The Bridge Between Regular Cost Functions and Omega-Regular Languages

Thomas Colcombet and Nathanaël Fijalkow

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

Abstract

In this paper, we exhibit a one-to-one correspondence between \(\omega\)-regular languages and a subclass of regular cost functions over finite words, called \(\omega\)-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 an optimal and simple 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, Quantitative Extensions of Automata, Determinisation of Automata, Safra Constructions

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

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 [11, 13] and finite trees [9], the finite power property [16], the boundedness of fixpoints for monadic second-order logic [2] or over guarded logic [1], or for attacking the Mostowski index problem [9]. 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 developing the theory of regular cost functions over finite trees [9]. 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. Another version inspired by the construction of Safra is known [5], however the proof is extremely technical and long.

The aim of this work is to give a simple description of the construction from [5], as well as a simple proof for it: given a cost-automaton, our goal is to construct an equivalent
2 Omega-Regular like Cost Functions

history-deterministic cost automaton. The main difference is that this novel presentation uses
the determinisation of $\omega$-regular languages as a black box. In particular, it does not depend
at all on the details of the complicated Safra construction. This 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 $\omega$-regular languages.

This is achieved by first describing a one-to-one correspondence between the theory
of $\omega$-regular languages and a subclass of regular cost functions, called the $\omega$-regular like
cost functions. This correspondence allows us to readily import from $\omega$-regular languages
constructions such as the last appearance record or determinisation results to regular cost
functions. In other words, $\omega$-regular like cost functions are determinisable.

In a second step, in combination with game theoretic techniques and 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 $\omega$-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
$\omega$-regular like cost functions combined with game theoretic techniques.

2 Definitions

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

The prefix run over a word $u = a_1a_2 \ldots$ (which can be finite or infinite) is a sequence of
transitions of the form

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

of length possibly smaller than the one of $u$, and such that $p_0$ is initial. Given a prefix run $\rho$,
we denote by $\rho|_A$ its projection to the alphabet $A$, and by $\rho|_B$ its projection to the alphabet $B$. If $A$ is an initial automaton structure, then a run over an infinite word $u \in A^\omega$ is an
infinite prefix run $\rho$ such that $\rho|_A = u$. If $A$ is an automaton structure, then a run over a
finite word $u \in A^*$ is a prefix run such that the last state is accepting, and $\rho|_B = u$.

$\omega$-automata

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

$$\{ \rho|_A \mid \rho \text{ run such that } \rho|_B \in W \}.$$
We define some of the classical $\omega$-accepting conditions.

- **Büchi** $= \{ w \in \{0, 1\}^\omega \mid w \text{ contains infinitely many } 0's \}$

- **coBüchi** $= \{ w \in \{1, 2\}^\omega \mid w \text{ contains finitely many } 1's \}$

- **Rabin** $\mathbb{1}$ $= \{ w \in \{I, R, \epsilon\}^\omega \mid w \text{ contains infinitely many } I's \} \text{ and finitely many } R's \}$

- **Rabin** $\mathbb{k}$ $= \{ w \in \{I, R, \epsilon\}^\omega \mid \text{ for all } \ell \in \{1, \ldots, k\}, \text{ w contains infinitely many } I'_{\ell} \text{ s} \} \text{ and finitely many } R'_{\ell} \text{ s} \}$

- **Parity** $\mathbb{k}$ $= \{ w \in \{0, \ldots, k\}^\omega \mid \text{ the smallest color appearing infinitely often in } w \text{ is even} \}$

A *Rabin automaton* is an $\omega$-automaton with a Rabin condition, and similarly for the other conditions. It is known that all these $\omega$-automata recognise the same $\omega$-languages, that are called the $\omega$-regular languages.

**Regular cost functions**

We consider functions from $A^*$ to $\mathbb{N} \cup \{\infty\}$. Let $f$ be such a function, and $X \subseteq A^*$, 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 $f \preceq g$, if for all $X \subseteq A^*$, if $g|_X$ is bounded then $f|_X$ is bounded. We say that $f$ and $g$ are equivalent, denoted $f \approx g$, if $f \preceq g$ and $g \preceq f$. The following lemma is central, see [7] for more considerations on this equivalence relation.

**Lemma 1.** $f \preceq g$ if, and only if, $f \leq \alpha \circ g$ for some non-decreasing $\alpha : \mathbb{N} \to \mathbb{N}$ extended with $\alpha(\infty) = \infty$.

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

In the theory of cost functions, many equivalent formalisms can be used in order to define regular cost functions. We use here mainly automata.

**Definition 2.** A *min-cost-automaton* (resp. *max-cost-automaton*) is denoted $A = (Q, A, B, I, F, g)$, given by an automaton structure $(Q, A, B, I, F)$ together with an accepting map $f : B^* \to \mathbb{N} \cup \{\infty\}$. It defines the map

$$[[A]]_{\text{min}} : A^* \to \mathbb{N} \cup \{\infty\}$$

$$w \mapsto \inf \{ f(\rho|_B) \mid \rho \text{ run of } A \text{ over } w \}$$

(resp. $[[A]]_{\text{max}} : w \mapsto \sup \{ f(\rho|_B) \mid \rho \text{ run of } A \text{ over } w \}$)

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

We first define the $\text{cost}_B$ 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 $\epsilon$ does nothing. Formally, the $\text{cost}_B$ map for one counter is defined by

$$\text{cost}_B : \{i, r, \epsilon\}^* \to \mathbb{N} \cup \{\infty\}$$

$$w \mapsto \max\{ n \in \mathbb{N} \mid w \in \{i, r, \epsilon\}^*(\epsilon^*i)^{n}\{i, r, \epsilon\}^* \}$$
The special cases over \( \{ \epsilon, i \}^* \) and over \( \{ r, i \}^* \) are called distance and desert, respectively denoted \( \text{dist}_B \) and \( \text{desert}_B \).

The \( \text{cost}_B \) map for \( k \) counters is defined similarly as for one counter, over the alphabet \( \{ \epsilon, i, r \}^k \), by taking the maximum over all counters.

The \( \text{cost}_{AB} \) map for \( k \) hierarchical counters is the restriction of \( \text{cost}_B \) over the alphabet \( \{ R_0, I_1, R_1, \ldots, I_k, R_k \} \), where \( I_\ell \) increments the \( \ell \)th counter and resets all counters of smaller index, and \( R_\ell \) resets all counters of index smaller than or equal to \( \ell \) (\( R_0 \) is equivalent to \( \epsilon \)).

A \( B \)-automaton is a min-cost-automaton equipped with a \( \text{cost}_B \) map. Similarly, a \( hB \)-automaton is equipped with a \( \text{cost}_{AB} \) map.

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

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

The map it recognises is

\[
\llbracket \mathcal{A} \rrbracket_{pmax} : A^* \to \mathbb{N} \cup \{ \infty \} \\
w \mapsto \sup \{ f(\rho|_B) \mid \rho \text{ prefix run of } \mathcal{A} \text{ over } w \}
\]

### 3 Omega Regular like Cost Functions

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

In Subsection 3.1 we define an operator defining the class and fleshing out the correspondence. We then explain how to construct \( \omega \)-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 \( \omega \)-regular languages to \( \omega \)-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 \( \omega \)-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, \( L^{\omega_1} \) is defined by

\[
L^{\omega_1} : A^* \to \mathbb{N} \cup \{ \infty \} \\
w \mapsto \sup \{ n \mid w = uv_1 \cdots v_n w', v_1, \ldots, v_n \neq \epsilon, u \cdot \{ v_1, \ldots, v_n \}^\omega \subseteq L \}.
\]

A cost function is \( \omega \)-regular like if it contains a map \( L^{\omega_1} \) for some \( \omega \)-regular language \( L \).

Note that we will mostly be interested in using the definition of \( \cdot^{\omega_1} \) with \( \omega \)-regular languages.

▶ Example 5.

- \( \text{B"uchli}^{\omega_1} = \text{dist}_B \): it is the function counting the number of 1's, i.e. the distance map where 0 is \( \epsilon \) and 1 is increment.
- \( \text{coB"uchli}^{\omega_1} = \text{desert}_B \): it is the function counting the size of the largest block of 2's, i.e. the desert map where 1 is reset and 2 is increment.
Rabin\(a_1\) = \(\text{cost}_R\): it is the function counting the number of \(I\)'s in a block containing no \(R\)'s, i.e. the \(\text{cost}_R\) map for one counter where \(I\) is increment and \(R\) is reset.

Rabin\(a_2\) = \(\text{cost}_R\): it is the \(\text{cost}_R\) map for \(k\) counters where \(I_1\) is increment for the \(\ell\)th counter and \(R_k\) is reset for the \(i^\text{th}\) counter.

Parity\(a_k\) = \(\text{cost}_R\): it is the \(\text{cost}_R\) map for \(k\) counters.

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

\begin{itemize}
\item \textbf{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:
\begin{enumerate}
\item \(uv^\omega \in L\),
\item \((L^{a_1}(uv^n))_{n \in \mathbb{N}}\) tends to infinity.
\end{enumerate}
\end{itemize}

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 \(L^{a_1}\) for \(L\) an \(\omega\)-regular language.

\begin{proof}
One direction is clear: if \(uv^\omega \in L\), then \((L^{a_1}(uv^n))_{n \in \mathbb{N}}\) tends to infinity.

We prove the converse implication. Assume that \((L^{a_1}(uv^n))_{n \in \mathbb{N}}\) 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 \(|u_1v_1 \cdots v_{\ell}|\) for \(\ell \in \{1, \ldots, n\}\): two of them have the same value modulo \(|v|\), denote the corresponding words \(u'v_1 \cdots v_{\ell} \) and \(u''v_1 \cdots v_{\ell}\), 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 \(u'v_1 \cdots v_{\ell}\). It follows that writing \(v = xy\), we have \(u'v_1 \cdots v_i = uv^p x\) for some \(p \in \mathbb{N}\) and \(v_{i+1} \cdots v_{j} = yv^q x\) for some \(q \in \mathbb{N}\).

Consider the infinite word
\[ u'v_1 \cdots v_i (v_{i+1} \cdots v_j)^\omega = uv^p x (yv^q x)^\omega. \]

Thanks to the equality \(s(ts)^\omega = (st)^\omega\), this implies that the word above is equal to \(uv^p (xyv^q)^\omega = uv^p (v^q)^\omega = uv^\omega\). Thus, \(uv^\omega \in L\).
\end{proof}

\begin{itemize}
\item \textbf{Theorem 7.} The map \(\cdot^{a_1}\) is a bijection between \(\omega\)-regular languages and \(\omega\)-regular like cost functions.

In particular, two \(\omega\)-regular like cost functions \(L^{a_1}, L'^{a_1}\) are equal if, and only if, \(L = L'\).
\end{itemize}

\begin{proof}
The map is surjective by definition of \(\omega\)-regular like cost functions.

We show that it is injective: consider \(L, L'\) two \(\omega\)-regular languages such that \(L^{a_1} = L'^{a_1}\). It follows from Lemma 6 that \(L\) and \(L'\) coincide on ultimately periodic words; being \(\omega\)-regular, this implies that they are equal.
\end{proof}

### 3.2 Automata Constructions

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

\begin{itemize}
\item \textbf{Theorem 8.} Let \(W\) be an \(\omega\)-regular language.

Consider a \(W\)-automaton \(A\), and denote by \(L\) the language it recognises. The prefix-max-cost-automaton induced by \(A\) with the map \(W^{a_1}\) recognises the cost function \(L^{a_1}\).
\end{itemize}
Before proving this theorem, we give an example.

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

$$L^\omega : A^\ast \to N \cup \{\infty\}$$

$$w \mapsto \text{length of } w$$

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

$$[A]_{\text{pmax}} : A^\ast \to N \cup \{\infty\}$$

$$w \mapsto \max \{\text{number of 1’s in } w, \text{size of the largest block of 0’s in } w\}$$

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

In the proof of Theorem 8, we will make use of Simon’s theorem [15]. 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 $\varphi : A^\ast \to M$ where $M$ is a finite semigroup, there exists a function $\alpha : N \to N$ such that for all words $w$ of length $n$, there exists a factorisation $w = uv_1 \cdots v_n u'$ such that

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

is an idempotent.

**Proof.** We denote $A^\omega_\ast$ the prefix-max-cost-automaton induced by $A$ with the accepting map $W^\omega$. By definition:

- $[A^\omega_\ast]_{\text{pmax}}(w)$ is the maximum value of $W^\omega$ over all prefix runs of $w$, and for a prefix run $\rho$ over $w$, the value of $W^\omega(\rho)$ is

$$\sup \{n \mid \rho = \rho_0 \rho_1 \cdots \rho_n \rho_{n+1}, \rho_1, \ldots, \rho_n \neq \varepsilon, \rho_0 \cdot \{\rho_1, \ldots, \rho_n\}^\omega \subseteq W\}.$$

- $L^\omega(w)$ is defined by

$$\sup \{n \mid w = uv_1 \cdots v_n u', v_1, \ldots, v_n \neq \varepsilon, u \cdot \{v_1, \ldots, v_n\}^\omega \subseteq L\}.$$
Let $[A^{o_1}]_{p_{\text{max}}}(w) = n$: there exists a prefix run $\rho$ over $w$ such that $\rho$ factorises $\rho_1 \cdots \rho_n \rho_{n+1}$ as in the definition of $W^{o_1}(\rho)$. Denote $w = w_1 \cdots w_n u'$ the factorisation of $w$ induced by $\rho$. Since $\rho_1 \cdots \rho_n \subseteq W$ and $A$ recognises $L$, this implies that $u_1 \cdots u_n \subseteq L$, so $W^{o_1}(\rho) \geq n$.

It follows that $[A^{o_1}]_{p_{\text{max}}} \leq L^{o_1}$.

Conversely, let $L^{o_1}(w) = n$: there exists a factorisation of $w$ in $w_1 \cdots w_n u'$ as in the definition of $L^{o_1}(w)$.

Note that so far the fact that $W$ is an $\omega$-regular language was immaterial; it will be used now. For the sake of readability, we will now assume that $A$ is a parity automaton, i.e. $W$ is the parity language. The proof easily generalises to the case of an $\omega$-regular language $W$.

We construct a morphism $\varphi : k \to M$, where $M$ is the semigroup of transitions of $A$. To start with, a transition profile is a tuple $(p, c, q)$ where $p$ and $q$ are states and $c$ is a color. The product of transition profiles is (partially) defined by $(p, c, q) \cdot (r, c', s) = (p, \min(c, c'), s)$ if $q = r$

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

We apply Simon’s theorem as stated above, to the word $v_1 \cdots v_n$, seen as a word of length $n$. Denote $m = \alpha(n)$. There exists a factorisation which we denote $\tilde{v}_1 \cdots \tilde{v}_m n'$ such that $\varphi(\tilde{v}_i) = \ldots = \varphi(\tilde{v}_m)$ is idempotent, denoted $S$. Note that each $\tilde{v}_i$ and $\tilde{u}$ is an infix $v_i \cdots v_j$; denote $\tilde{w}$ the infix corresponding to $\tilde{v}_1 \cdots \tilde{v}_m$. The element $\varphi(\tilde{w})$ is idempotent equal to $S$. Since $u \cdot \{v_1, \ldots, v_n\}^\omega \subseteq L$, in particular $u\tilde{u} \cdot \tilde{w} \omega \in L$. Because $A$ recognises $L$, there exists an accepting run over $u\tilde{u} \cdot \tilde{w} \omega$. Now, $\varphi(\tilde{w})$ being idempotent, this implies that there exist:

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

Recall that each $\varphi(\tilde{v}_i)$ is equal to $S = \varphi(\tilde{w})$, so it contains $(q, c, q)$. Thus, we obtain a run $\rho = \rho_a \cdots \rho_{m-1} \rho_m$, over $u\tilde{u} \tilde{v}_1 \cdots \tilde{v}_m$ such that $\rho_{a} \cdots \rho_{m-1} \rho_m \subseteq W$. This implies that $[A^{o_1}]_{p_{\text{max}}}(w) \geq \alpha(n)$.

It follows that $L^{o_1} \leq \alpha[A^{o_1}]_{p_{\text{max}}}$.

We conclude that $L^{o_1}$ and $[A^{o_1}]_{p_{\text{max}}}$ are equivalent. △

Recall that if $W$ is the Büchi language, then $W^{o_1}$ is the distance map. Similarly, the Rabin condition induces the cost map and the parity condition the cost map.

In particular, Theorem 8 implies that if $L$ is recognised by a Büchi automaton (resp. Rabin automaton, parity automaton), then $L^{o_1}$ is recognised by a prefix-max-cost-automaton with the distance map (resp. cost map, cost map).

### 3.3 Syntactical Constructions

The above subsection shows how to construct $\omega$-regular like cost functions using automata. Similar statements can be obtained by considering $\omega$-semigroups and $\omega$-regular expressions.

In particular, the algebraic framework allows to give a decidable characterisation of the class of $\omega$-regular like cost functions.

We refer to the appendix for the definitions and results regarding the algebraic approach to $\omega$-regular like cost functions.
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 acceptance condition into a parity acceptance condition, as stated in the following theorem.

- Theorem 10. For every $k$, there exists a deterministic parity automaton with $k!$ states and $k$ colors recognising the language $\text{Rabin}_k$.

This yields the following corollary.

- Corollary 11. For every $k$, there exists a hierarchical $B$-automaton with $k!$ states and $k$ counters recognising the cost function $\text{cost}_B$.

Consequently, for every regular cost function, one can effectively construct a hierarchical $B$-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 a hierarchical $B$-automaton.

The second construction is the determinisation of Büchi automata.

- Corollary 12. For every $\omega$-regular like cost function, one can effectively construct a deterministic $B$-automaton recognising it.

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

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

3.5 Games with Omega-Regular like Cost Functions

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

We refer to [10] for materials about games; here we only give the basic definitions.

A game is denoted $G = (V, A, V_E, V_A, E)$, where $V$ is a set of vertices, $A$ 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, and $E \subseteq V \times A \times V$ 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 $L \subseteq \kappa^\omega$: a play is won by Eve if its projection on $A$ belongs to $L$. A strategy for Eve is a map $\sigma : E \times V_E \rightarrow E$. A memory structure is denoted $M = (M, m_0, \mu)$, where $M$ is the (finite) set of memory states, $m_0 \in M$ is the initial memory state and $\mu : M \times E \rightarrow M$ is the (deterministic) update function. A finite-memory strategy is given by a memory structure $M$ and a next-move function $\sigma : M \times V_E \rightarrow E$.

- Theorem 13. Consider a finite game $G$ and $L$ an $\omega$-regular language. The following are equivalent:
1. There exists $N$, there exists a strategy for Eve, such that for all plays, the value for $L^{\omega}$ is less than $N$.

2. Eve wins for the winning condition $L$.

**Proof.** Since $L$ is $\omega$-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 $L = \text{Rabin}$. So $L^{\omega} = \text{cost}_b$.

The top to bottom direction is clear: indeed, for a play $\pi$, if $\text{cost}_b(\pi) \leq N$, then $\pi \in \text{Rabin}$.

To prove the converse implication, we rely on the fact that since $L$ is a Rabin 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 $\text{cost}_b$ 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 $\omega$-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 $L^{\omega}$ is less than $N$ then the value for $L'^{\omega}$ is less than $N'$,

2. Eve wins for the condition $L^{\omega} \cup L'$.

**Proof.** Since $L$ and $L'$ are $\omega$-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 $\{\epsilon, i, r\}^k \times \{\epsilon, i, r\}^{k'}$ with $L = \text{Rabin}^1$ and $L' = \text{Rabin}^2$. Thus $L^{\omega} = \text{cost}_b^1$ and $L'^{\omega} = \text{cost}_b^2$.

Assume that Eve wins for the condition $L^{\omega} \cup L'$: since it is $\omega$-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 $\{\epsilon, i, r\}^k$, it is either reset or not incremented,

then for each counter in $\{\epsilon, i, r\}^{k'}$, 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 $\text{cost}_b^1$ is less than $N$ then the value for $\text{cost}_b^2$ is less than $N$.

To prove the converse, we proceed by contrapositive. Assume that Eve does not win for the condition $L^{\omega} \cup L'$, since the game is determined this implies that Adam wins, and again because the winning condition is $\omega$-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 $\text{cost}_b^1$ is less than $N$ and the value for $\text{cost}_b^2$ is greater than $N'$.

**4 History-Determinisation of Cost Automata**

In this section we give a simple and direct procedure for constructing a history-deterministic automaton from a B-automaton. This relies on the properties we obtained for $\omega$-regular
like cost functions in the above section together with game theoretic techniques inspired by Bojańczyk [3].

Informally, 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 \( \omega \)-automata in [12]. 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.

Formally, an automaton \( \mathcal{B} \) is *history-deterministic* if for every \( n \), there exists a strategy \( \sigma : A^* \to \Delta \) such that for all words \( w \), we have

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

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

**Theorem 15.** For every \( h\mathcal{B} \)-automaton, one can effectively construct an equivalent history-deterministic \( h\mathcal{B} \)-automaton.

Let \( \mathcal{A} \) be a \( h\mathcal{B} \)-automaton.

We first sketch the construction, which involves two automata:

- a deterministic \( h\mathcal{B} \)-automaton \( \mathcal{C} \) recognising an \( \omega \)-regular like cost function denoted \( L^{\omega^1} \),
- a history-deterministic min-cost-automaton \( \mathcal{B} \) equipped with the map \( L^{\omega^1} \).

Recall that for a word \( w \), the value of \( \mathcal{A}(w) \) is the minimum value for the cost map over all runs of \( w \).

The automaton \( \mathcal{B} \) simulates \( \mathcal{A} \) and is in charge of guessing a minimal run. 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. As we shall see, thanks to the positionality of \( h\mathcal{B} \) games, \( \mathcal{B} \) can guess a set of near minimal runs.

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 cost map over all paths in the run tree. The crucial point is that this cost function is \( \omega \)-regular like, so one can effectively construct a deterministic \( h\mathcal{B} \)-automaton recognising it.

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

- \( \mathcal{A} \) and \( \mathcal{B} \) are equivalent,
- \( \mathcal{B} \) is history-deterministic.

We proceed with the formal construction.

Denote \( \mathcal{C} = \{ R_0, I_1, R_1, \ldots, I_k, R_k \} \), the set of actions over \( k \) hierarchical counters, it will be the output alphabet of \( \mathcal{C} \).

We define the alphabet \( B \), which will be 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. \( act \in \mathcal{C} \), such that \( (p, a, act, q) \in \Delta \) for some \( a \).

An element of \( B \) 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 \( T : Q \to \mathcal{C} \times Q \); this backward point of view will be useful in proving that \( \mathcal{B} \) is history-deterministic. See Figure 2 for the representation of a run tree.

A word over this alphabet is called a run tree. To avoid confusion, we distinguish between runs and paths: the former satisfy that the last element is accepting, which may not be true.
A run tree.

Figure 2

for the latter. Given a run tree $t$, we say that a path $\pi$ belongs to $t$ if for every position the transition of $\pi$ exists in the run tree $t$. A partial path in $t$ is the prefix of a path in $t$.

**Construction of $C$.** Denote

$$
L = \left\{ t \in B^\omega \mid \right. \\
\text{there exists an infinite path in } t \text{ such that,} \\
\text{the minimal action performed infinitely often} \\
\text{for the ordering } R_0 < I_1 < R_1 < \cdots < I_k < R_k \\
\text{is } R_\ell \text{ for some } \ell \right\}.
$$

The language $L$ is $\omega$-regular, and recognised by a parity automaton of linear size. This automaton guesses the path, hence it is non-deterministic.

Formally, $A$ and $C$ share the same the set of states and set of initial states. The transitions are $\{(p,T,act,q) \mid (p,act,q) \in T\}$. The parity condition is obtained by seeing $R_\ell$ as the color $2\ell$ and $I_\ell$ as the color $2\ell + 1$.

Theorem 8 implies that

$$
L_{\text{ol}} \approx \begin{cases} 
B \rightarrow \mathbb{N} \cup \{\infty\} \\
\max \{\text{cost}_B(\pi) \mid \pi \text{ partial path in } t\}
\end{cases}
$$

As in Corollary 12, we can effectively construct a deterministic hB-automaton $C$ recognising $L_{\text{ol}}$.

**Construction of $B$.** The automaton $B$ has $A$ as input alphabet and $B$ as output alphabet.

It is a min-cost-automaton equipped with the map $L_{\text{ol}}$, in charge of guessing a run tree and checking whether it contains a run. Note that the run is guessed non-deterministically; however, checking whether it contains a run is achieved in a deterministic way, with an exponential deterministic automaton.

Formally, the set of states of $B$ is the powerset of $Q$, the initial states are the subsets $S \subseteq I$, and a set is final if it has a non-empty intersection with the set of final states of $A$.

The set of transitions of $B$, denoted $\Delta_B$, is defined by

$$
\begin{cases} 
B^* \rightarrow \mathbb{N} \cup \{\infty\} \\
t \rightarrow \max \{\text{cost}_B(\pi) \mid \pi \text{ partial path in } t\}
\end{cases}
$$

▶ **Lemma 16.** $A$ and $B$ are equivalent.

**Proof.** By definition

- $A(w)$ is the minimum value for the $\text{cost}_B$ map over all runs of $w$.
- $B(w)$ is the minimum value for the $L_{\text{ol}}$ map over all runs of $w$. By construction of $B$, the runs of $w$ are the run trees of $w$ that contain a run of $w$ over $A$.

Let $A(w) \geq n$: there exists a run $\rho$ of $w$ such that $B(\rho) = n$. Consider the run tree $t$ consisting of exactly $\rho$, it satisfies $L_{\text{ol}}(t) = n$, so $B(w) \leq n$. It follows that $B \leq A$. 
Conversely, let $B(w) \geq n$: there exists a run tree $t$ of $w$ such that $L^{o_1}(t) = n$. Because it is a run of $B$, there exists a run $\rho$ in $t$, and $L^{o_1}(t) = n$ implies that $B(\rho) \leq n$, so $A(w) \leq n$. It follows that $A \leq B$.

We conclude that $A$ and $B$ are equivalent. ▶

Lemma 17. $B$ is history-deterministic.

This relies on the following positionality result, which is proved in [9]. It is also in essence in the proof of Bojańczyk [3].

Theorem 18. Eve has positional uniform strategies in $hB$-games.

We now prove Lemma 17.

Proof. To prove that $B$ is history-deterministic, we construct a function $\alpha$ satisfying the following: for all $n$, there exists a strategy $\sigma : A^* \rightarrow \Delta_B$ such that for all words $w$, if $B(w) \leq n$ then $B(\sigma(w)) \leq \alpha(n)$.

Observe that $\sigma : A^* \rightarrow \Delta_B$ can equivalently defined as a partial function $\sigma : Q \times A^* \rightarrow C \times Q$; what $B$ guesses is for each state $q$, at most one transition leading to $q$.

We define an $hB$-game. The set of vertices is $Q \times A^*$. The edges are $\{(wa, q), act, (w, p) | (p, a, act, q) \in \Delta\}$.

By definition, for all words $w$ such that $B(w) \leq n$, there exists $q \in F$ such that Eve has a strategy ensuring $hB(n) \cap \text{Safe}(\epsilon, Q \setminus F)$. It follows from Theorem 18 that there exists a uniform positional strategy, i.e. $\sigma : Q \times A^* \rightarrow \Delta$. By definition, for this strategy we have $B(\sigma(w)) \leq n$. It follows that $B$ is history-deterministic. ▶

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

References


\section{\omega-semigroups and stabilisation monoids}

\subsection{Ordered and unordered semigroups and monoids}

A semigroup $S = (S, \cdot)$ is a set $S$ equipped with an associative operation $\cdot$. An ordered semigroup $S = (S, \cdot, \leq)$ is a semigroup $(S, \cdot)$ equipped with an order $\leq$ such that $\cdot$ is compatible with $\leq$, i.e., for all $a \leq a'$ and $b \leq b'$, $a \cdot b \leq a' \cdot b'$. A monoid $M = (M, \cdot, 1)$ is a semigroup $(M, \cdot)$ enriched with an element 1 called the unit and such that $1 \cdot x = x = x \cdot 1$ for all $x \in M$. An ordered monoid $M = (M, \cdot, 1, \leq)$ is at the same time a monoid $(M, \cdot, 1)$ and an ordered semigroup $(S, \cdot, \leq)$. An element $e$ in a semigroup is called an idempotent if $e \cdot e = e$. The set of idempotents in a semigroup $S$ is written $E(S)$. A morphism of ordered semigroups (resp. a morphism of ordered monoids) is a morphism of semigroup (resp. of monoid) that preserves the order. Given a word $u = a_1 \ldots a_k \in S^+$ for $S$ a semigroup, we set $\pi(u) = a_1 \cdot a_2 \cdots a_k$. Over a monoid $M$, a word $\pi$ is extended to $u \in M^+$ by $\pi(\varepsilon) = 1$.

A triple $(M, h, F)$ of a monoid, a map from an alphabet $A$ to $M$, and a set $F \subseteq M$ is an algebraic acceptor. It recognises the language $L = \{ u \in A^* \mid \pi(h(u)) \in F \}$, where $h$ is extended into a map from $A^*$ to $M^*$ componentwise (the composition $\pi \circ h$ is the morphism that is usually used). An triple $(M, h, D)$ is an ordered algebraic acceptor if furthermore $M$ is an ordered monoid, and $D$ is downward closed.

\subsection{\omega-semigroup and Wilke algebras}

We expect the reader some familiarities with algebraic recognisers for infinite words (see for instance [14, 17]).

An \omega-semigroup $S = (S, S_\omega, \pi)$ where $\pi$ is a map from $S^+$ to $S$, from $S^* \times S_\omega$ to $S_\omega$ and from $S^*$ to $S_\omega$ at the same times, that satisfies $\pi(a) = a$, $\pi(ab) = a \cdot b$ and $\pi(u_1 \pi(u_2) \ldots) = \pi(u_1 u_2 \ldots)$ (for all meaningful combinations of finite words, infinite words, finite sequence, and infinite sequences).

A Wilke algebra is $W = (S, S_\omega, \omega)$ consists of two sets $S, S_\omega$, a binary operation $\cdot$ that can be used either as $S \times S \rightarrow S$, or as $S \times S_\omega \rightarrow S_\omega$, and a unary map $\omega : S \rightarrow S_\omega$, and such that:

\begin{itemize}
  \item $\cdot$ is associative: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for $a, b \in S$ and $c \in S \cup S_\omega$,
  \item $(a \cdot b)^\omega = a \cdot (b \cdot a)^\omega$ for $a, b \in S$, and
  \item $(a \cdot a)^\omega = a^\omega$.
\end{itemize}

Clearly, given a \omega-semigroup $S = (S, S_\omega, \pi)$, there is a natural induced Wilke algebra defined by $a \cdot b = \pi(ab)$ and $a^\omega = \pi(aa \cdots)$. The key result is that, given a finite Wilke algebra, there exists one, and only one \omega-semigroup that induces it. A \textit{algebraic \omega-language acceptor} $(S, h, F)$ consists of a finite \omega-semigroup/Wilke algebra, a map $h$ from an alphabet $A$ to $S$, and a subset $F \subseteq S_\omega$. The \omega-language is recognised is $\{ u \in S^\omega \mid \pi(h(u)) \in F \}$. It is denoted $L(S, h, F)$.

\subsection{Stabilisation monoids}

The definition of stabilisation monoids is quite complex, and we suggest the reader more complete texts such as [7, 6].

A stabilisation monoid $M = (M, \cdot, \leq, \sharp)$ is a finite ordered monoid $(M, \cdot, 1, \leq)$ together with a stabilisation operation $\sharp$ from $E(M)$, that has the following properties:

\begin{itemize}
  \item for all $a, b$ such that both $a \cdot b$ and $b \cdot a$ are idempotents, $(a \cdot b)^\sharp = a \cdot (b \cdot a)^\sharp \cdot b$,
  \item $e^\sharp \leq f^\sharp$ for all idempotents $e \leq f$,
\end{itemize}
\( e^1 \leq e \) for all idempotents \( e \), and
\( (e^2)^1 = e^2 \) for all idempotents \( e \).

A morphism of stabilisation monoids from a stabilisation monoid to another is an morphism of ordered monoids that preserves the stabilisation.

An algebraic cost function acceptor (ACFA) \((M, h, D)\) consists of a finite stabilisation monoid \(M\), a map \(h\) from the alphabet \(A\) to \(M\), and of a downward closed \(D\) subset of \(M\). A morphism of ACFA from \((M, h, D)\) to \((M', h', D')\) is a morphism of stabilisation monoids \(g\) such that \(g \circ h = h'\) and \(g^{-1}(D') = D\).

We shall now define what is the cost function recognised by a given ACFA. For the purpose of this paper, we provide a minimal number of definitions (these are not all sufficiently general for developing the full theory).

For a natural \(n\), an \(n\)-computation of word \(u \in M^+\) is an unranked ordered rooted tree, the nodes \(x\) of which are labelled by an element \(v(x) \in M\), and such that:

- the labels of the leaves read from left to right yield the word \(u\),
- each node \(x\) of children \(y_1, \ldots, y_k\) is of one of the following forms:
  - leaf \(k = 0\),
  - binary node \(k = 2\), and \(v(x) = v(y_1) \cdot v(y_2)\),
  - idempotent node \(k \in [3, n]\), and \(v(x) = v(y_1) = \cdots = v(y_k)\) is an idempotent,
  - stabilisation node \(k > n\), and \(v(x) = e^\ast\) where \(e = v(y_1) = \cdots = v(y_k)\) is an idempotent.

The value of the \(n\)-computation is the label of its root.

\begin{definition}
A function \(f\) is recognised by an ACFA \((M, h, D)\) if there exists a map \(\alpha : \mathbb{N} \rightarrow \mathbb{N}\) such that for all words \(u\) and all \(m, n\) with \(\alpha(m) \leq n\):

- if \(f(u) \geq n\), then there exists an \(m\)-computation for \(h(u)\) of value in \(D\),
- if there exists an \(n\)-computation for \(h(u)\) of value in \(D\), then \(f(u) \geq m\).

The following lemma will be our only required knowledge on these objects.
\end{definition}

\begin{lemma}
For \(f\) recognised by an ACFA, then \(g\) is recognised by the same ACFA if and only if \(f \approx g\).
\end{lemma}

Hence, we it makes sense to denote as \([M, h, D]_\ast\) the unique cost function recognised by an ACFA \(M, h, D\).

\begin{remark}
This is a simple definition for the notion of computations. In particular, we do not bound the height of the computations, and we do not refine these into under- and over-computations as in [7, 6].
\end{remark}

\begin{remark}
If an ACFA is sent by morphism to another one, then both do recognise the same cost functions.
\end{remark}

### A.4 The \(\omega\)-regular like stabilisation monoids

It turns out that it is good to understand the structure of the stabilisation monoids that recognise \(\omega\)-regular like cost functions. There are four properties that these have to satisfy that are described in the following definition.

\begin{definition}
An ACFA \((M, h, D)\) has the algebraic \(\omega\)-regular like property when
\begin{enumerate}
  \item if \(a \cdot e^\ast \cdot b \cdot f^\ast \cdot c \in D\), then either \(a \cdot e \cdot b \cdot f^\ast \cdot c \in D\) or \(a \cdot e^1 \cdot b \cdot f \cdot c \in D\),
  \item if \(a \cdot (e^\ast \cdot b)^\ast \cdot c \in D\), and \(e \cdot b\) is an idempotent, then either \(a \cdot e^\ast \cdot b \cdot c \in D\), or \(a \cdot (e \cdot b)^\ast \cdot c \in D\),
  \item if \(a \not\in D\) and if \(a \not\in D\) and \(\ell\) is a letter, then \(a \cdot h(\ell) \not\in D\),
  \item if \(a \in D\), then \(a \cdot b \in D\).
\end{enumerate}
\end{definition}
It turns out that the two first conditions correspond to the characterization of the cost functions that are recognised by ‘max-automata’ [8], and more precisely the variant of max-cost-automata that can output an infinite value (for instance, some transitions can be labelled with weight $\infty$). The following condition states that no ‘small word’ can have infinite value, and as a consequence, this corresponds to removing the ability of ma-automata to output $\infty$, i.e., this means going back to the more traditional definition of max-automata. Finally, the last condition stipulates the non-decreasingness: if $u$ is a prefix of $v$, then $f(u) \leq f(v)$ (more precisely, it means that there is a function in the recognised cost function that has this non-decreasingness property.

Lemma 22. Let $g$ be a morphism of ACFA from $(M, h, D)$ to $(M', h', D')$, if $(M, h, D)$ satisfy the algebraic $\omega$-regular like property, then $(M', h', D')$ does too (more precisely, morphisms preserve separately each of the items of the definition of algebraic $\omega$-regular like property).

Lemma 23. Let $f$ be recognised by $(M, h, D)$ that satisfies the algebraic $\omega$-regular like property. There exists $\alpha : \mathbb{N} \to \mathbb{N}$ such that for all $m \in \mathbb{N}$ and all words $u$ such that $f(u) \geq \alpha(m)$, $u$ can be decomposed into $uw_1 \ldots w_nv$, where

- $\pi(u) \notin D$,
- the $w_i$’s are non-empty,
- $\pi(w_1) = \cdots = \pi(w_n) = e$ is an idempotent,
- and $\pi(v) \cdot e^i \in D$.

Proof. For simplifying, we will make use of Lemma 1.22 of [8] (this requires the appendices, that can be found at https://www.irif.univ-paris-diderot.fr/~colcombe/Publications/STACS16-colcombet-kuperberg-manuel-torunczyk.pdf). It states that\(^1\), assuming only the two first items of the definition of the algebraic $\omega$-regular like property,

If $f(u) \geq \alpha(n)$ then
- $\pi(h(w)) \in D$,
- or $w = uw_1 \ldots w_nv$ such that $\pi(h(w_1)) = \cdots = \pi(h(w_n)) = e$ is an idempotent,
- and $\pi(v) \cdot e^i \in D$.

Note first that $\pi(h(w)) \notin D$ is obtained directly from the third item of the algebraic $\omega$-regular like property. This immediately means that first situation described in the above statement is impossible. Hence, $w$ can be decomposed as in the second item. And for the same consideration, $\pi(h(u)) \notin D$. Now, let us assume that $\pi(h(u)) \cdot e^i \notin D$, then again, by the third item, we would obtain $\pi(h(u)) \cdot e^i \cdot \pi(h(v)) \notin D$. A contradiction. It follows that $\pi(h(u)) \cdot e^i \in D$. Finally, assume that $w_i$ would be empty for some $i$. This would mean that $e = 1$. But since $1^2 = 1$ (in every stabilisation monoid), we would get $\pi(h(u)) \cdot e^i = \pi(h(u)) \in D$. A contradiction. Hence all the $w_i$’s are non-empty. ▶

Corollary 24. If $f$ is recognised by $(M, h, D)$ that satisfies the algebraic $\omega$-regular like property, then $f$ is $\omega$-regular like.

Proof. Let $L$ be the $\omega$-language defined as:

$$L = \{uv_0v_1 \cdots | \pi(h(u)) \cdot e^i \in D, \pi(h(v_i)) = e \text{ for all } i \in \mathbb{N} \text{ and } e \in E(M)\}.$$ 

\(^1\) After three minor modifications: (a) notations are changed and in particular $h$ there becomes $\pi \circ h$ here, (b) the input being $f$, a function $\alpha$ has been added, and (c) the $\omega$ symbol (that we did not introduce in this paper) has been removed, becoming $e$ idempotent, and yielding a use of Ramsey that is ‘absorbed’ in $\alpha$.\footnote{After three minor modifications: (a) notations are changed and in particular $h$ there becomes $\pi \circ h$ here, (b) the input being $f$, a function $\alpha$ has been added, and (c) the $\omega$ symbol (that we did not introduce in this paper) has been removed, becoming $e$ idempotent, and yielding a use of Ramsey that is ‘absorbed’ in $\alpha$.}
Let us assume $f(w)$ is large, then $w = uv_1 \ldots v_n$ such that $\pi(h(u)) \cdot e^\omega \in D$, where $\pi(h(v_1)) = \cdots = e$. Thus, $L^{\omega 1}(w) \geq n$. Conversely, assume that $L^{\omega 2}(w) \geq \alpha(n)$ (for a sufficiently large value of $\alpha(n)$), this means that $w = uv_1 \ldots v_n$ with $u(v_1, \ldots, v_n) \omega \subseteq L$. By Ramsey, we can choose, $\pi(v_1) = \cdots = \pi(v_n) = e$ to be an idempotent.

**A.5 The construction from $\omega$-semigroup to stabilisation monoids**

In this section, we show how to convert an algebraic $\omega$-language acceptor $(W, h, F)$ recognising $L$ into an algebraic cost function acceptor $(W^{\omega 1}, h_{\omega 1}, F_{\omega 1})$ (Lemma 25) that recognises a cost function $f$, and show that $f \approx L^{\omega 1}$ (Lemma 27) and that $(W^{\omega 1}, h_{\omega 1}, F_{\omega 1})$ satisfy the algebraic $\omega$-regular like property (Lemma 26).

Hence, let us fix $(W, h, F)$ and construct $W^{\omega 1} = (M, 1_{\omega 1}, \cdot, \leq, \sharp)$ as follows. The elements are

$$M = S \times \mathcal{P}(S_{\omega}) \cup \{1_{\omega 1}\},$$

where $1_{\omega 1}$ is a new element. The new product $\cdot_{\omega 1}$ has $1_{\omega 1}$ as unit and is defined elsewhere by

$$(a, A) \cdot_{\omega 1} (b, B) = (a \cdot b, A \cup b \cdot B).$$

Now idempotents (other that $1_{\omega 1}$) for $\cdot_{\omega 1}$ are the elements of the form $(c, E) \in M$ such that $c$ is idempotent, and $c \cdot E \subseteq E$. For such an element, one defines

$$(e, E)^2 = (e, E \cup \{e^\omega\}).$$

It remains to define the order as

$$(a, A) \leq_{\omega 1} (b, B) \quad \text{if} \quad a = b \quad \text{and} \quad B \subseteq A.$$

Let us now establish that this definition is meaningful.

**Lemma 25.** $W^{\omega 1}$ is a stabilisation monoid.

**Proof.** This is syntactic verification. By definition, $1_{\omega 1}$ is a unit for $\cdot_{\omega 1}$. It is also obvious that $\cdot_{\omega 1}$ is associative; indeed $((a, A) \cdot_{\omega 1} (b, B)) \cdot_{\omega 1} (c, C) = (a \cdot b \cdot c, A \cup a \cdot B \cup b \cdot C) = (a, A) \cdot_{\omega 1} (b, B) \cdot_{\omega 1} (c, C))$. The compatibility of $\cdot_{\omega 1}$ with respect to $\leq_{\omega 1}$ is obvious. This means that $(M, 1_{\omega 1}, \cdot, \leq_{\omega 1})$ is an ordered monoid. Let us now prove the correctness of the choice of the stabilisation $\sharp$. Note first that every property to be checked becomes obvious when $1_{\omega 1}$ is used as one of the parameters. We treat below the remaining cases.

- Clearly we have $(e, E)^2 = (e, E \cup e^\omega) \leq_{\omega 1} (e, E)$.
- It is also clear that $(e, E)^3 = (e, E)^2 = (e, E)^2).
- Also $(e, E)^3 = (e, E \cup \{e^\omega\} \leq (e, E)^2).

Let now $(a, A)$ and $(b, B)$ be such that both $(a, A) \cdot_{\omega 1} (b, B)$ and $(b, B) \cdot_{\omega 1} (a, A)$ are idempotents. Then

$$(a, A) \cdot_{\omega 1} (b, B))^2 = (a \cdot b, A \cup a \cdot B)^2$$

$$= ((a \cdot b) \cdot (a \cdot b), A \cup a \cdot B \cup \{a \cdot (b \cdot a)^\omega\})$$

$$= (a \cdot (b \cdot a) \cdot b, A \cup a \cdot B \cup \{a \cdot (b \cdot a)^\omega\})$$

$$= (a \cdot (b \cdot a) \cdot b, A \cup a \cdot (B \cup b \cdot A \cup \{a \cdot (b \cdot a)^\omega\}) \cup a \cdot b \cdot a \cdot B)$$

$$= (a, A) \cdot_{\omega 1} ((b, B) \cdot_{\omega 1} (a, A))^2 \cdot_{\omega 1} (b, B).$$

Hence $W^{\omega 1}$ is a stabilisation monoid.
We shall now also define $h_{o_1}$ and $F_{o_1}$ in order to turn $(W^{o_1}, h_{o_1}, F_{o_1})$ into an algebraic cost function acceptor with

$$h_{o_1}(\ell) = (h(\ell), \emptyset) \quad \text{for all letters } \ell,$$

and $F_{o_1} = \{(a, A) | a \cdot x \in F, \text{ for } x \in A\}$.

Note that $F_{o_1}$ is clearly downward closed for $\leq_{o_1}$.

**Lemma 26.** $(W^{o_1}, h_{o_1}, F_{o_1})$ satisfy the algebraic $\omega$-regular like property.

**Proof.** As above, we do not treat the obvious cases involving $1_{o_1}$.

First item. Assume $(a, A) \cdot_{o_1} (e, E)^1 \cdot_{o_1} (b, B) \cdot_{o_1} (f, F)^2 \cdot_{o_1} (c, C) \in F_{o_1}$. This means $x \in A \cup a \cdot (E \cup \{e^\omega\}) \cup a \cdot E \cup a \cdot b \cdot (F \cup \{f^2\}) \cup a \cdot e \cdot b \cdot f \cdot C$ for some $x \in F$. There are essentially three cases. Case 1: $x = A \cup a \cdot E \cup a \cdot e \cdot B \cup a \cdot e \cdot b \cdot f \cup a \cdot e \cdot b \cdot f \cdot C$. This means that $x \in A \cup a \cdot E \cup a \cdot e \cdot B \cup a \cdot e \cdot b \cdot f \cup a \cdot e \cdot b \cdot f \cdot C$. Note that $x$ can be decomposed as $x = a \cdot e \cdot b \cdot f^2$. In this case, $(a, A) \cdot_{o_1} (e, E)^1 \cdot_{o_1} (b, B) \cdot_{o_1} (f, F)^2 \cdot_{o_1} (c, C) \in F_{o_1}$.

Second item. Assume $(a, A) \cdot_{o_1} ((e, E)^2 \cdot_{o_1} (b, B)) \cdot_{o_1} (c, C) \in F_{o_1}$. This means that $x \in A \cup a \cdot ((E \cup \{e^\omega\}) \cup e \cdot B \cup ((e \cdot b)^2) \cup a \cdot e \cdot c \cdot C$ for some $x \in F$. Once more three cases can happen. The interesting ones are $x = a \cdot e \cdot b \cdot f^2$. In this case, $x$ is a witness that $(a, A) \cdot_{o_1} ((e, E)^2 \cdot_{o_1} (b, B)) \cdot_{o_1} (c, C) \in F_{o_1}$. The second one is $x = a \cdot e \cdot b \cdot f^2$. Then $x$ is a witness that $(a, A) \cdot_{o_1} ((e, E)^2 \cdot_{o_1} (b, B)) \cdot_{o_1} (c, C) \in F_{o_1}$.

Third item. Clearly $1_{o_1} \notin F_{o_1}$. Note now for all letters $\ell$, $1_{o_1} \cdot_{o_1} (\ell) = h_{o_1}(\ell) = (h(\ell), \emptyset) \notin F_{o_1}$. Furthermore, if $(a, A) \notin F_{o_1}$, this means $A \cap F = \emptyset$. Compute now $(a, A) \cdot_{o_1} (\ell) = (a \cdot h(\ell), A)$ which does not belong to $F_{o_1}$ for the same reason.

Fourth item. Assume $(a, A) \in F_{o_1}$, this means that there is $x \in A \cap F$. Since $(a, A) \cdot_{o_1} (b, B) = (a \cdot b, A \cup a \cdot B)$, the same $x$ is also a witness that $(a, A) \cdot_{o_1} (b, B) \in F_{o_1}$.

In the end, we can prove the correctness of the construction.

**Lemma 27.** $\mathcal{L}(W, h, F)^{o_1} \approx [W^{o_1}, h_{o_1}, F_{o_1}]_2$.

**Proof.** Let $f \in [W^{o_1}, h_{o_1}, F_{o_1}]_2$. Assume that $f(u) \geq \alpha(n)$ for $\alpha$ from Lemma 23 (we can use it according to Lemma 26). This means that $u$ is decomposed into $u = w_1 \ldots w_n v'$ such that $\pi(h_{o_1}(w_i)) \leq (e, E)$ for all $i$, for some idempotent $(e, E)$, and $\pi(h(v)) \cdot_{o_1} (e, E)^2 \in D$. But, $\pi(h_{o_1}(w_1)), \pi(h_{o_1}(w_2)), \ldots, \pi(h_{o_1}(w_n))$ all have $\emptyset$ as second component. This means that $\pi(h(v)) \cdot e^\omega \in F$. Hence $v\{u_1, \ldots, w_n\} \subseteq \mathcal{L}(W, h, F)$. Thus $\mathcal{L}(W, h, F)^{o_1} \geq n$.

Conversely, assume that $\mathcal{L}(W, h, F)^{o_1} \geq \alpha(n)$ for $\alpha(n)$ sufficiently large in front of $n$ (see below). This means that $u = v w_1 \ldots w_m v'$ with $v\{w_1, \ldots, w_m\} \subseteq \mathcal{L}(W, h, F)$. By factoring the $w_i$’s together using Ramsey’s theorem (we assume $\alpha(n)$ has been chosen sufficiently large), one can find a factorisation $u = v' w'_1 \ldots w'_m s'$ with the same properties and such that furthermore $\pi(h(w'_i)) = e$ for all $i = 1 \ldots n$ where $e$ is some idempotent. Since $v'\{w'_1, \ldots, w'_m\} \subseteq \mathcal{L}(W, h, F), \pi(h(v')) \cdot e^\omega \in F$. Hence, one can construct an $n$-computation with as root value $\pi(h(v')) \cdot (e, \emptyset)^2$, which belongs to $F_{o_1}$ using $x = \pi(h(v')) \cdot e^\omega \in F$ as witness. Thus, $[W^{o_1}, h_{o_1}, F_{o_1}]_2 \geq n$.

Overall $\mathcal{L}(W, h, F)^{o_1} \approx [W^{o_1}, h_{o_1}, F_{o_1}]_2$. 

}\hfill\blacktriangle