quasi-weak automata [18], we obtain the following decidability result as a corollary.

Corollary 3. *Given quasi-weak cost functions f and g over infinite trees, it is decidable whether or not $f \preccurlyeq g$.*

We now describe our approach for proving Theorem 2. The translation from B-quasi-weak automata to the logic is standard, so we concentrate on the translation from QWCMSO to B-quasi-weak automata, which goes via 2-way automata.

We use a well-known technique to enable automata to refer to free set variables. Given $t \in \mathcal{T}_{\mathbb{A}}$ and sets of nodes $E_1, \ldots, E_k$, we write $(t, E_1, \ldots, E_k)$ for the tree over the alphabet $\mathbb{A} \times \{0, 1\}^k$ obtained from t by labeling each node v by k bits of extra information, the i-th bit being 1 if $v \in E_i$ and 0 otherwise. First-order variables are treated as singleton sets. Using this encoding, every statement relating automata to logic can be used in a context with free variables, so we can consider cost automata $\mathcal{A}$ that correspond to formulae with free set variables $\bar{X}$. Given values $\bar{E}$ for these variables, $[\![\mathcal{A}]\!](t, \bar{E})$ denotes the evaluation of the automaton on the tree $(t, \bar{E})$.

As usual, translating formulae to automata amounts to showing the closure of the automaton model under operations that simulate the constructions of the logic. For simulating disjunction and conjunction, we must show closure under taking the minimum and maximum (respectively) of the functions computed by B-quasi-weak automata. This is obvious for alternating automata. B-quasi-weak automata must also be provided for the atomic predicates. All of this is standard.

There are three non-trivial constructs: weak existential quantifiers, weak universal quantifiers, and bounded expansions. Essentially, by adapting the proof from [17], one can show that 1-way B-quasi-weak automata have the closure properties corresponding to the weak quantifiers (it corresponds to the closure under weak $\inf$- and $\sup$-projection of 1-way B-weak automata in [17]).

The natural way to deal with bounded expansion is to perform it on localized 2-way B-quasi-weak automata, so it remains to show how to (i) translate a 1-way B-quasi-weak automaton into a localized 2-way B-quasi-weak automaton (Section 4.3), (ii) construct from this a localized 2-way B-quasi-weak automaton for the bounded expansion (Section 4.2), and (iii) translate a (localized) 2-way B-quasi-weak automaton into a 1-way B-quasi-weak automaton (Section 4.4).

4.2 The bounded expansion operation on automata

The bounded expansion operator is applied to a formula $\varphi(x, Y)$ which is syntactically monotonic in Y. Testing if

$$z \in \mu^N Y.\{x \,:\, \varphi(x, Y)\}$$

can be viewed as a game which starts in node $x = z$ with a given value n for the number of iterations and that proceeds in turns as follows:

- Eve chooses some set Y such that $\varphi(x, Y)$ holds (if it is not possible, she loses), then
- Adam chooses some element $x \in Y$ (if it is not possible, he loses), and the game proceeds to the next turn.

If the game exceeds n turns, Adam is declared the winner. It is straightforward to check that Eve wins this game if and only if $z \in \mu^N Y.\{x \,:\, \varphi(x, Y)\}$ for $N \leq n$.

We implement this idea by taking an automaton $\mathcal{A}$ equivalent to $\varphi(x, Y)$, and transforming it into a new automaton that simulates the game. For $\mathcal{A}$ we take a localized 2-way B-quasi-weak automaton with one free variable Y. We assume priorities label states. This automaton n-accepts a tree (t, Y) starting from position x if $t, n \models \varphi(x, Y)$ (in fact, modulo $\approx$).

The construction outputs a new localized 2-way B-quasi-weak automaton $\mathcal{B}$ with no free variables, which n-accepts a tree t from position x iff Eve wins the above game from initial position x. This new automaton $\mathcal{B}$ has the same states, the same initial state, the same priority function, and the same counters as $\mathcal{A}$, along with one additional counter, say, γ. The transition function $\delta_{\mathcal{B}}$ is defined for all states p and all letters a as:

$$\delta_{\mathcal{B}}(p, a) = \delta_{\mathcal{A}}(p, (a, 0)) \vee [\delta_{\mathcal{A}}(p, (a, 1)) \wedge (\circlearrowleft, \mathtt{I}_\gamma, q_0)] \,,$$

where $\mathtt{I}_\gamma$ resets all counters of $\mathcal{A}$ and increments the counter γ. This transition function expresses that Eve is required to say whether she uses the assumption that the current a-labelled position, say y, belongs to the set Y. If she assumes it does not, she can take the transition allowed in $\mathcal{A}$ for nodes not in Y (the first disjunct in the transition function above). If she assumes $y \in Y$ (the second disjunct), Adam and Eve can continue to play in the current round according to $\delta_{\mathcal{A}}(p, (a, 1))$ (intuitively, this corresponds to Adam checking that $t, n \models \varphi(x, Y)$ truly holds). Otherwise, Adam can ask to use y as his choice in the game by using the transition $(\circlearrowleft, \mathtt{I}_\gamma, q_0)$. This advances one turn in the game, so the new counter is incremented, all counters of $\mathcal{A}$ are reset, and the automaton $\mathcal{A}$ is restarted in state q_0 at y.

Theorem 4. *For every localized 2-way quasi-weak automaton $\mathcal{A}$ with one free set variable, there exists a localized 2-way quasi-weak automaton $\mathcal{B}$ with no free variables such that*

$$[\![\mathcal{B}]\!]_z(t) \approx^{t,z} [\![z \in \mu^N X.\{x \,:\, [\![\mathcal{A}]\!]_x(X) \leq N\}]\!](t) \,.$$

4.3 From 1-way to localized 2-way

Translating a 1-way automaton into an equivalent 2-way automaton is straightforward. The subtlety here is that we need to transform a 1-way automaton that has a first-order variable x as input, *i.e.*, reading the word $(t, \{x\})$ into a localized 2-way automaton that starts from node x, but reads only t.

Theorem 5. *For every 1-way B-quasi-weak automaton $\mathcal{A}$ with one free first-order variable, there exists a localized 2-way B-quasi-weak automaton $\mathcal{A}^\ell$ with no free variables such that*

$$[\![\mathcal{A}]\!](t, \{x\}) \approx^{t,x} [\![\mathcal{A}^\ell]\!]_x(t) \,.$$

The difficulty here is that the automaton $\mathcal{A}$ over the input $(t, \{x\})$ may cross the node x several times (on different plays), and make use of the fact that x is labeled. This is problematic since a localized automaton cannot remember this position, so the automaton would not know if/when it returns to x.

This would not be a problem if $\mathcal{A}$ were nondeterministic. Unfortunately, $\mathcal{A}$ is alternating, and making it nondeterministic would leave the class of quasi-weak automata (this situation occurs already for weakly definable languages). There is a known solution to this problem (due to Muller, Saoudi, and Schupp [25]), which is to make the automaton $\mathcal{A}$ nondeterministic, but only on a finite portion of the tree containing both the root and x. We are able to accomplish this using machinery already present in the proof for the closure of 1-way cost automata under the weak existential quantifier.

4.4 From 2-way to 1-way

We now turn to the key technical contribution of the paper: transforming 2-way B-quasi-weak automata into their 1-way version. We remark that it is trivial to transform a localized 2-way B-quasi-weak automaton into an equivalent non-localized one, so the localized property is irrelevant here.

Before proceeding, let us briefly review the construction from Vardi [26] that transforms a 2-way parity automaton $\mathcal{A}$ into an equivalent 1-way parity automaton $\mathcal{B}$ (this result can also be deduced from other constructions like the unfolding or iteration). The behavior of $\mathcal{A}$ on t can be represented as a parity game. By positional determinacy of parity games, Eve has a positional winning strategy if $\mathcal{A}$ accepts t. Moreover, for any position $x \in \mathcal{T}$, loops (*i.e.*, finite paths from x to x) in this strategy can be summarized by their starting state, ending state, and maximum priority. The 1-way version guesses a labeling of t with a positional strategy together with these loop summaries, and then runs a 1-way deterministic parity automaton that checks that the labeling is valid and that every play consistent with the strategy satisfies the parity condition. The loop summaries are used to avoid backtracking.

The above approach fails in our case for two reasons. First, there is no known result of positional or finite memory determinacy for the games produced by 2-way B-quasi-weak automata (the results are known for acyclic arenas like those produced by 1-way B-quasi-weak automata [18], but the arenas produced by 2-way automata may be cyclic). Second, the construction described above naturally outputs a nondeterministic automaton, and hence the output automaton cannot be quasi-weak (since quasi-weak and weak nondeterministic automata are strictly less expressive than their alternating versions). Thus, we have to use a less direct approach to prove the following theorem.

Theorem 6. *Given an alternating 2-way B-parity automaton $\mathcal{A}_2$, there effectively exists an alternating 1-way B-parity automaton $\mathcal{A}_1$ such that*

$$[\![\mathcal{A}_2]\!](t) \approx^t [\![\mathcal{A}_1]\!](t)\,.$$

Moreover, if $\mathcal{A}_2$ is quasi-weak, then $\mathcal{A}_1$ is also quasi-weak.

The proof consists of four steps.

(1) We first describe a way to summarize the history of a play of $\mathcal{A}_2$. The idea is that the summary of a play collapses loops (finite paths that start and end at the same position in the tree) in this play to a single move described by a global action. This global action records the maximum priority, and (for each counter) whether the counter was reset at least once, incremented at least once but not reset, or left unchanged.

We then prove that Eve always has a *summary-dependent strategy*, *i.e.*, a strategy where Eve's choices depend only on the summary of the history of the play rather than the full history of the play.

Lemma 7. *Assume that $\mathcal{A}_2$ n-accepts a tree t for some $n \in \mathbb{N}$. Then Eve has an $\alpha(n)$-winning summary-dependent strategy, for α independent of t.*

This summary-dependent strategy is constructed from an arbitrary n-winning strategy for Eve. In case there are several loops in the original strategy with the same summary, the summary-dependent strategy chooses the loop in which the counters had the greatest value, and then continues from there. This ensures that the counter values in the resulting summary-dependent strategy will not grow too much (and in fact are bounded by $\alpha(n)$ for some correction function α independent of the input tree). The parity condition will also be satisfied, since replacing loops by other loops having the same maximal priority, does not change whether or not the play is winning. We give the details of this proof in the next subsection.

(2) We construct a new alternating 2-way automaton $\mathcal{A}_{1.5}$ which never goes up (but may stay in the same node). It is not a B-parity automaton because it uses a more complicated acceptance condition, described below.

In general, the automaton $\mathcal{A}_{1.5}$ simulates the run of $\mathcal{A}_2$ on some t. However, when $\mathcal{A}_2$ wants to go down from some node x, Eve has to make some declarations about the two parts of the rest of the game: the part of the game before returning to x (*i.e.*, the part in the subtree of t below x), and the part after returning to x (which may visit the children of x again). For the first part she gives a set D of *constraints*, that is of pairs (c, q) that assert that there is a play compatible with her strategy which returns to x, finishing in state q and performing global action c. To describe the second part, Eve declares another set $C_{c,q}$ of constraints for each pair $(c, q) \in D$. This set $C_{c,q}$ describes the different ways her plays could go up to the parent of x if she returns to x in state q after performing global action c. In other words, Eve is declaring relevant information about loops and upward moves, to help us simulate the operation of $\mathcal{A}_2$ without actually moving upwards in t.

Indeed, after Eve makes these declarations, Adam can decide to verify either the part after a loop (in which case he stays in the same node) or he can decide to verify the assumptions about a loop (in which case he moves down in the tree). Later, when $\mathcal{A}_2$ wants to go up, we just check whether the current summary and state are in the previously declared set of constraints, and immediately win or lose in $\mathcal{A}_{1.5}$ based on this.

What is the winning condition of $\mathcal{A}_{1.5}$? Whenever $\mathcal{A}_{1.5}$ directly simulates $\mathcal{A}_2$, we output the same actions. When $\mathcal{A}_{1.5}$ follows some declared loop with global action c, we just output c. The sequence of actions output in this way should be bounded by a number n. But this is not enough, since it does not bound the values on paths of $\mathcal{A}_2$ which go up in the tree. To deal with such paths, whenever $\mathcal{A}_{1.5}$ moves down, we also output Eve's declarations after coming back from a loop (based on the sets $C_{c,q}$ for each $(c, q) \in D$ described above). We can view these declarations as a graph. The graph connects the pairs responsible for returning to a node x with the pairs describing moves up to the parent of x; an edge in the graph is labeled by the global action of such a path going up. At each moment when we go down in $\mathcal{A}_{1.5}$ we output a slice of such a graph, and the winning condition requires that in the whole graph constructed from such slices, each path should have value bounded by n. As mentioned earlier, this ensures that the value coming from upward paths in the plays consistent with Eve's strategy is also bounded.

Notice that in this construction, Eve must make the same declarations after coming up from two loops that have the same summary. Thus, to show that $\mathcal{A}_2$ and $\mathcal{A}_{1.5}$ are equivalent, it is important that Eve has a summary-dependent strategy in $\mathcal{A}_2$, as obtained in point (1).

(3) To replace our winning condition by a B-parity condition, we form a product of $\mathcal{A}_{1.5}$ with an automaton recognizing the winning condition of $\mathcal{A}_{1.5}$. Usually, such a construction would use a deterministic automaton. Unfortunately, cost automata cannot always be determinized [24] so this is not possible. However, every cost automaton can be made "history-deterministic", which is enough to ensure that it can be run on every branch of the game tree of $\mathcal{A}_{1.5}$ without causing conflicts. We refer the interested reader to [24].

(4) Finally, we eliminate from $\mathcal{A}_{1.5}$ moves using the $\circlearrowleft$ direction. For each state q, letter a, and goal set G of downward moves, we consider a *local game* $\mathcal{G}(q, a, G)$ describing the local ($\circlearrowleft$ direction) moves that are possible before moving downwards. Downward moves are terminal positions in the game: they are winning if they are in the goal set G, and losing otherwise. It can be shown that it is decidable whether Eve wins such a game. The idea is to transform this B-parity game into a game with a Streett winning condition, and solve the resulting Streett game using standard methods (see [27] for more details). The desired 1-way automaton $\mathcal{A}_1$ has the same input alphabet, set of states, set of counters, and initial state as $\mathcal{A}_{1.5}$. For a state q and input letter a, its transition function is defined as follows: Eve chooses a goal set G such that she wins in $\mathcal{G}(q, a, G)$; then Adam chooses any transition $(d, c, q') \in G$ and performs it.

Combining steps (1)–(4), we obtain an alternating 1-way B-parity automaton $\mathcal{A}_1$ that is equivalent to the original alternating 2-way B-parity automaton $\mathcal{A}_2$ (up to $\approx$). A closer examination of the construction shows that quasi-weakness is preserved.

4.5 Summary-dependent strategies (Proof of Lemma 7)

As mentioned earlier, it is an open problem whether in each B-parity game that is n-winning for Eve, she has a finite memory strategy that is $\alpha(n)$-winning, for some correction function α that is independent of the size of the game. In this section, we seek to prove a weaker property, saying that there exists a strategy which depends only on a summary of the play (Lemma 7).

Let $\mathcal{A}_2 = \langle Q, \mathbb{A}, q_0, \Gamma, \delta \rangle$ be a 2-way B-parity automaton. The transition function δ has type

$$Q \times \mathbb{A} \to \mathcal{B}^+(\{\uparrow, \swarrow, \searrow, \circlearrowleft\} \times Act \times Q)\,.$$

The set of actions Act gathers counter actions and parity ranks: $Act = \mathbb{C} \times \{i, i+1, \ldots, j\}$, where $\mathbb{C} = \{\mathtt{ic}, \varepsilon, \mathtt{r}\}^\Gamma$ and $i \leq j$ are natural numbers. For simplicity, we will assume that the automaton $\mathcal{A}_2$ satisfies the following conditions.

- We assume that the counter actions are *hierarchical*, which means that when some counter is incremented or reset, then simultaneously all counters with higher numbers are reset, *i.e.,* every action is of the form
$$(\varepsilon, \dots, \varepsilon, \mathtt{r}, \dots, \mathtt{r}) \quad \text{or} \quad (\varepsilon, \dots, \varepsilon, \mathtt{ic}, \mathtt{r}, \dots, \mathtt{r}).$$
It is known that every B-parity automaton can be converted into an equivalent one using only hierarchical actions [24].
- We assume that our alphabet is a product $\mathbb{A} = \mathbb{A}' \times \{0, 1\}$, and that each input tree t has the root marked by 1 in the second coordinate; additionally we assume that $\mathcal{A}_2$ never tries to go in the $\uparrow$ direction from the root of the tree. This assumption does not decrease the generality of the result.
- We assume that the states in $\mathcal{A}_2$ can be partitioned as follows.
    - *d-states* $q \in Q_d$ for $d \in \{\uparrow, \swarrow, \searrow\}$ are such that for all $a \in \mathbb{A}$, $\delta(q, a)$ is a single transition performing action ε in direction d.
    - *Universal states* $q \in Q_\wedge$ are such that for all a, $\delta(q, a)$ is a conjunction of transitions staying in the same node (direction $\circlearrowleft$).
    - *Existential states* $q \in Q_\vee$ are such that for all a, $\delta(q, a)$ is a disjunction of transitions staying in the same node (direction $\circlearrowleft$).

  Moreover, we assume that in the game $(\mathcal{A}_2, t)$ (for each tree t), there is some move possible from each position, and between each pair of positions there is at most one move. It is always possible to achieve this normal form, by using additional intermediate states.

We are now ready to analyze the operation of $\mathcal{A}_2$. The set of actions Act in the transition function is naturally equipped with a product operation, describing the *global action* of a sequence of actions. This product is defined component-wise: on the part from $\{i, i+1, \dots, j\}$, it corresponds to the maximum of the priorities, and on the part from $\{\mathtt{ic}, \varepsilon, \mathtt{r}\}$, it corresponds to the maximum according to the order $\varepsilon \le \mathtt{ic} \le \mathtt{r}$. Therefore, if c and c' are actions in Act, we can talk about the resulting action cc'. We use the symbol ε also to denote the neutral element of this product, that is $((\varepsilon, \dots, \varepsilon), i)$. We define the *value* of a finite sequence of counters as the maximum value of any counter in any moment while executing this sequence of actions. Recall that the value of an infinite sequence of actions takes into account also the parity condition. When we have a path whose edges are labeled by actions, we define the global action or the value of this path as the global action or the value of the sequence of actions on this path.

Notice that the global action contains all pertinent information about priorities, but loses some information about counter values, namely the value of a path, since for instance $\mathtt{ic}^n$ is contracted to $\mathtt{ic}$. This means that we will have to be able to retrieve this information in some way when doing such a contraction.

We must now define a *summary* of a play. Formally, let
$$q_0, (d_1, c_1, q_1), (d_2, c_2, q_2), \dots, (d_m, c_m, q_m)$$
be an initial fragment of a play in the game $(\mathcal{A}_2, t)$. Let x_i be the node in the tree after following the directions $d_1 d_2 \dots d_i$ starting from the root. The summary starts from q_0. Then for $i = 1, 2, \dots, m$ we proceed as follows. If x_{i-1} is such that all x_j for $j \ge i$ are proper descendants of x_{i-1}, then we simply append (d_i, c_i, q_i) to the summary. Otherwise, let $j \ge i$ be the smallest index for which $x_j = x_{i-1}$. Then to the summary we append $(\curvearrowright, c_i c_{i+1} \dots c_j, q_j)$ (on the second coordinate we have the product of the actions), and we continue generating the summary from $i := j+1$. For example if $d_1 \dots d_7 = \swarrow\swarrow\swarrow\uparrow\uparrow\circlearrowleft\swarrow$, the summary is
$$q_0, (\swarrow, c_1, q_1), (\curvearrowright, c_2 c_3 c_4 c_5, q_5), (\circlearrowleft, c_6, q_6), (\swarrow, c_7, q_7).$$
Notice that in a summary we never use the $\uparrow$ direction.

A strategy (of Eve) is called *summary-dependent* if the choices of Eve depend only on the summary of the history of the play. We will show that it is enough to consider summary-dependent strategies.

We are now ready to prove the following strengthening of Lemma 7:

> Assume that $\mathcal{A}_2$ n-accepts a tree t for some $n \in \mathbb{N}$. Then Eve has an $(n+1)^k$-winning summary-dependent strategy, where k is the number of hierarchical counters in $\mathcal{A}_2$.

Proof. Let t be an $\mathbb{A}$-labeled binary tree, and σ an n-winning strategy in the game $(\mathcal{A}_2, t)$. Let t_σ be the strategy tree describing all plays compatible with σ. Each node of t_σ is labeled by a position in $(\mathcal{A}_2, t)$, which is a pair: a node of t and a state. Recall that, due to our normalization of $\mathcal{A}_2$, two consecutive positions of the game determine the action of the move between these positions. To each node x of t_σ we assign a tuple $v_\sigma(x) = (n_1, \dots, n_{|\Gamma|})$ of values of the counters after performing the actions on the path from the root of t_σ to x. We will be comparing such tuples lexicographically. For a summary $\eta = q_0, (d_1, c_1, q_1), \dots, (d_m, c_m, q_m)$ we define $node(\eta)$ to be the node of t reachable by following the directions $d_1 d_2 \dots d_i$ starting from the root (where $\curvearrowright$ corresponds to staying in the same node).

First, some summaries $\eta = q_0, (d_1, c_1, q_1), \dots, (d_m, c_m, q_m)$ are assigned a node $g(\eta)$ of t_σ, labeled by $(node(\eta), q_m)$ This is done by induction on m. If $m = 0$, as $g(\eta)$ we just take the root of t_σ. Otherwise, let $\eta' := q_0, (d_1, c_1, q_1), \dots, (d_{m-1}, c_{m-1}, q_{m-1})$. If $g(\eta')$ is not defined, we also leave $g(\eta)$ undefined. Otherwise, assume first that $d_m \neq \curvearrowright$. If $g(\eta')$ has a child y labeled by $(node(\eta), q_m)$ (by our normalization assumption, there is at most one such child), we take $g(\eta) := y$. Otherwise, we leave $g(\eta)$ undefined. The other possibility is that $d_m = \curvearrowright$. Consider the set Y of all descendants y of $g(\eta')$ which are labeled by $(node(\eta), q_m)$, such that all nodes on the path from $g(\eta')$ to y, excluding $g(\eta')$ and y, are labeled by proper descendants of $node(\eta)$, and that the path from $g(\eta')$ to y has global action c_m. As $g(\eta)$ we take a node $y \in Y$ such that $v_\sigma(y) = \max_{z \in Y} v_\sigma(z)$, where the max-operator refers to the lexicographic order. If there are multiple nodes $y \in Y$ with maximal $v_\sigma(y)$, we can take any of them. If Y is empty, we leave $g(\eta)$ undefined. Observe that in both cases, $g(\eta)$ (if defined) is a descendant of $g(\eta')$, and the global action of the path from $g(\eta')$ to $g(\eta)$ is c_m.

Next, we construct a summary-dependent strategy ρ. When the history of the play has summary η, we simply look at the node $g(\eta)$, and we move in the same way as σ. Notice that $g(\eta)$ is labeled by our current position, so this move is legal. If $g(\eta)$ is undefined, we can move in any way; we will see below that when playing according to ρ we will never reach such situation.

It remains to see that ρ is winning. Take any play $\pi = q_0, (d_1, c_1, q_1), (d_2, c_2, q_2), \dots$ consistent with ρ. Let π_i be the prefix of π of length i, and let η_i be its summary. For each i, let y_i denote the child of $g(\eta_{i-1})$ such that the transition from $g(\eta_{i-1})$ to y_i is (d_i, c_i, q_i). Such a node exists (assuming that $g(\eta_{i-1})$ is defined): recall that the position after π_{i-1} is the same as in $g(\eta_{i-1})$; either it is a position of Eve and (d_i, c_i, q_i) is the move to the only child of $g(\eta_{i-1})$, or it is a position of Adam and we have children corresponding to all moves from this position.

Now, by induction on $i > 0$ we will show that $g(\eta_i)$ is defined, and that $v_\sigma(g(\eta_i)) \geq v_\sigma(y_i)$. Let

$$q'_0, (d'_1, c'_1, q'_1), \dots, (d'_m, c'_m, q'_m) := \eta_i \,,$$
$$\eta' := q'_0, (d'_1, c'_1, q'_1), \dots, (d'_{m-1}, c'_{m-1}, q'_{m-1}) \,.$$

We have two possibilities. First assume that $d'_m \neq \circlearrowleft$. Then we simply have $\eta_{i-1} = \eta'$, and we see that $g(\eta_i) = y_i$. The other possibility is that $d'_m = \circlearrowleft$. Then η' is a prefix of η_{i-1}, so $g(\eta_{i-1})$ (and y_i) is a descendant of $g(\eta')$, and the global action of path from $g(\eta')$ to y_i is c'_m. This means that y_i belongs to the set Y used to define $g(\eta_i)$. So $g(\eta_i)$ is defined, and $v_\sigma(g(\eta_i)) \geq v_\sigma(y_i)$.

Next, we observe the value of counter k. Consider k-tuples $v_k(x)$ which are defined as $v_\sigma(x)$, but contain only the values of the first k counters. Notice that $v_\sigma(x) \leq v_\sigma(y)$ implies $v_k(x) \leq v_k(y)$. If c_i increments counter k, then all counters with smaller numbers are not changed (recall that in $\mathcal{A}_2$ we only have hierarchical actions), so $v_k(g(\eta_{i-1})) < v_k(y_i) \leq v_k(g(\eta_i))$. And if the k-th counter is not changed by c_i, then all counters with smaller numbers are unchanged as well, so $v_k(g(\eta_{i-1})) = v_k(y_i) \leq v_k(g(\eta_i))$. We have only $(n+1)^k$ tuples with k values between 0 and n. This means that in π we cannot have a fragment where the k-th counter is incremented more than $(n+1)^k$ times and never reset. Thus the value of the counter is always at most $(n+1)^k$.

It remains to verify the parity condition. We can construct the summary also for the infinite play π; it is an infinite sequence $\eta = q'_0, (d'_1, c'_1, q'_1), (d'_2, c'_2, q'_2), \dots$. Consider the (unique) sequence $j_0, j_1, j_2, \dots$ such that η_{j_i} is the prefix of η of length i (in other words, j_i is the length of the prefix π that yields the summary η_{j_i} of length i). Notice that the global action of the fragment of π between the j_{i-1}-th and j_i-th step has global action c'_i, and also the path from $g(\eta_{j_{i-1}})$ to $g(\eta_{j_i})$ has global action c_i. It follows that the maximal priority appearing infinitely often in π is the same as in the branch of t_σ containing all $g(\eta_{j_i})$. All branches of t_σ are winning, so this priority is even. □

5. The mu-calculus view of quasi-weakness

In this section, we look at quasi-weakness from the point of view of the μ-calculus. This section is independent of the rest of the paper. We assume the reader to have some familiarity with the μ-calculus (see, *e.g.,* [28]).

It is well-known that μ-calculus is equivalent to alternating automata. The simplicity of the corresponding translations implies that many properties can be expressed equally well at the automaton level and at the logic level. For instance, the structure of the acceptance condition of the automaton is tightly related to the nesting structure of fixed points in μ-calculus formulae: least fixed points correspond to odd priorities, and greatest fixed points to even priorities. Moreover, the number of priorities in the automaton reflects the nesting of μ and ν operators in the formula. As a special case that we are particularly interested in, the *alternation free* fragment of μ-calculus (see below) corresponds exactly to weak alternating automata. We can extend some of these relationships to cost functions.

Let us recall the definition of the *μ-calculus*. We assume an infinite set of variables $X, Y, Z, \dots$ ranging over sets of nodes. The semantics of a μ-calculus formula with free variables $\bar{X}$ is given by a monotonic function from tuples of sets indexed by $\bar{X}$ to sets. The following constructs are allowed. There are *modal operators* that consist here of constant modalities a, for all letters a, and successor modalities $\mathtt{X}(E, F)$. A modality of the first kind, evaluates to the set of nodes carrying the letter a, while a modality of the second kind returns, given sets of nodes E and F, the set of nodes such that the left child is in E and the right child in F. We also have boolean operators $\vee$, $\wedge$, $\neg$ and fixed point operators. The *least fixed point* $\mu X.F(X, \bar{V})$ is the least set E such that $E = F(E, \bar{V})$ and the *greatest fixed point* $\nu X.F(X, \bar{V})$ is the greatest such set. Every μ-calculus formula with free variables $\bar{X}$ *evaluates* to a mapping from a tree t and a tuple of sets of nodes indexed by $\bar{X}$ to a set of nodes.

The *μ^N-calculus* extends μ-calculus in a way similar to CMSO, meaning that the semantics of a formula is parametrized by some non-negative integer n. So a μ^N-calculus formula *n-evaluates* to a function. One adds the new construct $\mu^N X.\psi(X)$ which evaluates to $F_n^{(n)}$, assuming that ψ n-evaluates to F_n for all n (where $F^{(n)}$ is defined as in Section 2.3).

Given a μ^N-calculus formula ψ without free variables, it defines a function $\mathcal{T}_{\mathbb{A}} \to \mathbb{N}_\infty$ by

$$[\![\psi]\!](t) = \inf\{n \ : \ \psi \ n\text{-evaluates to } E \text{ with } \epsilon \in E\} \ ,$$

where ϵ is the root of t.

Example 8. Let $\mathbb{A} := \{a, b\}$. The formula

$$\mu^N X.(b \vee (a \wedge \mathtt{X}(X, X))$$

computes the maximum number of consecutive a's that occur on some branch starting in the root (the same cost function as Example 1). The idea is that we can start evaluating at the root. If the current node is a-labelled, then both successors are forced to "loop" back to X, and continue as before. Because the fixed point operator corresponding to X is a μ^N, we count these unfoldings. The least number of unfoldings that are needed is exactly the maximum number of consecutive a's on some branch starting in the root.

Now consider the formula

$$\mu^N X.\nu Y.[(b \wedge \mathtt{X}(Y, Y)) \vee (a \wedge \mathtt{X}(X, X))]$$

which computes the maximum number of a's (not necessarily consecutive) that occur on some branch. As above, an a-labelled node forces a loop back on X for both successors, and these unfoldings are counted. However, when a b-labelled node is encountered the successors loop back to Y. This fixed point variable Y corresponds to a ν operator, so these unfoldings are not counted, and indeed can be taken infinitely many times. Because the ν operator is nested inside of μ^N, looping back on Y does not "reset" the count of the number of unfoldings we have used for X, so this formula computes the desired cost function.

Finally, the formula

$$\begin{aligned} \mu^N X.\nu Y.[(b \wedge (\mathtt{X}(Y, \top) \vee \mathtt{X}(\top, Y))) \vee \\ (a \wedge (\mathtt{X}(X, \top) \vee \mathtt{X}(\top, X)))] \end{aligned}$$

computes the minimal number of a's that occur on some branch ($\top$ is an abbreviation for $\nu Z.Z$). This is similar to the previous example, except that only a single branch is picked out during the evaluation of the formula (intuitively, the branch with the minimum number of a's).

So far, we have described μ^N-calculus in its most general form, which is equivalent to B-parity automata. However, we are interested in the quasi-weak form of such automata. The nice thing about μ-calculus is that, again, quasi-weakness has a natural interpretation. Renaming bound variables, we may ensure that no variable is used in two distinct fixed points (μ, ν or μ^N) in a formula. Then the scope of the variable is the set of nodes of the formula (seen as a tree) that are below the fixed-point operator binding the variable, and above some use of the variable. If the variable is introduced with a μ (resp. ν or μ^N), we call it a μ-scope (resp. a ν-scope or a μ^N-scope). A μ^N-calculus formula is

weak if no ν-scope intersects a μ-scope, and no μ^N-scope simultaneously intersects a μ-scope and a ν-scope,

quasi-weak if no ν-scope intersects a μ-scope.

Note that for μ-calculus formulae without μ^N operators, the two definitions coincide. The classical equivalence between automata and μ-calculus formulae carries over without surprise to cost functions (the proofs are essentially the same).

Theorem 9. *In terms of recognizing cost functions,*

- *B-parity automata are effectively equivalent to μ^N-formulae,*
- *B-weak automata are effectively equivalent to weak μ^N-formulae,*
- *B-quasi-weak automata are effectively equivalent to quasi-weak μ^N-formulae.*

What is interesting about these notions is that the definition of quasi-weakness is significantly simpler than the one of weakness. Moreover, the definition for a quasi-weak μ^N-formula is the standard definition of the alternation-free fragment of the μ-calculus. This supports the fact that, in the cost setting, quasi-weakness is the correct counterpart to the notion of weakness from the classical setting.

6. Conclusion

We have described some natural logics that correspond to the class of quasi-weak cost functions over infinite trees, a robust and natural class of cost functions. Along the way, we have developed some technical tools, like summary-dependent strategies and the conversion of 2-way cost automata to 1-way cost automata.

Overall, the hope is that these results and tools may eventually prove useful for solving full cost monadic logic over infinite trees, and for solving challenging boundedness questions like the parity index problem and weak definability problem.

References

[28] A. Arnold and D. Niwiński. *Rudiments of μ-Calculus*, volume 146 of *Studies in Logic and The Foundations of Computer Science*. North-Holland, 2001.

[7] M. Benedikt, G. Puppis, and C. Riveros. Regular repair of specifications. In *LICS*, pages 335–344. IEEE Computer Society, 2011. ISBN 978-0-7695-4412-0.

[6] A. Blumensath, M. Otto, and M. Weyer. Decidability Results for the Boundedness Problem. *Logical Methods in Computer Science*. to appear.

[5] A. Blumensath, M. Otto, and M. Weyer. Boundedness of Monadic Second-Order Formulae over Finite Words. In *Proc. 36th Int. Colloquium on Automata, Languages and Programming, ICALP, Part II, LNCS 5556*, pages 67–78, 2009.

[22] M. Bojańczyk. A bounding quantifier. In J. Marcinkowski and A. Tarlecki, editors, *CSL*, volume 3210 of *LNCS*, pages 41–55. Springer, 2004. ISBN 3-540-23024-6.

[23] M. Bojańczyk and T. Colcombet. Bounds in ω-regularity. In *LICS*, pages 285–296. IEEE Computer Society, 2006.

[11] J. R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92, 1960.

[24] T. Colcombet. Regular cost functions over words, 2009. Manuscript at `http://www.liafa.jussieu.fr/~colcombe/`.

[9] T. Colcombet. The theory of stabilisation monoids and regular cost functions. In *ICALP (2)*, volume 5556 of *LNCS*, pages 139–150. Springer, 2009. ISBN 978-3-642-02929-5.

[13] T. Colcombet. Regular cost functions, part i: Logic and algebra over words. *Logical Methods in Computer Science*, 9(3), 2013.

[4] T. Colcombet and C. Löding. The nesting-depth of disjunctive mu-calculus. In *CSL*, volume 5213 of *LNCS*, pages 416–430. Springer, 2008. ISBN 978-3-540-87530-7.

[16] T. Colcombet and C. Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In L. Aceto, I. Damgard, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP (2)*, volume 5126 of *LNCS*, pages 398–409. Springer, 2008. ISBN 978-3-540-70582-6.

[15] T. Colcombet and C. Löding. Regular cost functions over finite trees. In *LICS*, pages 70–79. IEEE Computer Society, 2010. ISBN 978-0-7695-4114-3. Online at `http://www.liafa.jussieu.fr/~colcombe/`.

[19] T. Colcombet, D. Kuperberg, C. Löding, and M. Vanden Boom. Deciding the weak definability of büchi definable tree languages. In S. R. D. Rocca, editor, *CSL*, volume 23 of *LIPIcs*, pages 215–230. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013. ISBN 978-3-939897-60-6.

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

[2] K. Hashiguchi. Relative star height, star height and finite automata with distance functions. In J.-É. Pin, editor, *Formal Properties of Finite Automata and Applications*, volume 386 of *LNCS*, pages 74–88. Springer, 1988. ISBN 3-540-51631-X.

[3] D. Kirsten. Distance desert automata and the star height problem. *ITA*, 39(3):455–509, 2005.

[18] D. Kuperberg and M. Vanden Boom. Quasi-weak cost automata: a new variant of weakness. In S. Chakraborty and A. Kumar, editors, *FSTTCS*, volume 13 of *LIPIcs*, pages 66–77. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011. ISBN 978-3-939897-34-7. Online at `http://www.liafa.jussieu.fr/~dkuperbe/`.

[14] D. Kuperberg and M. Vanden Boom. On the expressive power of cost logics over infinite words. In A. Czumaj, K. Mehlhorn, A. M. Pitts, and R. Wattenhofer, editors, *ICALP (2)*, volume 7392 of *Lecture Notes in Computer Science*, pages 287–298. Springer, 2012. ISBN 978-3-642-31584-8.

[8] M. Lang. Resource-bounded reachability on pushdown systems. Master's thesis, RWTH Aachen University, 2011.

[21] H. Leung. On the topological structure of a finitely generated semigroup of matrices. *Semigroup Forum*, 37:273–287, 1988.

[25] D. E. Muller, A. Saoudi, and P. E. Schupp. Alternating automata. The weak monadic theory of the tree, and its complexity. In L. Kott, editor, *ICALP*, volume 226 of *LNCS*, pages 275–283. Springer, 1986. ISBN 3-540-16761-7.

[12] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969. ISSN 0002-9947.

[10] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, Apr. 1959. ISSN 0018-8646. . URL `http://dx.doi.org/10.1147/rd.32.0114`.

[20] I. Simon. Limited subsets of a free monoid. In *FOCS*, pages 143–150. IEEE, 1978.

[27] W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 389–455. Springer, 1997.

[17] M. Vanden Boom. Weak cost monadic logic over infinite trees. In F. Murlak and P. Sankowski, editors, *MFCS*, volume 6907 of *LNCS*, pages 580–591. Springer, 2011. ISBN 978-3-642-22992-3.

[26] M. Y. Vardi. Reasoning about the past with two-way automata. In K. G. Larsen, S. Skyum, and G. Winskel, editors, *ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer, 1998. ISBN 3-540-64781-3.