The Bridge Between Regular Cost Functions and Omega-Regular Languages*

Thomas Colcombet[1] and Nathanaël Fijalkow[2]

1 **CNRS, IRIF, Université Paris Diderot, Paris, France**
thomas.colcombet@irif.univ-paris-diderot.fr
2 **University of Oxford, Oxford, United Kingdom**
nathanael.fijalkow@cs.ox.ac.uk

Abstract

In this paper, we exhibit a one-to-one correspondence between ω-regular languages and a subclass of regular cost functions over finite words, called ω-regular like cost functions. This bridge between the two models allows one to readily import classical results such as the last appearance record or the McNaughton-Safra constructions to the realm of regular cost functions. In combination with game theoretic techniques, this also yields a simple description of an optimal procedure of history-determinisation for cost automata, a central result in the theory of regular cost functions.

1998 ACM Subject Classification F 1.1 Models of Computations

Keywords and phrases Theory of Regular Cost Functions, Automata with Counters, Cost-automata, Quantitative Extensions of Automata, Determinisation of Automata

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.126

1 Introduction

The theory of regular cost functions [4] aims at offering a uniform framework dealing with boundedness questions in automata theory. It provides a toolbox of concepts and results for solving questions involving resource constraints, such as the star height problem over finite words [10, 12] and finite trees [7], the finite power property [14], the boundedness of fixpoints for monadic second-order logic [2] or over guarded logic [1], or for attacking the Mostowski index problem [7]. The strength of regular cost functions is that it is a quantitative setting where many of the crucial results of regular languages generalise, including the cornerstone effective equivalence between logic, automata, algebra and expressions.

For regular languages, determinising plays a central role, as for instance for complementing automata over infinite trees, or for solving games. The situation is different for cost functions, even over finite words: it is impossible to determinise cost automata, deterministic cost automata being strictly less expressive. The notion of history-deterministic automata overcomes this shortcoming. These are non-deterministic cost automata that have the semantical property that an oracle resolves the non-determinism in an optimal way. The non-determinisability issue is resolved by establishing that cost automata can be effectively transformed into history-deterministic ones [4]. This is crucially used for instance when

* This project has received funding from the European Research Council Seventh Framework Programme (FP7/2007-2013) under grant agreement 259454 (GALE) and the European Union's Horizon 2020 research and innovation programme under grant agreement 670624 (DuaLL). The second author gratefully acknowledges the support of the EPSRC grant EP/M012298/1.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 126; pp. 126:1–126:13


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

developing the theory of regular cost functions over finite trees [7]. However, the proof of this result is so far very complicated. The original version [4] was going through algebra (stabilisation monoids), incurring a double exponential blowup. An optimal version inspired by the construction of Safra is known [5], but its description and correctness proof are extremely complicated.

One aim of the present work is to give a simple description and correctness proof of the construction from [5] which, given a cost-automaton as input, produces an equivalent history-deterministic automaton. The key advantage of the novel presentation in this work is that it uses the determinisation of ω-regular languages as a *black box*. In particular, it does not depend at all on the details of the (relatively complicated) Safra construction. This also makes both the construction and the proof much simpler. Further, it is optimal, meaning that it yields an automaton of exponential size, matching known lower bound for the case of ω-regular languages [8].

In order to obtain this completely new presentation, we describe a one-to-one correspondence between the theory of ω-regular languages and a subclass of regular cost functions, called the ω-regular like cost functions. This correspondence allows us to readily import from ω-regular languages constructions such as the last appearance record or determinisation results to regular cost functions. In other words, ω-regular like cost functions are determinisable.

In a second step, combining game theoretic techniques with an idea of Bojańczyk [3], we obtain a simple, direct and optimal construction of history-deterministic cost automata.

Structure of the document

We define the class of ω-regular like cost functions in Section 3, and study its properties. We give in Section 4 the history-determinisation procedure, relying on the results about ω-regular like cost functions combined with game theoretic techniques.

2 Definitions

Let $\mathtt{A}$ and $\mathtt{B}$ be *alphabets*. An *initial automaton structure* is denoted $\mathcal{A} = (Q, \mathtt{A}, \mathtt{B}, I, \Delta)$, where $\mathtt{A}$ is the *input alphabet*, $\mathtt{B}$ is the *output alphabet*, Q is a finite *set of states*, $I \subseteq Q$ is the set of *initial states* and $\Delta \subseteq Q \times \mathtt{A} \times \mathtt{B} \times Q$ is the *transition relation*. An element $(p, a, b, q) \in \Delta$ is called a *transition*. An *automaton structure* $\mathcal{A} = (Q, \mathtt{A}, \mathtt{B}, I, \Delta, F)$ is an initial automaton structure enriched with a set $F \subseteq Q$ of accepting states.

A *run* (which can be finite or infinite) is a sequence of transitions of the form

$$(p_0, a_1, b_1, p_1)(p_1, a_2, b_2, p_2) \cdots$$

such that p_0 is initial. We denote by $\rho|_A$ its projection to the alphabet $\mathtt{A}$, and by $\rho|_B$ its projection to the alphabet $\mathtt{B}$. For w a finite or infinite word over $\mathtt{A}$, we say that a run ρ is a run of w if $\rho|_A = w$ and a prefix run of w if $\rho|_A$ is a prefix of w. When dealing with an automaton structure, we further require that a run of a finite word ends in an accepting state.

ω-automata

An *ω-automaton* is denoted $\mathcal{A} = (Q, \mathtt{A}, \mathtt{B}, I, \Delta, W)$, where $(Q, \mathtt{A}, \mathtt{B}, I, \Delta)$ is an initial automaton structure, and $W \subseteq \mathtt{B}^\omega$ is called the *accepting condition*. The ω-language recognised by the ω-automaton is the set

$$\{w \in \mathtt{A}^\omega \mid \text{there exists a run } \rho \text{ over } w \text{ such that } \rho|_B \in W\}.$$

We define some of the classical ω-accepting conditions.

$$\begin{aligned}
\texttt{Büchi} &= \{w \in \{0,1\}^\omega \mid w \text{ contains infinitely many } 0's\} \\
\texttt{coBüchi} &= \{w \in \{1,2\}^\omega \mid w \text{ contains finitely many } 1's\} \\
\texttt{Rabin}_1 &= \left\{ w \in \{I,R,\epsilon\}^\omega \;\middle|\; \begin{array}{l} w \text{ contains infinitely many } I's \\ \text{and finitely many } R's \end{array} \right\} \\
\texttt{Rabin}_k &= \left\{ w \in \left(\{I,R,\epsilon\}^k\right)^\omega \;\middle|\; \begin{array}{l} \text{for some } \ell \in \{1,\dots,k\}, \\ w \text{ contains infinitely many } I'_\ell s \\ \text{and finitely many } R'_\ell s \end{array} \right\} \\
\texttt{Parity}_k &= \left\{ w \in \{1,\dots,k\}^\omega \;\middle|\; \begin{array}{l} \text{the smallest colour appearing} \\ \text{infinitely often in } w \text{ is even} \end{array} \right\}
\end{aligned}$$

A *Rabin automaton* is an ω-automaton with a Rabin condition, and similarly for the other conditions. It is known that Büchi, parity and Rabin automata recognise the same ω-languages, that are called the *ω-regular languages*.

Regular cost functions

We consider functions from $\mathtt{A}^*$ to $\mathbb{N} \cup \{\infty\}$. Let f be such a function, and $X \subseteq \mathtt{A}^*$, we say that $f|_X$ is bounded if there exists $n \in \mathbb{N}$ such that $f(u) \leq n$ for all $u \in X$.

Let f, g be two such functions, then g *dominates* f, denoted $f \preccurlyeq g$, if for all $X \subseteq \mathtt{A}^*$, if $g|_X$ is bounded then $f|_X$ is bounded. We say that f and g are equivalent, denoted $f \approx g$, if $f \preccurlyeq g$ and $g \preccurlyeq f$. The following lemma is central, see [6] for more considerations on this equivalence relation.

▶ **Lemma 1.** *$f \preccurlyeq g$ if, and only if, $f \leq \alpha \circ g$ for some function $\alpha : \mathbb{N} \to \mathbb{N}$ such that $\lim \alpha = \infty$, extended with $\alpha(\infty) = \infty$.*

A *cost function* is an equivalence class for the relation $\approx$.

Many equivalent formalisms can be used in order to define regular cost functions; this paper studies automata.

▶ **Definition 2.** A *min-cost-automaton* is denoted $\mathcal{A} = (Q, \mathtt{A}, \mathtt{B}, I, \Delta, F, f)$, given by an automaton structure $(Q, \mathtt{A}, \mathtt{B}, I, \Delta, F)$ together with an *accepting map* $f : \mathtt{B}^* \to \mathbb{N} \cup \{\infty\}$. It recognises the cost function induced by the map

$$\begin{aligned}
[\![\mathcal{A}]\!]_{\min} : \mathtt{A}^* &\to \mathbb{N} \cup \{\infty\} \\
w &\mapsto \inf \{f(\rho|_B) \mid \rho \text{ run over } w\} \ .
\end{aligned}$$

A *max-cost-automaton* is defined similarly, recognising the cost function induced by the map

$$\begin{aligned}
[\![\mathcal{A}]\!]_{\max} : \mathtt{A}^* &\to \mathbb{N} \cup \{\infty\} \\
w &\mapsto \sup \{f(\rho|_B) \mid \rho \text{ run over } w\} \ .
\end{aligned}$$

We define some of the classical accepting maps for regular cost functions.

We first define the $\texttt{cost}_\mathtt{B}$ map for one counter. The value of the counter is initialised by 0. The letter `i` is an increment, it adds 1 to the value of the counter, the letter `r` is a

reset, it resets the value of the counter to 0, and the letter ϵ does nothing. Formally, the $\mathtt{cost}_{\mathtt{B}}$ map for one counter is defined by

$$\begin{aligned}\mathtt{cost}_{\mathtt{B}} : \{\mathtt{i},\mathtt{r},\epsilon\}^* &\to \mathbb{N}\cup\{\infty\}\\ w &\mapsto \max\{n\in\mathbb{N} \mid w\in\{\mathtt{i},\mathtt{r},\epsilon\}^*\,(\epsilon^*\mathtt{i})^n\,\{\mathtt{i},\mathtt{r},\epsilon\}^*\}\,.\end{aligned}$$

The restrictions over $\{\epsilon,\mathtt{i}\}^*$ and over $\{\mathtt{r},\mathtt{i}\}^*$ are called distance and desert, respectively denoted $\mathtt{dist}_{\mathtt{B}}$ and $\mathtt{desert}_{\mathtt{B}}$.

The $\mathtt{cost}_{\mathtt{B}}$ map for k counters is defined similarly as for one counter, over the alphabet $\{\epsilon,\mathtt{i},\mathtt{r}\}^k$, by taking the maximum over all counters.

The $\mathtt{cost}_{\mathtt{hB}}$ map for k hierarchical counters is the restriction of $\mathtt{cost}_{\mathtt{B}}$ over the alphabet $\{I_1,R_1,\ldots,I_k,R_k\}$, where I_ℓ increments the ℓ^{th} counter and resets all counters of smaller index, and R_ℓ resets all counters of index smaller than or equal to ℓ.

A *B-automaton* is a *min-cost-automaton* equipped with a $\mathtt{cost}_{\mathtt{B}}$ map. Similarly, a *hB-automaton* is equipped with a $\mathtt{cost}_{\mathtt{hB}}$ map. The class of cost functions recognised by B-automata (or equivalently, hB-automata) is called regular cost functions.

We will make use of the following special case of max-cost-automata.

▶ **Definition 3.** A *prefix-max-cost-automaton* is denoted $\mathcal{A}=(Q,\mathtt{A},\mathtt{B},I,\Delta,f)$, given by an initial automaton structure $(Q,\mathtt{A},\mathtt{B},I,\Delta)$ together with an *accepting map* $f:\mathtt{B}^*\to\mathbb{N}\cup\{\infty\}$. It recognises the cost function induced by the map

$$\begin{aligned}[\![\mathcal{A}]\!]_{\text{pmax}} : \mathtt{A}^* &\to \mathbb{N}\cup\{\infty\}\\ w &\mapsto \sup\{f(\rho|_B) \mid \rho \text{ prefix run over } w\}\ .\end{aligned}$$

3 Omega Regular like Cost Functions

In this section we introduce the subclass of regular cost functions called ω-regular like cost functions, that we show are in one-to-one correspondence with ω-regular languages.

In Subsection 3.1 we define an operator defining the class and fleshing out the correspondence. We then explain how to construct ω-regular like cost functions with different models: in Subsection 3.2 using automata, and in Subsection 3.3 using algebra.

This strong correspondence allows us to transfer results from ω-regular languages to ω-regular like cost functions; in Subsection 3.4 we show how to transfer the latest appearence record and the Safra constructions.

Finally, we show the interplay between ω-regular like cost functions and games in Subsection 3.5.

3.1 Bijection with Omega-Regular Languages

The following is the main definition of this paper.

▶ **Definition 4.** Given a language L over infinite words, $L^{\mathtt{ol}}$ is defined by

$$\begin{aligned}L^{\mathtt{ol}} : \mathtt{A}^* &\to \mathbb{N}\cup\{\infty\}\\ w &\mapsto \sup\{n \mid w = uv_1\cdots v_nu',\ v_1,\ldots,v_n\neq\varepsilon,\ u\cdot\{v_1,\ldots,v_n\}^\omega\subseteq L\}\,.\end{aligned}$$

A cost function is *ω-regular like* if it contains a map $L^{\mathtt{ol}}$ for some ω-regular language L.

Note that we will mostly be interested in using the definition of $\cdot^{\mathtt{ol}}$ with ω-regular languages.

▶ **Example 5.**

- $\texttt{Büchi}^{\texttt{ol}} = \texttt{dist}_{\texttt{B}}$: it is the function counting the number of 1's, *i.e.* the distance map where 1 is ϵ and 0 is $\texttt{i}$.
- $\texttt{coBüchi}^{\texttt{ol}} = \texttt{desert}_{\texttt{B}}$: it is the function counting the size of the largest block of 2's, *i.e.* the desert map where 1 is $\texttt{r}$ and 2 is $\texttt{i}$.
- $\texttt{Rabin}_1{}^{\texttt{ol}} = \texttt{cost}_{\texttt{B}}$: it is the function counting the number of I's in a block containing no R's, *i.e.* the $\texttt{cost}_{\texttt{B}}$ map for one counter where I is $\texttt{i}$ and R is $\texttt{r}$.
- $\texttt{Rabin}_k{}^{\texttt{ol}} \approx \texttt{cost}_{\texttt{B}}$: it is the $\texttt{cost}_{\texttt{B}}$ map for k counters where I_ℓ is increment for the ℓ^{th} counter and R_ℓ is reset for the ℓ^{th} counter. Note that here the functions are not equal, one can see that $\texttt{cost}_{\texttt{B}} \leq \texttt{Rabin}_k{}^{\texttt{ol}} \leq k \cdot \texttt{cost}_{\texttt{B}}$.
- $\texttt{Parity}_{2k}{}^{\texttt{ol}} \approx \texttt{cost}_{\texttt{hB}}$: it is the $\texttt{cost}_{\texttt{hB}}$ map for k counters, where I_ℓ is the colour $2(k-\ell)$ and R_ℓ is $2(k-\ell)-1$.

The following lemma is central, it shows the interplay between the above definition and ultimately periodic words.

▶ **Lemma 6.** *Let L be a language over infinite words, and u, v two finite words with v non-empty. The following statements are equivalent:*

1. $uv^\omega \in L$,
2. $(L^{\texttt{ol}}(uv^n))_{n\in\mathbb{N}}$ *tends to infinity.*

Note that this lemma does not make any assumption on the regularity of L; in the rest of the paper, we shall always look at $L^{\texttt{ol}}$ for L an ω-regular language.

Proof. One direction is clear: if $uv^\omega \in L$, then $(L^{\texttt{ol}}(uv^n))_{n\in\mathbb{N}}$ tends to infinity.

We prove the converse implication. Assume that $(L^{\texttt{ol}}(uv^n))_{n\in\mathbb{N}}$ tends to infinity, and let n be larger than $|uv|$. There exists k such that uv^k can be factorised $u'v_1\cdots v_n u''$ such that $u' \cdot \{v_1, \ldots, v_n\}^\omega \subseteq L$.

Consider the lengths $|u'v_1\cdots v_\ell|$ for $\ell \in \{|u|, \ldots, n\}$: two of them have the same value modulo $|v|$, denote the corresponding words $u'v_1\cdots v_i$ and $u'v_1\cdots v_j$, with $i < j$. Note that since $\ell \geq |u|$ and $v_1, \ldots, v_\ell$ are not empty, the word u is a strict prefix of $u'v_1\cdots v_i$. It follows that we have $u'v_1\cdots v_i = uv^p x$ for some p and $v = xy$, and $v_{i+1}\cdots v_j = yv^q x$ for some q.

Consider the infinite word

$$u'v_1\cdots v_i(v_{i+1}\cdots v_j)^\omega = uv^p x(yv^q x)^\omega \ ,$$

by assumption it belongs to L. Thanks to the equality $s(ts)^\omega = (st)^\omega$, the word above is equal to $uv^p(xyv^q)^\omega = uv^p(v^{q+1})^\omega = uv^\omega$. Thus, $uv^\omega \in L$. ◀

▶ **Theorem 7.** *The map $\cdot^{\texttt{ol}}$ is a bijection between ω-regular languages and ω-regular like cost functions.*

In particular, two ω-regular like cost functions $L^{\texttt{ol}}, L'^{\texttt{ol}}$ are equal if, and only if, $L = L'$.

Proof. The map is surjective by definition of ω-regular like cost functions.

We show that it is injective: consider L, L' two ω-regular languages such that $L^{\texttt{ol}} \approx L'^{\texttt{ol}}$. It follows from Lemma 6 that L and L' coincide on ultimately periodic words; being ω-regular, this implies that they are equal. ◀

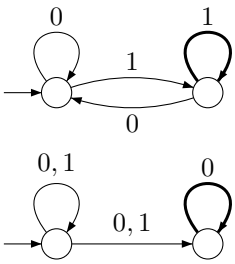

Figure 1 The Büchi automaton for Example 9.

3.2 Automata Constructions

The following theorem shows how to construct automata recognising ω-regular like cost functions. The construction is very simple, as it amounts to consider an ω-automaton and to see it as a prefix-max-cost-automaton, without any further changes. The correctness proof however is bit more involved.

▶ **Theorem 8.** *Let W be an ω-regular condition.*

Consider a W-automaton $\mathcal{A}$, and denote by L the language it recognises. The prefix-max-cost-automaton induced by $\mathcal{A}$ with the map $W^{\mathtt{ol}}$ recognises the cost function $L^{\mathtt{ol}}$.

Before proving this theorem, we give an example.

▶ **Example 9.** Consider the Büchi automaton represented in Figure 1. The alphabet is $\mathtt{A} = \{0, 1\}$, the Büchi transitions are represented by a boldface loop. The top part checks whether the word contains infinitely many 1's, and the bottom part checks whether the word contains finitely many 1's. It follows that this automaton recognises all ω-words, *i.e.* $L = \mathtt{A}^\omega$, so

$$\begin{aligned} L^{\mathtt{ol}} : \mathtt{A}^* &\to \mathbb{N} \cup \{\infty\} \\ w &\mapsto \text{length of } w \end{aligned}$$

The induced prefix-max-cost-automaton recognises the following cost function:

$$\begin{aligned} [\![\mathcal{A}]\!]_{\text{pmax}} : \mathtt{A}^* &\to \mathbb{N} \cup \{\infty\} \\ w &\mapsto \max\{\text{number of } 1's \text{ in } w, \text{size of the largest block of } 0's \text{ in } w\} . \end{aligned}$$

These two functions are indeed equivalent: $[\![\mathcal{A}]\!]_{\text{pmax}} \leq L^{\mathtt{ol}} \leq [\![\mathcal{A}]\!]^2_{\text{pmax}}$.

In the proof of Theorem 8, we will make use of Simon's theorem [13]. We state here the corollary that we use. Recall that a semigroup is a set equipped with an associative binary product, and that an idempotent in a semigroup is an element e such that $e \cdot e = e$.

For every morphism $\varphi : \mathtt{A}^+ \to M$ where M is a finite semigroup, there exists a function $\alpha : \mathbb{N} \to \mathbb{N}$ such that $\lim \alpha = \infty$ and for all words w of length n, there exists a factorisation $w = uv_1 \cdots v_{\alpha(n)}u'$ where the words $v_1, \ldots, v_{\alpha(n)}$ are non-empty and such that

$$\varphi(v_1) = \varphi(v_2) = \cdots = \varphi(v_{\alpha(n)})$$

is an idempotent.

Proof. For the sake of simplicity, we will assume that $\mathcal{A}$ is a parity automaton, *i.e.* W is the parity language. The proof generalises to the case of an ω-regular language W by considering a deterministic parity automaton recognising W.

We denote $\mathcal{A}^{\mathtt{ol}}$ the prefix-max-cost-automaton induced by $\mathcal{A}$ with the accepting map $\mathtt{Parity}^{\mathtt{ol}}$. By definition:

- $[\![\mathcal{A}^{\mathtt{ol}}]\!]_{\mathrm{pmax}}(w)$ is defined by

$$\sup\left\{\mathtt{Parity}^{\mathtt{ol}}(\rho|_B) \mid \rho \text{ prefix run over } w\right\}.$$

- $L^{\mathtt{ol}}(w)$ is defined by

$$\sup\left\{n \mid w = uv_1\cdots v_nu',\ v_1,\dots,v_n \neq \varepsilon,\ u\cdot\{v_1,\dots,v_n\}^\omega \subseteq L\right\}.$$

We will apply Simon's theorem twice, once for each direction. To this end, we construct a morphism $\varphi : \mathtt{A} \to M$, where M is the transition semigroup of $\mathcal{A}$. A transition profile is a tuple (p,c,q) where p and q are states and c is a colour. The product of transition profiles is (partially) defined by

$$(p,c,q)\cdot(r,c',s) = \begin{cases} (p,\min(c,c'),s) \text{ if } q=r \\ \text{undefined otherwise.}\end{cases}$$

An element of M is a set of transition profiles. The product is inherited by the product for transition profiles. The morphism φ associates to a letter a the set of transitions over the letter a in the automaton $\mathcal{A}$.

Assume $[\![\mathcal{A}^{\mathtt{ol}}]\!]_{\mathrm{pmax}}(w) \geq n$: there exists a prefix run ρ over w such that $\mathtt{Parity}^{\mathtt{ol}}(\rho|_B) \geq n$. It follows that ρ factorises $\rho\rho_1\cdots\rho_n\rho'$, where in each ρ_i the smallest colour appearing is even. We apply Simon's theorem to the word $\rho_1\cdots\rho_n$, seen as a word of length n, *i.e.* where we interpret each ρ_i as a single letter. Denote $m=\alpha(n)$. There exists a factorisation which we denote $\widetilde{\tau}\widetilde{\tau_1}\cdots\widetilde{\tau_m}\widetilde{\tau'}$ such that $\varphi(\widetilde{\tau_1}) = \cdots = \varphi(\widetilde{\tau_m})$ is idempotent, denoted S. This implies the existence of (q,c,q) in S, where c is even. Denote $w = uv_1\cdots v_mu'$ the factorisation of w it induces. Observe that $u\cdot\{v_1,\dots,v_m\}^\omega \subseteq L$, as for each such word one can construct a ultimately periodic run ρ in $\mathcal{A}$ whose smallest colour appearing infinitely often is c, hence such that $\rho|_B \in \mathtt{Parity}$. So $W^{\mathtt{ol}}(\rho) \geq \alpha(n)$.

It follows that $[\![\mathcal{A}^{\mathtt{ol}}]\!]_{\mathrm{pmax}} \preccurlyeq L^{\mathtt{ol}}$.

Conversely, assume $L^{\mathtt{ol}}(w) \geq n$: there exists a factorisation of w in $uv_1\cdots v_nu'$ as in the definition of $L^{\mathtt{ol}}(w)$. We apply Simon's theorem to the word $v_1\cdots v_n$, seen as a word of length n, *i.e.* where we interpret each v_i as a single letter. Denote $m=\alpha(n)$. There exists a factorisation which we denote $\widetilde{u}\widetilde{v_1}\cdots\widetilde{v_m}\widetilde{u'}$ such that $\varphi(\widetilde{v_1}) = \cdots = \varphi(\widetilde{v_m})$ is idempotent, denoted S. Note that each $\widetilde{v_\ell}$ and $\widetilde{u}$ is an infix $v_i\cdots v_j$; denote $\widetilde{w}$ the infix corresponding to $\widetilde{v_1}\cdots\widetilde{v_m}$. The element $\varphi(\widetilde{w})$ is idempotent equal to S. Since $u\cdot\{v_1,\dots,v_n\}^\omega \subseteq L$, in particular $u\widetilde{u}\cdot\widetilde{w}^\omega \in L$. Because $\mathcal{A}$ recognises L, there exists an accepting run of $u\widetilde{u}\cdot\widetilde{w}^\omega$. Now, $\varphi(\widetilde{w})$ being idempotent, this implies that there exist:

- a transition profile in $\varphi(u\cdot\widetilde{u})$ of the form $(p,\_,q)$ where p is initial, and
- a transition profile in $\varphi(\widetilde{w})$ of the form (q,c,q) where c is even.

Recall that each $\varphi(\widetilde{v_\ell})$ is equal to $S=\varphi(\widetilde{w})$, so it contains (q,c,q). Thus, we obtain a run $\rho = \rho_{u\widetilde{u}}\rho_{\widetilde{v_1}}\cdots\rho_{\widetilde{v_m}}$ over $u\widetilde{u}\widetilde{v_1}\cdots\widetilde{v_m}$ such that $\rho_{u\widetilde{u}}\left\{\rho_{\widetilde{v_1}},\dots,\rho_{\widetilde{v_m}}\right\}^\omega \subseteq W$. This implies that $[\![\mathcal{A}^{\mathtt{ol}}]\!]_{\mathrm{pmax}}(w) \geq \alpha(n)$.

It follows that $L^{\mathtt{ol}} \preccurlyeq [\![\mathcal{A}^{\mathtt{ol}}]\!]_{\mathrm{pmax}}$.

We conclude that $L^{\mathtt{ol}}$ and $[\![\mathcal{A}^{\mathtt{ol}}]\!]_{\mathrm{pmax}}$ are equivalent. ◀

Recall that if W is the Büchi language, then $W^{\circ 1}$ is the distance map. Similarly, the Rabin condition induces the $\mathtt{cost}_{\mathtt{B}}$ map and the parity condition the $\mathtt{cost}_{\mathtt{hB}}$ map.

In particular, Theorem 8 implies that if L is recognised by a Büchi automaton (resp. Rabin automaton, parity automaton), then $L^{\circ 1}$ is recognised by a prefix-max-cost-automaton with the distance map (resp. $\mathtt{cost}_{\mathtt{B}}$ map, $\mathtt{cost}_{\mathtt{hB}}$ map).

3.3 Syntactical Constructions

The above subsection shows how to construct ω-regular like cost functions using automata.

3.4 Transferring Results

We show in this subsection how to use the above correspondence to transfer two automata theoretic constructions.

The first construction is the latest appearance record construction, which allows to transform a Rabin condition into a parity condition, as stated in the following theorem.

▶ **Theorem 10.** *For every k, there exists a deterministic parity automaton with $k!$ states and k colours recognising the language $\mathtt{Rabin}_k$.*

This yields the following corollary.

▶ **Corollary 11.** *For every k, there exists a hierarchical B-automaton (hB-automaton) with $k!$ states and k counters recognising the cost function $\mathtt{cost}_{\mathtt{B}}$.*

Consequently, for every regular cost function, one can effectively construct an hB-automaton recognising it.

The first part is obtained by applying Theorem 8 to the automaton constructed by Theorem 10. For the second part, it amounts to compose the B-automaton with the automaton constructed by the first item to obtain an hB-automaton.

The second construction is the determinisation of Büchi automata.

▶ **Corollary 12.** *For every ω-regular like cost function, one can effectively construct a deterministic B-automaton recognising it.*

Proof. Consider an ω-regular language L given by a non-deterministic Büchi automaton, inducing the ω-regular like cost function $L^{\circ 1}$.

The McNaughton-Safra construction yields an equivalent deterministic Rabin automaton, denoted $\mathcal{A}$. Thanks to Theorem 8, this implies a prefix-max-cost-automaton equipped with the $\mathtt{Rabin}^{\circ 1}$ condition recognising $L^{\circ 1}$. Since $\mathcal{A}$ is deterministic and $\mathtt{Rabin}^{\circ 1}$ is the $\mathtt{cost}_{\mathtt{B}}$ map, $\mathcal{A}$ is in fact a deterministic B-automaton recognising $L^{\circ 1}$. ◀

3.5 Games with Omega-Regular like Cost Functions

In this subsection, we show how to solve games with ω-regular like cost functions.

We refer to [9] for materials about games; here we only give the basic definitions.

A *game* is denoted $G = (V, \mathtt{A}, V_E, V_A, E)$, where V is a set of vertices, $\mathtt{A}$ is the output alphabet, V_E is the set of vertices controlled by the first player Eve, V_A is the set of vertices controlled by the opponent Adam with $V = V_E \uplus V_A$ and $E \subseteq V \times \mathtt{A} \times V$ is the set of edges. A game is said finite if V is finite.

A token is initially placed on a given initial vertex v_0, and the player who controls this vertex pushes the token along an edge, reaching a new vertex; the player who controls this

new vertex takes over, and this interaction goes on forever, describing an infinite path called a play. A *winning condition* is a language $L \subseteq \mathtt{A}^\omega$: a play is won by Eve if its projection on $\mathtt{A}$ belongs to L. A *strategy* for Eve is a map $\sigma : E^* V_E \to E$. A *memory structure* is denoted $\mathcal{M} = (M, m_0, \mu)$, where M is the (finite) set of memory states, $m_0 \in M$ is the initial memory state and $\mu : M \times E \to M$ is the (deterministic) update function. A *finite-memory strategy* is given by a memory structure $\mathcal{M}$ and a next-move function $\sigma : M \times V_E \to E$.

▶ **Theorem 13.** *Consider a finite game G and L an ω-regular language. The following are equivalent:*

1. *There exists n, there exists a strategy for Eve, such that for all plays, the value for $L^{\mathtt{o1}}$ is less than n,*
2. *Eve wins for the winning condition $L^{\mathtt{C}}$.*

Proof. Since L is ω-regular, it is recognised by a deterministic Rabin automaton. By considering the product of the game with this automaton, we can assume without loss of generality that $L = \mathtt{Rabin}$, so $L^{\mathtt{o1}} = \mathtt{cost}_{\mathtt{B}}$.

The top to bottom direction is clear: indeed, for a play π, if $\mathtt{cost}_{\mathtt{B}}(\pi) \leq n$, then $\pi \in \mathtt{Rabin}^{\mathtt{C}}$.

To prove the converse implication, we rely on the fact that since $L^{\mathtt{C}}$ is an ω-regular condition, Eve has a finite-memory winning strategy. By considering the product of the game with the memory structure, we observe that in each cycle, for each counter, either it is not incremented or it is both incremented and reset. It follows that this strategy ensures that the values for $\mathtt{cost}_{\mathtt{B}}$ is bounded over all plays by twice the size of the graph times the size of the memory. ◀

We can strengthen this theorem:

▶ **Theorem 14.** *Consider a finite game G and L, L' two ω-regular languages. The following are equivalent:*

1. *For all n, there exists n', there exists a strategy for Eve, such that for all plays:*

   if the value for $L^{\mathtt{o1}}$ is less than n then the value for $L'^{\mathtt{o1}}$ is less than n',

2. *Eve wins for the condition $L \cup L'^{\mathtt{C}}$.*

Proof. Since L and L' are ω-regular, they are each recognised by a deterministic Rabin automaton. By considering the product of the game with the two automata, we can assume without loss of generality that the alphabet is $\{\epsilon, \mathtt{i}, \mathtt{r}\}^k \times \{\epsilon, \mathtt{i}, \mathtt{r}\}^{k'}$ with $L = \mathtt{Rabin}^1$ and $L' = \mathtt{Rabin}^2$. Thus $L^{\mathtt{o1}} = \mathtt{cost}^1_{\mathtt{B}}$ and $L'^{\mathtt{o1}} = \mathtt{cost}^2_{\mathtt{B}}$.

Assume that Eve wins for the condition $L \cup L'^{\mathtt{C}}$: since it is ω-regular, Eve has a finite-memory winning strategy. By considering the product of the game with the memory structure, we observe that for each cycle,

if for each counter in $\{\epsilon, \mathtt{i}, \mathtt{r}\}^k$, it is either reset or not incremented,
then for each counter in $\{\epsilon, \mathtt{i}, \mathtt{r}\}^{k'}$, it is either reset or not incremented.

Let n be twice the size of the graph times the size of the memory. It follows that this strategy ensures that for all plays, if the value for $\mathtt{cost}^1_{\mathtt{B}}$ is less than n then the value for $\mathtt{cost}^2_{\mathtt{B}}$ is less than n.

To prove the converse, we proceed by contrapositive. Assume that Eve does not win for the condition $L \cup L'^{\mathtt{C}}$, since the game is determined this implies that Adam wins, and again because the winning condition is ω-regular Adam has a finite-memory winning strategy. The same reasoning as before concludes that each cycle satisfies the negation of the above

property, which implies for the same value of n that this strategy ensures the following: for all n', there exists a play such that the value for $\mathtt{cost}^1_{\mathtt{B}}$ is less than n and the value for $\mathtt{cost}^2_{\mathtt{B}}$ is greater than n'. ◀

4 History-Determinisation of Cost Automata

In this section we give a simple and direct procedure for history-determinisation of B-automata: given a B-automaton, construct an equivalent history-deterministic B-automaton. Note that for the sake of simplicity we consider here hierarchical B-automata. Our contruction relies on the properties we obtained for ω-regular like cost functions in the above section together with game theoretic techniques inspired by Bojańczyk [3].

An automaton is *history-deterministic* if it is non-deterministic but its non-determinism can be resolved by a function considering only the input read so far. This notion has been introduced for studying ω-automata in [11]. We specialise it here to the case of cost functions, involving a relaxation on the values allowing for a good interplay with the definition of equivalence for cost functions.

A B-automaton $\mathcal{B}$ is *history-deterministic* if there exists a function $\alpha : \mathbb{N} \to \mathbb{N}$ such that $\lim \alpha = \infty$ and for every n, there exists a strategy $\sigma : \mathtt{A}^* \to \Delta$ such that for all words w, we have

$$[\![\mathcal{B}]\!]_{\min}(w) \leq n \implies [\![\mathcal{B}_\sigma]\!]_{\min}(w) \leq \alpha(n).$$

The automaton $\mathcal{B}_\sigma$ is infinite but deterministic, as for each situation the strategy σ chooses the transition to follow.

▶ **Theorem 15.** *For every hB-automaton, one can effectively construct an equivalent history-deterministic hB-automaton.*

Let $\mathcal{A} = (Q, \mathtt{A}, \{I_1, R_1, \ldots, I_k, R_k\}, I, \Delta, F, \mathtt{cost}_{\mathtt{hB}})$ be a hB-automaton. We first sketch the construction, which involves two automata:

- a deterministic hB-automaton $\mathcal{C}$ recognising an ω-regular like cost function denoted $L^{\circ\mathtt{l}}$,
- a history-deterministic min-cost-automaton $\mathcal{B}$ equipped with the map $L^{\circ\mathtt{l}}$.

Recall that for a word w, the value of $[\![\mathcal{A}]\!]_{\min}(w)$ is the minimum value for the $\mathtt{cost}_{\mathtt{hB}}$ map over all runs of w.

The automaton $\mathcal{B}$ simulates $\mathcal{A}$ and is in charge of guessing an optimal run, *i.e.* having minimal value for the $\mathtt{cost}_{\mathtt{hB}}$ map. However, we want $\mathcal{B}$ to be history-deterministic; to achieve this, $\mathcal{B}$ will do something easier than guessing *one* run, it will guess for each transition whether it belongs to *some* optimal run. In other words, $\mathcal{B}$ guesses a run tree (see Figure 2 for the representation of a run tree). As we shall see, thanks to the positionality of hB-games, $\mathcal{B}$ can guess a set of near optimal runs in a history-deterministic fashion. In effect, $\mathcal{B}$ inputs a word and outputs a run tree.

The automaton $\mathcal{C}$ recognises the cost function which given a run tree computes the maximum value for the $\mathtt{cost}_{\mathtt{hB}}$ map over all paths in the run tree. The crucial point is that this cost function is ω-regular like, so one can effectively construct a deterministic hB-automaton recognising it.

The composition of the automata $\mathcal{B}$ and $\mathcal{C}$ yields a history-deterministic hB-automaton equivalent to $\mathcal{A}$. The correctness of this construction relies on the following two properties:

- $[\![\mathcal{A}]\!]_{\min}$ and $[\![\mathcal{B}]\!]_{\min}$ are equivalent,
- $\mathcal{B}$ is history-deterministic.

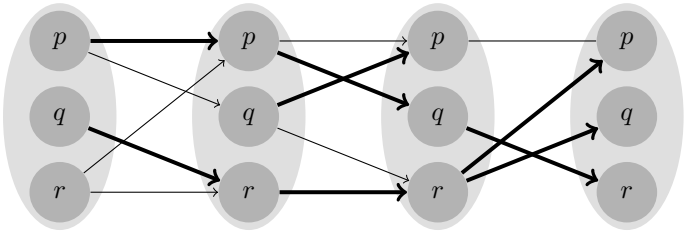

Figure 2 A run tree. Each state has at most one ingoing transition.

We proceed with the formal construction.

We define the alphabet $\mathtt{B}$, which is the output alphabet of $\mathcal{B}$ and the input alphabet of $\mathcal{C}$. A transition profile is a triple (p, act, q) where p and q are states and act is an action on the k counters, *i.e.* $act \in \{I_1, R_1, \dots, I_k, R_k\}$. An element of $\mathtt{B}$ is a set of transition profiles T such that for every state q, there exists *at most* one p such that $(p, act, q) \in T$ for some act. Equivalently, it is a partial function $T : Q \to \{I_1, R_1, \dots, I_k, R_k\} \times Q$; this backward point of view will be useful in proving that $\mathcal{B}$ is history-deterministic. A word over this alphabet is called a run tree; see Figure 2 for the representation of a run tree.

Construction of $\mathcal{C}$. The automaton $\mathcal{C}$ recognises an ω-regular like cost function; to construct it we define a parity automaton and turn it into a prefix-max-cost automaton relying on Theorem 8. Denote

$$L = \left\{ t \in \mathtt{B}^\omega \;\middle|\; \begin{array}{c} \text{there exists an infinite path in } t \text{ such that,} \\ \text{the minimal action performed infinitely often} \\ \text{for the ordering } I_1 < R_1 < \cdots < I_k < R_k \\ \text{is } R_\ell \text{ for some } \ell \end{array} \right\}.$$

The language L is recognised by a non-deterministic parity automaton of linear size which guesses the witnessing path. Formally, the parity automaton is

$$(Q, \mathtt{B}, \{I_1, R_1, \dots, I_k, R_k\}, I, \Delta_{\mathcal{C}}, \mathtt{Parity})$$

The transition relation is $\Delta_{\mathcal{C}} = \{(p, T, act, q) \mid (p, act, q) \in T\}$. The parity condition is obtained by seeing R_ℓ as the colour 2ℓ and I_ℓ as the colour $2\ell - 1$.

Theorem 8 implies that

$$L^{\mathtt{ol}} \approx \left\{ \begin{array}{lcl} \mathtt{B}^* & \to & \mathbb{N} \cup \{\infty\} \\ t & \mapsto & \max \{\mathtt{cost}_{\mathtt{hB}}(\pi) \mid \pi \text{ prefix path in } t\} \end{array} \right.$$

Following Corollary 12, we can effectively construct a deterministic hB-automaton $\mathcal{C}$ recognising $L^{\mathtt{ol}}$.

Construction of $\mathcal{B}$. The automaton $\mathcal{B}$ is a min-cost-automaton equipped with the map $L^{\mathtt{ol}}$, in charge of guessing a run tree and deterministically checking whether it contains a run.

Formally, $\mathcal{B} = (\mathcal{P}(Q), \mathtt{A}, \mathtt{B}, \mathcal{P}(I), \Delta_{\mathcal{B}}, F_{\mathcal{B}}, L^{\mathtt{ol}})$. The transition relation $\Delta_{\mathcal{B}}$ is defined by

$$\{(S, a, T, S') \mid S' \text{ is the set of states reached from } S \text{ using the transitions in } T\}.$$

The set of final states $F_{\mathcal{B}}$ is $\{S \subseteq Q \mid S \cap F \neq \emptyset\}$.

▶ **Lemma 16.** *$[\![\mathcal{A}]\!]_{\min}$ and $[\![\mathcal{B}]\!]_{\min}$ are equivalent.*

Proof. By definition

- $[\![\mathcal{A}]\!]_{\min}(w)$ is the minimum value for the $\mathtt{cost}_{\mathtt{hB}}$ map over all runs of w.
- $[\![\mathcal{B}]\!]_{\min}(w)$ is the minimum value for the $L^{\mathtt{o1}}$ map over all runs of w. By construction of $\mathcal{B}$, the runs of w are the run trees of w that contain a run of w over $\mathcal{A}$.

Thanks to the equivalence above about $L^{\mathtt{o1}}$, this implies that

$$[\![\mathcal{B}]\!]_{\min} \approx \left\{ \begin{array}{lcl} \mathtt{A}^* & \to & \mathbb{N} \cup \{\infty\} \\ w & \mapsto & \min \left\{ \mathtt{cost}_{\mathtt{hB}}(t) \;\middle|\; \begin{array}{c} t \text{ run tree of } w \text{ which} \\ \text{contains a run of } w \text{ over } \mathcal{A} \end{array} \right\}, \end{array} \right.$$

where $\mathtt{cost}_{\mathtt{hB}}(t) = \max\{\mathtt{cost}_{\mathtt{hB}}(\pi) \mid \pi \text{ prefix path in } t\}$.

Let $[\![\mathcal{A}]\!]_{\min}(w) \leq n$: there exists a run ρ of w such that $\mathtt{cost}_{\mathtt{hB}}(\rho) \leq n$. Consider the run tree t consisting of exactly ρ, it satisfies $\mathtt{cost}_{\mathtt{hB}}(t) \leq n$. It follows that $[\![\mathcal{B}]\!]_{\min} \preccurlyeq [\![\mathcal{A}]\!]_{\min}$.

Conversely, let $[\![\mathcal{B}]\!]_{\min}(w) \leq n$: there exists a run tree t of w such that $\mathtt{cost}_{\mathtt{hB}}(t) \leq n$. Because it is a run of $\mathcal{B}$, there exists a run ρ in t, and $\mathtt{cost}_{\mathtt{hB}}(t) \leq n$ implies that $\mathtt{cost}_{\mathtt{hB}}(\rho) \leq n$. It follows that $[\![\mathcal{A}]\!]_{\min} \preccurlyeq [\![\mathcal{B}]\!]_{\min}$.

We conclude that $[\![\mathcal{A}]\!]_{\min}$ and $[\![\mathcal{B}]\!]_{\min}$ are equivalent. ◀

▶ **Lemma 17.** *$\mathcal{B}$ is history-deterministic.*

This relies on the following positionality result, which is proved in [7]. It is also in essence in the proof of Bojańczyk [3].

▶ **Theorem 18** ([7])**.** *Eve has positional uniform strategies in hB-games.*

We now prove Lemma 17.

Proof. To prove that $\mathcal{B}$ is history-deterministic, we show that there exists a function $\alpha : \mathbb{N} \to \mathbb{N}$ such that $\lim \alpha = \infty$ and for every n, there exists a strategy $\sigma : \mathtt{A}^* \to \Delta_{\mathcal{B}}$ such that for all words w, if $[\![\mathcal{B}]\!]_{\min}(w) \leq n$ then $[\![\mathcal{B}_\sigma]\!]_{\min}(w) \leq \alpha(n)$.

Observe that $\sigma : \mathtt{A}^* \to \Delta_{\mathcal{B}}$ can equivalently defined as a partial function $\sigma : Q \times \mathtt{A}^* \to \{I_1, R_1, \ldots, I_k, R_k\} \times Q$; what $\mathcal{B}$ guesses is for each state q, at most one transition leading to q.

We define an hB-game. The set of vertices is $Q \times \mathtt{A}^*$. The edges are

$$\{((wa, q), act, (w, p)) \mid (p, a, act, q) \in \Delta\}.$$

By definition, for all words w such that $[\![\mathcal{B}]\!]_{\min}(w) \leq n$, there exists $q \in F$ such that Eve has a strategy ensuring $\mathtt{cost}_{\mathtt{hB}}(n) \cap \mathtt{Safe}(\varepsilon, Q \setminus I)$. It follows from Theorem 18 that there exists a uniform positional strategy, *i.e.* $\sigma : Q \times \mathtt{A}^* \to \Delta$. By definition, for this strategy we have $[\![\mathcal{B}_\sigma]\!]_{\min}(w) \leq n$. It follows that $\mathcal{B}$ is history-deterministic. ◀

Composing the two automata yields a history-deterministic automaton equivalent to $\mathcal{A}$. Denote n the number of states of $\mathcal{A}$, the constructed automaton has $2^n \times \mathtt{Safra}(n)$ states, where $\mathtt{Safra}(n)$ is the number of states obtained by applying the Safra determinisation on an ω-automaton with n states. Since $\mathtt{Safra}(n) = 2^{O(n \log(n))}$, the constructed automaton also has $2^{O(n \log(n))}$ states.

Acknowledgements. We thank the anonymous reviewers for their constructive comments and suggestions.

References

1. Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. The complexity of boundedness for guarded logics. In *LICS*, pages 293–304, 2015.
2. Achim Blumensath, Martin Otto, and Mark Weyer. Decidability results for the boundedness problem. *Logical Methods in Computer Science*, 10(3), 2014.
3. Mikołaj Bojańczyk. Star height via games. In *LICS*, pages 214–219, 2015. doi:10.1109/LICS.2015.29.
4. Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *ICALP*, pages 139–150, 2009.
5. Thomas Colcombet. Safra-like constructions for regular cost functions over finite words. Unpublished, March 2011.
6. Thomas Colcombet. Regular cost functions, part I: logic and algebra over words. *Logical Methods in Computer Science*, 9(3), 2013.
7. Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *ICALP*, pages 398–409, 2008.
8. Thomas Colcombet and Konrad Zdanowski. A tight lower bound for determinization of transition labeled büchi automata. In *ICALP*, pages 151–162, 2009.
9. Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.
10. Kosaburo Hashiguchi. Improved limitedness theorems on finite automata with distance functions. *Theoretical Computer Science*, 72(1):27–38, 1990.
11. Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *CSL*, pages 395–410, 2006.
12. Daniel Kirsten. Distance desert automata and the star height problem. *ITA*, 39(3):455–509, 2005.
13. Imre Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990.
14. Imre Simon. On semigroups of matrices over the tropical semiring. *ITA*, 28(3-4):277–294, 1994.