Uniformization Results on Regular Cost Functions

Thomas Colcombet[1], Stefan Göller[1,2], Amaldev Manuel[1]

[1] LIAFA, Université Paris Diderot - Paris 7,
{thomas.colcombet, amal}@liafa.jussieu.fr
[2] University of Bremen, goeller@informatik.uni-bremen.de

Abstract. B-automata are automata that can compute functions from words to non-negative integers or infinity. In this paper we give another semantics to the hierarchical variant of these automata. This new semantics is equivalent to the classical one in the terminology of cost functions: functions computed are equivalent up to a polynomial factor. We provide three applications of this technique.
In our first application we provide a deterministic streaming algorithm (it reads an input word from left to right) for evaluating a fixed B-automaton which uses logarithmic memory only. A similar evaluation would require polynomial memory when using the original semantics.
Our second theorem shows that for games with hB-quantitative objectives, there are memoryless strategies that are uniformly optimal, i.e., optimal from any starting point.
Finally, we introduce a new form of history-determinism that is uniform in the sense that translation strategies are independent from bounds. While it is known that uniform history-determinism cannot be enforced for usual B-automata, we disclose new forms of (equivalent) automata that have this property.

1 Introduction

Regular cost functions provide a quantitative extension to the notion of regular languages [9, 11]. One way to understand it is as a framework in which classical qualitative questions such as "Is it true that all inputs satisfy some property?" have a natural quantitative counterpart, namely "Is it true that for some n all inputs satisfy some property, within a resource constraint of n?". Cost functions provide several formalisms for describing such resource constrained properties. A typical such model is the one of B-automata, where a specific action executed during a transition can consume some resource, and another action can refill the resource. Another example is cost monadic logic, which is a logic formalism extending monadic second order logic, that can furthermore constrain the cardinality of the quantified sets.

The specificity of cost functions compared to other quantitative notions of regularity is that the functions are considered modulo a "mutual boundedness equivalence" (denoted $\approx$). Two functions are *$\approx$-equivalent* if and only if they are bounded over the same sets of inputs. To understand the interest of using such an equivalence relation, let us recall Krob's undecidability result [20]: the problem of deciding whether two functions recognized by distance automata (a subcase of B-automata) are equal is undecidable. This means that the above models seem difficult to effectively manipulate while keeping all the precision. Considering functions modulo "mutual boundedness $\approx$" allows to circumvent this difficulty, ant the equivalence of two B-automata up to $\approx$ turns out to be a decidable problem.

An interesting aspect of the theory of regular cost functions is that it provides results that are, in many ways, as robust as the theory of regular languages but extend them. In particular such models have been used for solving the star-height problem for words [16, 19] and for trees [12], the finite substitution problem [1, 18] or the boundedness of fixpoint over words and trees [2, 3]. Deciding the level in the nondeterministic Mostowski hierarchy – a problem that is still open in general – of a language of infinite trees also reduces to such questions [13, **?**].

B-automata are nondeterministic finite state automata that possess several counters. At each transition, the automaton can leave a counter unchanged, increment it by one, or reset it to zero. The value computed by the automaton is the minimum over all accepting runs of the maximum value assumed by a counter during the run. Its hierarchical variant, hierarchical B-automata (hB-automata), furthermore constrain the use of counters with a nesting policy. *Hierarchical B-automata* play a particularly important role as the hB-condition is particularly well-behaved, comparable to how parity condition is related to Muller condition: both are equally expressive but the former is usually simpler to handle than the latter.

Contributions. The question we address in this work is the one of *uniformization*. In the following we informally state what we mean by that. The semantics of B-automata can be stated in the following way. One first chooses a minimal n and tests whether the input can be accepted by a run in which none of the counters exceed the value n. By this definition, for knowing the cost of an input, one first needs to guess the correct n and only then run the automaton. This way to model semantics is in fact shared by all models of computation involved in the theory of cost-functions, be it logical, algebraic, automata- or game-theoretic. However this is inconvenient in several situations, for instance to efficiently compute the value of an automaton running over an input.

By *uniformization* we mean the generalizations of known results to situations when the bound n cannot be guessed advance. In this paper we instantiate this approach in three different ways. The heart of the contribution is a new definition of the value computed by an hB-automaton. This value is different from the usual semantics, but is equivalent to it up to "mutual boundedness $\approx$". This makes this new semantics as good as the original one as far as cost functions are concerned. Based on this new semantics, we present three new results.

Efficient deterministic streaming algorithm for evaluation. For each hB-automaton (that we assume as fixed) that computes a function f we concern ourselves with the question of computing the value $f(u)$ for an input word u. We furthermore want the algorithm to be deterministic and "streaming" in the sense that it reads the input word only once from left to right. We show that with the usual semantics of $(h)B$-automata, such an algorithm requires memory polynomial in $|u|$, but with the new semantics, logarithmic space suffices.

Uniform memoryless strategies in hB-games. Memoryless determinacy (*a.k.a.* the existence of memoryless strategies in determined games) is an important notion in automata theory. It means that there is no need to possess any information for playing optimally in a game of a given objective. It is known that for the hB-winning condition, and up to "mutual boundedness", we have for each n that if the first player wins the condition $W_n = \{u \mid \text{no counter exceeds the value } n \text{ when executing } u\}$, she achieves this by using a memoryless strategy [12]. We show a uniform version of this result which can be stated as follows. In every hB-game, there exists a memoryless strategy such that whatever is the chosen initial position, if the first player can win the game with condition W_n for some n then the memoryless strategy is winning for W_n. Hence, we are able to synthesize a uniform strategy that is not tied to a specific resource bound. To this end, we prove a generic proposition for proving memoryless determinacy; it can be seen to unify standard techniques for proving determinacy and can be of independent interest.

Uniform history-deterministic automata. History-deterministic automata are an important tool in the context of games and are automata that compose correctly with games (these are also known as "good for solving games" [17] in the Boolean setting). This notion was not so much investigated in the Boolean setting because every language is recognized by a deterministic automaton, which is an even stronger notion. The situation is different in the context of cost functions where automata cannot be made deterministic, even up to "mutual boundedness", but can be made history-deterministic [9]. For this reason, history-determinism plays a central role in the proofs concerning cost-functions over trees [14, 21].

We are interested here in the natural uniform version of history-determinism. We show that hB-automata cannot be made uniformly history-deterministic. Thus we consider an extension of them, $\max$-*automata*. These have the same expressive power as hB-automata, and further can be transformed into uniformly history-deterministic $\max$-automata.

Related work. The theory of regular cost functions is of course related to weighted automata, and more specifically automata weighted over the tropical semiring [22, 23]. However, the viewpoint of considering functions up to "mutual boundedness" makes all the results of very different nature. This notion of mutual boundedness takes its root in the limitedness (which is a variant of boundedness) results of Hashiguchi [15]. Distance automata were extended to nested distance desert automata (i.e., hB-automata here) by Kirsten in [19]. These were combined with works on the logic MSO+$\mathbb{U}$ over infinite words [5] to yield the theory of regular cost functions, first for finite words in [11], then for finite trees [14], and for infinite trees for weak logics [8, 21].

1.1 Structure of the paper

The remaining of this paper is organized as follows.

2 Preliminaries

In this section we successively present the notion of cost functions, B-automata, and hB-automata that will be used throughout the rest of this paper.

As usual $\mathbb{Z}$ is the set of integers and $\mathbb{N}$ is the set of non-negative integers. For each $i, j \in \mathbb{Z}$ we denote by $[i, j]$ the set $\{i, i+1, \ldots, j\}$ if $i \leq j$ and $\emptyset$ otherwise. For any set X and any $\tau = (x_1, \ldots, x_k) \in X^k$, we set $\pi_i(\tau) = x_i$ the projection of τ to the i^{th} component of τ, for each $i \in [1, k]$. The latter is extended to a homomorphism π_i from $(X^k)^*$ to X in the usual way. For all $x \in X$, we denote by $\overline{x}^k$ the k-tuple $(x, \ldots, x)$. We set $\mathbb{N}_\infty$ to be $\mathbb{N} \cup \{\infty\}$. We take the convention that $\inf \emptyset = \min \emptyset = \infty$ and $\sup \emptyset = \max \emptyset = 0$. Given two sets A and B we let B^A denote the set of all functions from A to B.

Let Σ be a finite alphabet. The set of words over Σ is Σ^*. The length of a word w is $|w|$.

Cost functions. Let us fix ourselves a set X (that will be in practice Σ^* for some finite alphabet Σ). Our object of interest are functions from X to $\mathbb{N}_\infty$. Given two such functions f, g, we say that g *dominates* f, denoted $f \preccurlyeq g$, if all $Y \subseteq X$, if g is bounded over Y (meaning $\sup(g(Y)) < \infty$), then f is also bounded over Y. The functions f and g are *cost-equivalent*, in notation $f \approx g$, if both $f \preccurlyeq g$ and $g \preccurlyeq f$.

Another presentation, equivalent, of this relation is as follows. A *correction function* is an increasing function $\alpha : \mathbb{N} \to \mathbb{N}$. We extend α to the domain $\mathbb{N}_\infty$ by setting $\alpha(\infty) = \infty$. For any set X, and given two functions f, g from X to $\mathbb{N}_\infty$, we write $f \preccurlyeq_\alpha g$ whenever $f(x) \leq \alpha(g(x))$ for each $x \in X$. We write $f \approx_\alpha g$ in case $f \preccurlyeq_\alpha g$ and $g \preccurlyeq_\alpha f$. It is an easy exercice to show that $f \preccurlyeq g$ if and only $f \preccurlyeq_\alpha g$ for some correction function α (see for instance [9]).

A *cost function* (over X) is an equivalence class of $\approx$ over $(\mathbb{N}_\infty)^X$. We denote by $[f]$ the equivalence class of $f \in (\mathbb{N}_\infty)^X$.

Example 1. Let X be the set $\mathbb{N} \times \mathbb{N}$. Consider the following functions from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$, $+ : x, y \mapsto (x+y)$ and $\max : x, y \mapsto \max(x, y)$. These two functions are cost-equivalent since $\max(x, y) \leq x + y \leq 2 \cdot \max(x, y)$. Whereas the functions $\max$ and $\min : x, y \to \min(x, y)$ are *not* cost-equivalent since on the set of pairs $\{(1,0), (2,0), \ldots\}$ the function $\min$ is bounded but not the function $\max$.

Example 2. Let $\Sigma = \{0, 1\}$. Consider the function $b : \Sigma^* \to \mathbb{N}$ such that $b(w) = n$ if w is a binary representation of n (possibly with a trail of zeros on the left). For instance $b(0011010) = 26$. Let $f : \Sigma^* \to \mathbb{N}$ be the function defined as

$$f(w) = |w| - |u|, \text{ where } u \text{ is the largest prefix of } w \text{ which is in } 0^*.$$

The functions b and f are cost-equivalent. To see this, consider the correction function $\alpha(i) = 2^i$. It is easy to show that $f(w) \leq |w| \leq b(w) \leq \alpha(f(w))$.

Regular cost functions. A class of cost functions which form the analog of regular languages over words is the class of regular cost functions. Like regularity on words this class also enjoys many good properties; they are closed under the operations $\min$ and $\max$ (which are the analogue of intersection and union) and projections. They are also defined by a variety of equivalent formalisms; cost monadic second order logic, stabilization monoids, B-automata, S-automata (and their hierarchical versions) and cost regular expressions [11]. The most interesting aspect, however, is that the boundedness (given f, is $f \preccurlyeq \mathbf{0}$? where $\mathbf{0}$ is the constant zero function) and even domination (given f and g, is $f \preccurlyeq g$?) problems are decidable for regular cost functions. In the following we introduce B-automata and its hierarchical form.

B-automata. A *B-automaton* is a nondeterministic finite state automaton extended with a finite set of counters $\Gamma = \{1, \ldots, k\}$ with counter operations `ic` (increments the counter), `r` (resets the counter) and ε (does nothing). Formally it is a tuple $\mathcal{B} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$, where Q is a finite set of states, Σ is the input alphabet, $\Delta \subseteq Q \times \Sigma \times \{\texttt{ic}, \texttt{r}, \varepsilon\}^{\Gamma} \times Q$ is the set of transitions, $q_I \in Q$ is the initial state and $F \subseteq Q$ is the set of final states.

Initially all the counters of the automaton are set to zero. A *run* ρ on a word $w = a_1 \cdots a_n$ is a sequence of transitions $(q_0, a_1, \sigma_1, q_1)(q_1, a_2, \sigma_2, q_2) \ldots (q_{n_1}, a_n, \sigma_n, q_n) \in \Delta^*$. A run is *successful* if it ends in a final state. For every counter $i \in \Gamma$ the run ρ defines a sequence of counter operations $\sigma_1 \cdots \sigma_n \in \{\texttt{ic}, \texttt{r}, \varepsilon\}^*$ and in turn a sequence of values $c_1, \ldots, c_n$. Let $\mathsf{val}_i(\rho)$ be the set $\{c_1, \ldots, c_n\}$ for each $i \in \Gamma$ and let $\mathsf{val}(\rho)$ be the set $\bigcup_{i \in \Gamma} \mathsf{val}_i(\rho)$.

The *cost* of the run ρ is defined as $\mathsf{cost}(\rho) = \sup \mathsf{val}(\rho)$. The *value* of $\mathcal{B}$ on w is defined as $[\![\mathcal{B}]\!](w) = \inf\{\mathsf{cost}(\rho) \mid \rho \text{ is a successful run of } \mathcal{B} \text{ on } w\}$.

Example 3. Let $\Sigma = \{a, b\}$. We note that each word in Σ^* has a unique decomposition of the form $a^{n_0} b a^{n_1} \cdots b a^{n_k}$, where $n_i \in \mathbb{N}$ for each $i \in [0, k]$, possibly $k = 0$. The cost function $\mathsf{SL} : \Sigma^* \to \mathbb{N}_\infty$, given such a word, computes the *second largest* number in the sequence $(n_i)_{i \in [0,k]}$. Note that $\mathsf{SL}(w)$ can be the largest number also, if it occurs twice or more. Formally $\mathsf{SL}\,(a^{n_0} b a^{n_1} \cdots b a^{n_k})$ is $\max\{\min\{n_i, n_j\} \mid i, j \in [0, k] : i \neq j\}$. Note that according to this definition, the value is 0 when $k = 0$.

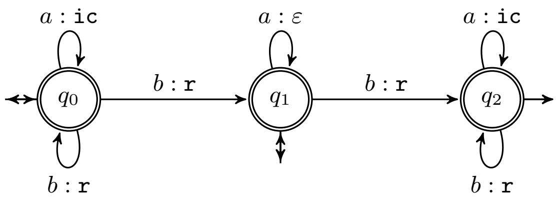

Fig. 1. B-automaton in the Example 3.

The function SL is computed by a B-automaton (shown in Figure 1) with one counter c in the following way. We refer to the maximal consecutive a-positions as a-blocks. While reading the word w the automaton increments c on all a-blocks except for a nondeterministically chosen possibly empty a-block and resets c on all b positions. Hence for a given run ρ, the value of the run corresponds to the maximum of the lengths of all a-blocks except the excluded a-block. Since the cost $\mathsf{SL}(w)$ is the minimum among values of all runs, it is obtained when the largest a-block is excluded which in turn corresponds to the length of the second largest a-block in the word w.

Hierarchical B-automata. A *hierarchical B-automaton* (*hB-automaton* for short) is a B-automaton in which a stack-like discipline is imposed on the counters, i.e. whenever a counter γ_i with $i > 1$ is incremented or reset, the counters $\gamma_1, \ldots, \gamma_{i-1}$ are reset. With this restriction the possible counter operations reduces to the set $A_{\Gamma_h} = \{\texttt{R}_0, \texttt{IC}_1, \texttt{R}_1, \ldots, \texttt{IC}_k, \texttt{R}_k\}$ where $\texttt{R}_i$ resets all counters up to i (as a pathological case, $\texttt{R}_0$ has no effect), and $\texttt{IC}_i$ increments counter i and resets all counters up to $i - 1$. Hence the set of transitions of a hierarchical B-automaton is of the form $\Delta \subseteq Q \times \Sigma \times A_{\Gamma_h} \times Q$.

We now formally define the semantics of counter operations. For each $\sigma \in A_{\Gamma_h}$ and each $v = (n_1, \ldots, n_k) \in \mathbb{N}^k$ the operation $\mathsf{val}(\sigma, v)$ is defined as follows:

$$\mathsf{val}(\sigma, v) = \begin{cases} (0, \ldots, 0, n_{j+1}, \ldots, n_k) & \text{if } \sigma = \texttt{R}_j \\ (0, \ldots, 0, n_j + 1, n_{j+1}, \ldots, n_k) & \text{if } \sigma = \texttt{IC}_j \end{cases}$$

The operation val is extended to $A^*_{\Gamma_h}$ inductively by $\mathsf{val}(\varepsilon, v) = v$ and $\mathsf{val}(u\sigma, v) = \mathsf{val}(\sigma, \mathsf{val}(u, v))$ for each $u \in A^*_{\Gamma_h}$, each $\sigma \in A_{\Gamma_h}$ and each $v \in \mathbb{N}^k$. For any $u \in A^*_{\Gamma_h}$ and any $v \in \mathbb{N}^k$ we define

$$\mathsf{cost}_{hB}(u, v) = \max\{\pi_i(\mathsf{val}(u', v)) \mid i \in [1, k], u' \text{ is a prefix of } u\}$$

and $\mathsf{cost}_{hB}(u) = \mathsf{cost}_{hB}(u, \bar{0}^k)$.

As above, a run ρ defines a sequence of counter operations $u = \sigma_1 \cdots \sigma_n \in A^*_{\Gamma_h}$. We define the cost of the run ρ as $\mathsf{cost}(\rho) = \mathsf{cost}_{hB}(u)$ and the cost of the word w is defined as $[\![\mathcal{B}]\!](w) = \inf\{\mathsf{cost}_{hB}(\rho) \mid \rho \text{ is a successful run of } \mathcal{B} \text{ on } w\}$.

Every B-automaton is equivalent to an hB-automaton up to $\approx$ (consult [9] and [14] for an efficient construction).

3 A cost-equivalent semantics for counter actions

The core contribution of the paper is to introduce a variant of the semantics of hB-automata. This new semantics evaluates a sequence of actions on the counters differently. We will prove this approach to be correct in the sense that this change has no effect as far as cost functions are concerned, i.e., the values of automata using the original semantics are $\approx$-equivalent to the ones with the new semantics. This idea extends methods used in [12]. The notions related to this new semantics are annotated with *.

The new semantics for hB-automata. From now, k is the number of counters of some hB-automaton. We set $V = \mathbb{N}^{k+1}$ to be the set of all $(k+1)$-tuples $(n_1, \ldots, n_k, N)$ over $\mathbb{N}$ satisfying $n_i \leq N$ for each $i \in [1, k]$. The informal idea is that the components from 1 to k will take care of the counters from 1 to k, and that the component $k+1$ carries the highest value seen so far. However, the way the counters are updated is not as we defined above for the hB-condition, but follows a technique of carry propagation as introduced in [12]. Let us define how a tuple of V evolves when one of the actions from A_{Γ_h} is encountered.

The $\mathtt{R}_j$ action is similar to the classical evaluation of the hB-semantics. The idea behind the action $\mathtt{IC}_j$ is that the jth counter is incremented, and, if it exceeds N, then it is reset, and the counter $j+1$ is incremented, thus this behavior is propagated to higher counters. It may happen that this carry propagation reaches the component $k+1$ to exceed N, which gets incremented in this case, but propagation stops. This is the sole case that can entail an increase of N. We define the semantics formally below.

For each $\sigma \in A_{\Gamma_h}$ and each $v = (n_1, \ldots, n_k, N) \in V$ the operation $\mathsf{val}^*(\sigma, v)$ is defined as follows:

$$\mathsf{val}^*(\sigma, v) = \begin{cases} v & \text{if } \sigma = \mathtt{R}_0 \\ (\overline{0}^i, n_{i+1}, \ldots, n_k, N) & \text{if } \sigma = \mathtt{R}_i \text{ for some } i = 1 \ldots k \\ (\overline{0}^\ell, n_\ell + 1, n_{\ell+1}, \ldots, n_k, N) & \text{if } \sigma = \mathtt{IC}_i \text{ for some } i = 1 \ldots k, \\ & \quad n_i = \ldots = n_{\ell-1} = N \text{ and } n_\ell < N \text{ where } \ell \in [i+1, k] \\ (\overline{0}^k, N+1) & \text{if } \sigma = \mathtt{IC}_i \text{ for some } i = 1 \ldots k, \text{ and } n_i = \ldots = n_k = N \end{cases}$$

Example 4. Let us assume $k = 2$.Then,

$$(0,0,0) \xrightarrow{\mathtt{IC}_1} (0,0,1) \xrightarrow{\mathtt{IC}_1} (1,0,1) \xrightarrow{\mathtt{IC}_1} (0,1,1) \xrightarrow{\mathtt{IC}_2} (0,0,2).$$

The definition of val^* is extended to the domain $A^*_{\Gamma_h} \times V$ inductively as $\mathsf{val}^*(\varepsilon, v) = v$ and $\mathsf{val}^*(u\sigma, v) = \mathsf{val}^*(\sigma, \mathsf{val}^*(u, v))$ for each $u \in A^*_{\Gamma_h}$, $\sigma \in A_{\Gamma_h}$ and $v \in V$.

An important reason why this new way to evaluate runs has good properties is that it satisfies certain monotonicity properties. The order involved there is the reverse lexicographic ordering, the definition of which we recall now. For $v = (n_1, \ldots, n_k, N)$ and $v' = (n'_1, \ldots, n'_k, N')$ two vectors from V we write $v <_{\text{rlex}} v'$ if either

- $N < N'$ or
- $N = N'$ and for some i we have $n_i < n'_i$ and $n_j = n'_j$ for all $j \in [i+1, k]$.

We set now $v \leq_{\text{rlex}} v'$ to hold if either $v = v'$ or $v <_{\text{rlex}} v'$. The monotonicity property we mentioned above reads as follows.

Lemma 5 (monotonicity). *Let $v, v' \in V$ and $u \in A^*_{\Gamma_h}$. Then $v \leq_{rlex} v'$ implies $\mathsf{val}^*(u, v) \leq_{rlex} \mathsf{val}^*(u, v')$.*

Proof. The proof is by induction on the length of u and it is enough to verify the claim for each σ in A_{Γ_h}. Let us consider some $v = (n_1, \ldots, n_k, N) \leq_{\text{rlex}} v' = (n'_1, \ldots, n'_k, N')$. We make a case distinction on σ. *Case* $\sigma = \mathtt{R}_i$ is immediate. Indeed,

$$\mathsf{val}^*(\mathtt{R}_i, v) = (0, \ldots, 0, n_{i+1}, \ldots, n_k, N) \leq_{\text{rlex}} (0, \ldots, 0, n'_{i+1}, \ldots, n'_k, N') = \mathsf{val}^*(\mathtt{R}_i, v').$$

Case $\sigma = \mathtt{IC}_i$. Assume first that $N = N' = b$. Then, the action $\mathtt{IC}_i$ can be seen as performing the action $\mathtt{R}_{i-1}$ and then adding 1 to the i^{th} component of v (resp. v') seen as integer written in base $b+1$ (the latter means adding $(b+1)^i$ to it). Of course this operation is also monotonic. Hence $\mathsf{val}^*(\sigma, v) \leq_{\text{rlex}} \mathsf{val}^*(\sigma, v')$.

Otherwise we have $N < N'$. In this case, remark that the component N' can only increase by performing action $\mathtt{IC}_i$ and the result of performing action $\mathtt{IC}_i$ on v is that

- either N is unchanged and we have $\mathsf{val}^*(\sigma, v) <_{\text{rlex}} \mathsf{val}^*(\sigma, v')$ by just looking at the $(k+1)^{\text{th}}$ component,
- or N is incremented and in this case $n_1, \ldots, n_k$ are all reset and once more it is the case that $\mathsf{val}^*(\sigma, v) \leq_{\text{rlex}} \mathsf{val}^*(\sigma, v')$.

This establishes the inductive step. □

When evaluating a run using the new semantics, only the last component matters. This is formalized with the definition $\mathsf{cost}^*_{hB}(u, v) = \pi_{k+1}(\mathsf{val}^*(u, v))$ and $\mathsf{cost}^*_{hB}(u) = \mathsf{cost}^*_{hB}(u, \bar{0}^{k+1})$. A second important fact is that our new semantics is equivalent to the standard one. A consequence of the monotonicity lemma reads as follows.

Corollary 6. *Let $x, v, y \in A^*_{\Gamma_h}$ then $\mathsf{cost}^*_{hB}(v) \leq \mathsf{cost}^*_{hB}(xvy)$.*

Proof. Let $u = xvy$. First observe that due to $\bar{0}^{k+1} \leq_{\text{rlex}} \mathsf{val}^*_{hB}(x, \bar{0}^{k+1})$ it follows $\mathsf{val}^*_{hB}(v, \bar{0}^{k+1}) \leq_{\text{rlex}} \mathsf{val}^*_{hB}(v, \mathsf{val}^*_{hB}(x, \bar{0}^{k+1}))$ by Lemma 5. The latter reads as $\mathsf{val}^*_{hB}(v, \bar{0}^{k+1}) \leq_{\text{rlex}} \mathsf{val}^*_{hB}(xv, \bar{0}^{k+1})$. It follows that $\mathsf{cost}^*_{hB}(v) \leq \mathsf{cost}^*_{hB}(xv)$. Clearly $\mathsf{cost}^*_{hB}(xv) \leq \mathsf{cost}^*_{hB}(xvy) = \mathsf{cost}^*_{hB}(u)$ since the counter actions in y cannot decrease the last component. Finally we have $\mathsf{cost}^*_{hB}(v) \leq \mathsf{cost}^*_{hB}(u)$ by transitivity. □

Let us finally prove the equivalence of the new and old semantics. The proof is inspired from [12].

Lemma 7 (Equivalence). *There is a correction function α such that for all sequences of counter actions $u \in A^*_{\Gamma_h}$ we have $\mathsf{cost}_{hB}(u) \approx_\alpha \mathsf{cost}^*_{hB}(u)$.*

Proof. For the first direction, let us prove that for all $u \in A^*_{\Gamma_h}$ we have $\mathsf{cost}^*_{hB}(u) \leq \mathsf{cost}_{hB}(u)$ thus working with the identity as correction function. Assume that $\mathsf{cost}_{hB}(u) = M$. Let us define $v_0 = (\bar{0}^k, M)$. Let us argue that in case $\mathsf{val}(u, v_0) = (n_1, \ldots, n_k, M)$ we have $\mathsf{val}^*(u, v_0) = (n_1, \ldots, n_k, M)$. Indeed, the only moment the new semantics differs from the old one is when a propagation of the carry occurs. This only happens when some counter exceeds the value M. But, because of the assumption that $\mathsf{cost}_{hB}(u) = M$, this never happens. By applying the Lemma 5, we get

$$\mathsf{cost}^*_{hB}(u) = \pi_{k+1}(\mathsf{val}^*(u, \bar{0}^{k+1})) \leq \pi_{k+1}(\mathsf{val}^*(u, v_0)) \leq M = \mathsf{cost}_{hB}(u) .$$

For the converse direction, we need to prove that there is some correction function α such that for each sequence of counter actions $u \in A^*_{\Gamma_h}$ we have $\mathsf{cost}_{hB}(u) \leq \alpha(\mathsf{cost}^*_{hB}(u))$. We will say that a sequence of counter actions $u \in A_{\Gamma_h}$ is an *m-increase*, if, for all $v \in V$ satisfying $\pi_{k+1}(v) \geq m$ we have $\pi_{k+1}(\mathsf{val}^*(u, v)) > m$. In other words a sequence of counter actions is an m-increase if it forces the carry to propagate to the last component starting from any configuration that has its last component at least m. Remark that if some infix of a word is an m-increase, then the same holds for the word itself.

We claim that for each $i \in [1, k]$ we have

$$(\mathtt{IC}_i)^{m^k} \text{ is an } m\text{-increase.} \qquad (\star)$$

Let us assume some $v \in V$ such that $\pi_{k+1}(v) = m$; the case $\pi_{k+1}(v) > m$ is obvious. Indeed, the effect of executing $\mathtt{IC}_i$ on v is to increment by 1 the components $[i, k]$ as if they were encoding a number of

$k-i+1$ digits in base m. Hence, after $m^{(k-i+1)}$ such increments, an overflow eventually occurs, resulting in an increment of the $(k+1)^{\text{th}}$ component. Thus $\pi_{k+1}(\mathsf{val}^*(u,v)) > m$. The claim ($\star$) is established.

Our second claim is that for the sequence $u = \mathtt{IC}_i^{\alpha(n)}$ with $\alpha(n) = \sum_{m=1}^{n} m^k$ we have $\mathsf{cost}^*_{hB}(u) > n$. Note that u can be decomposed into $u_0 u_1 \cdots u_n$ such that $u_m = \mathtt{IC}_i^{m^k}$. One can now prove that $\pi_{k+1}(\mathsf{val}^*(u_0 \cdots u_m, \bar{0}^{k+1})) > m$ holds for each $m \in [0,n]$ by induction: in the induction step one applies ($\star$) and the monotonicity Lemma 5. Thus we can deduce $\pi_{k+1}(\mathsf{val}^*(u, \bar{0}^{k+1})) > n$.

Consider now the case of some sequence of counter actions $u \in A^*_{\Gamma_h}$ that contains $\alpha(n)$ occurrences of $\mathtt{IC}_i$, while all the other actions are from the alphabet $A' = \{\mathtt{R}_0, \mathtt{IC}_1, \mathtt{R}_1, \ldots, \mathtt{IC}_{i-1}, \mathtt{R}_{i-1}\}$ (i.e., none of the other actions is reseting the counter i). In this case, let us remark that for each action $\sigma \in A'$ and each $v \in V$ it holds $\mathsf{val}^*(\mathtt{IC}_i, v) \leq_{\text{rlex}} \mathsf{val}^*(\mathtt{IC}_i\sigma, v)$, and $\mathsf{val}^*(\mathtt{IC}_i, v) \leq_{\text{rlex}} \mathsf{val}^*(\sigma\mathtt{IC}_i, v)$ (in particular, if, for instance, σ is a reset of some counter $j < i$, since $\mathtt{IC}_i$ is reseting this counter in any case, this is harmless). In combination with the monotonicity Lemma 5, we directly obtain that

$$\mathsf{val}^*(\mathtt{IC}_i^{\alpha(n)}, \bar{0}^{k+1}) \leq_{\text{rlex}} \mathsf{val}^*(u, \bar{0}^{k+1}) \ .$$

From the latter we obtain $\mathsf{cost}^*_{hB}(\mathtt{IC}_i^{\alpha(n)}) \leq \mathsf{cost}^*_{hB}(u)$, which combined with the previous case yields $\mathsf{cost}^*_{hB}(u) > n$.

Consider now some sequence of counter actions $u \in A^*_{\Gamma_h}$ such that $\mathsf{cost}_{hB}(u) > \alpha(n)$. This means that there exists an infix v of u and some counter $i \in [1,k]$ such that $\mathtt{IC}_i$ occurs $\alpha(n)$ times in v, while all the other actions in v are from the alphabet $\{\mathtt{R}_0, \mathtt{IC}_1, \mathtt{R}_1, \ldots, \mathtt{R}_{i-1}\}$. By the previous case and the Monotonicity Lemma, we immediately get $\mathsf{cost}^*_{hB}(v) > n$. Using Corollary 6, one obtains $\mathsf{cost}^*_{hB}(u) > n$. Hence it follows by contraposition that if $\mathsf{cost}^*_{hB}(u) \leq n$ then $\mathsf{cost}_{hB}(u) \leq \alpha(n)$ which proves the claim. □

4 Efficient streaming evaluation of regular cost functions

Our first, immediate, application of the new semantics is to evaluate the cost of a word w with respect to a fixed hierarchical B-automaton $\mathcal{B}$. Given a computable function f which takes as input $w \in \Sigma^*$ and outputs a value $f(w) \in \mathbb{N}_\infty$, we say *$f$ computes $[\![\mathcal{B}]\!]$ exactly if $f = [\![\mathcal{B}]\!]$* and we say *f computes $\mathcal{B}$ up to $\approx$* if $f \approx [\![\mathcal{B}]\!]$.

Let us first argue that for every fixed automaton $\mathcal{B}$ the problem of computing $[\![\mathcal{B}]\!](u)$ *exactly* is in nondeterministic logspace. Indeed, it suffices to guess n and a successful run for deciding if $f(u) \leq n$ and use the fact that nondeterministic logarithmic space is closed under complement for the converse inequality.

Proposition 8. *For each fixed B-automaton $\mathcal{B}$ computing the function $\mathcal{B} : \Sigma^* \to \mathbb{N}_\infty$ exactly can be done in nondeterministic logarithmic space.*

Proof. For every input u it is either the case that $[\![\mathcal{B}]\!](u) = \infty$ or $[\![\mathcal{B}]\!](u) \in [0, |u|]$. First observe that for a given $n \leq |u|$ one can test whether $[\![\mathcal{B}]\!](u) \leq n$ in nondeterministic logspace. This is achieved easily by guessing a run on the fly and checking that its cost is at most n. Analogously one can test (i) for a given $n \leq |u|$ if $[\![\mathcal{B}]\!](u) > n$ and (ii) whether $[\![\mathcal{B}]\!](u) = \infty$. The former holds since nondeterministic logspace is closed under complement, the latter can even be achieved in deterministic logspace. Hence, a nondeterministic logspace machine can compute $[\![\mathcal{B}]\!](u)$ as follows: (a) test whether $[\![\mathcal{B}]\!](u) = \infty$, if applicable, terminate and output ∞, otherwise (b) guess some $n \leq |u|$ and (b1) check that $[\![\mathcal{B}]\!](u) \leq n$ and (b2) check that $[\![\mathcal{B}]\!](u) > n-1$. We do not know if a deterministic logspace algorithm exists for such kind of evaluation. □

However, in this section we are interested in *deterministic streaming algorithms*, that have to process the input from left to right only once, and for this reason have to keep in their memory all the information concerning the prefix read so far. Hence, such a machine can be seen as a memory (the size of which will change as the word is getting longer), together with a transition function that takes the current memory, and combines it with the current letter.

Proposition 9. *There exists a one-counter hB-automaton $\mathcal{B}$ such that every deterministic streaming algorithm that exactly computes $[\![\mathcal{B}]\!]$ requires memory $\Omega(\sqrt{|u|})$ on input u.*

Proof. Consider the function $f : \{a, b, \$\}^* \to \mathbb{N}_\infty$ defined as

$$f(u) = \begin{cases} \min\{\max(m_i, n_i + n) \mid i \in [1, k-1]\} & \text{if } u = a^{m_1} b^{n_1} \dots a^{m_k} b^{n_k} \$ b^n, \\ \infty & \text{otherwise.} \end{cases}$$

The function f can be represented easily by a hB-automaton with one counter. Let us prove that any deterministic streaming algorithm that computes f *exactly* requires memory $\Omega(\sqrt{|u|})$. The idea is to use the technique of Myhill-Nerode equivalence. Two words $u, v \in \{a, b, \$\}^*$ are *equivalent (with respect to f)* if for all $w \in \{a, b, \$\}^*$ we have $f(uw) = f(vw)$. If two words are not equivalent, this means that a machine that has processed the input needs to remember in its memory some information concerning the difference between the two words. If there are many equivalence classes for words of a given length, this means that an important memory is required. Let us count the equivalence classes corresponding to the above function f.

Fix n, and consider some subset $X \subseteq [1, n]$. One constructs the word u_X as follows

$$u_X = \prod_{i \in X} a^{2n-i} b^i,$$

where the product can be made in any order, this will not change for what follows. Clearly, the length of such a word is at most $2n^2$. Let us prove that none of these words belong to the same equivalence class. Indeed, consider two distinct subsets $X, Y \subseteq [1, n]$. This means, (without loss of generality) that there exists some $i \in X \setminus Y$. Let $w = \$ b^{2(n-i)}$. Clearly, the factor $a^{2n-i} b^i$ in u_X is a witness that $f(u_X w) \leq 2n - i$ since $i + 2(n-i) = 2n - i$. However, no such factor exists in u_Y. Consider some other factor $a^{2n-j} b^j$ for some $j \neq i$. Then the corresponding term in the computation of f is $\max(2n-j, j+2(n-i))$. But, if $j < i$, $2n - j > 2n - i$, and if $j > i$ then $j + 2(n-i) > 2n - i$. Thus $f(u_Y w) > 2n - i$. This proves that u_X and u_Y are not equivalent. Thus, there are 2^n such non-equivalent words that are all of length at most $2n^2$. Hence, for processing inputs of length $2n^2$, the memory should separate at least 2^n situations, which thus must contain at least n bits. Hence, for words of length n, a memory of $\Omega(\sqrt{n})$ is necessary. □

We can highly improve this if we use the new semantics, which would provide an answer up to $\approx$.

Theorem 10. *For every B-automaton $\mathcal{B}$ there exists a deterministic streaming algorithm that computes $[\![B]\!]$ up to $\approx$ and uses memory $O(\log n)$ on inputs of length n.*

Proof. Since working up to $\approx$ we can replace every fixed B-automaton with an equivalent (up to $\approx$) hB-automaton $\mathcal{B} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$ by [11]. We emphasize that $\mathcal{B}$ is assumed to be fixed from now on. Let us assume an input word $x \in \Sigma^*$. Our deterministic streaming algorithm will compute a value $\mathbb{B}(x) \in \mathbb{N}_\infty$ and works by operating on the counters in the new semantics and doing an on-the-fly power set construction. During the simulation we store the reachable configurations of $\mathcal{B}$ by a partial function $s : Q \to \mathbb{N}^{k+1}$. Initially we start with the partial function $s = \{(q_I, \bar{0}^{k+1})\}$. Assume that after reading a prefix of w we reached the set of configurations $s = \{(q_1, v_1), \dots, (q_i, v_i)\}$. On a letter a we update the set of reachable configurations by executing all possible transitions and retaining the smallest vector for each reachable state q. Formally the new set of reachable configurations s' is defined as follows, for each $q \in Q$:

$$s'(q) = \begin{cases} \text{undefined} & \text{if } \{v' \in V \mid (p, a, \sigma, q) \in \Delta, (p, v) \in s, \mathsf{val}^*(\sigma, v) = v'\} = \emptyset \\ \widehat{v} & X = \{v' \in V \mid (p, a, \sigma, q) \in \Delta, (p, v) \in s, \mathsf{val}^*(\sigma, v) = v'\} \neq \emptyset \\ & \text{and } \widehat{v} = \min(X), \end{cases}$$

where the minimum is computed with respect to the order $\leq_{\mathsf{rlex}}$. On reaching the end of our input word x with set of configurations s we output eventually

$$\mathbb{B}(x) = \mathsf{cost}^*_{hB}\left(\min\{v \mid (q, v) \in s, q \in F\}\right)$$

which is the cost of the least tuple associated with a final state.

Next we need to show that $\mathbb{B} \approx [\![\mathcal{B}]\!]$. It is straightforward to see that for all words $x \in \Sigma^*$ we have $[\![\mathcal{B}]\!](x) = \infty$ if and only if there is no successful run of the automaton $\mathcal{B}$ on x, that is to say $\mathbb{B}(x) = \infty$. If $\mathbb{B}(x) = n$ then there is a successful run ρ of the automaton $\mathcal{B}$ on x such that $n = \mathsf{cost}^*_{hB}(u)$, where $u \in \mathcal{A}^*_{\Gamma_h}$ is the sequence of counter actions performed in ρ. Hence it is clear from the Equivalence Lemma (Lemma 7) that there is a correction function α such that for all $x \in \Sigma^*$ we have $[\![\mathcal{B}]\!](x) \le \mathsf{cost}_{hB}(u) \le \alpha(\mathsf{cost}^*_{hB}(u)) \le \alpha(\mathbb{B}(x))$.

It remains to show that there is a correction function α such that for all words $x \in \Sigma^*$ we have $\mathbb{B}(x) \le \alpha([\![\mathcal{B}]\!](x))$. Recall that for every $u \in A^*_{\Gamma_h}$ it holds $\mathsf{cost}^*_{hB}(u) \le \mathsf{cost}_{hB}(u)$. Therefore if there is a run ρ (with associated sequence of counter actions u) of the automaton $\mathcal{B}$ on x, then there is a configuration (q, v) in the set of reachable configurations s of the procedure such that $\mathbb{B}(x) \le \pi_{k+1}(v) \le \mathsf{cost}_{hB}(u)$. In particular, this is the case for the run ρ which computes $[\![\mathcal{B}]\!](x)$ and hence $\mathbb{B}(x) \le [\![\mathcal{B}]\!](x)$. This shows that our procedure is correct.

In the above procedure the set of configurations $s : Q \to \mathbb{N}^{k+1}$ requires space $O(k \cdot \log |x|)$, where the factors $|Q|$ and k are fixed since we assume $\mathcal{B}$ to be fixed. $\square$

The algorithm maintains for each state q a $(k+1)$-tuple $\bar{m}_q$ of integers, or ∞. The meaning is that m_q is the least value* for $\le_{\mathsf{rlex}}$ that can have a run of the automaton starting from some initial state and reading the input seen so far. This information is encodable in logarithmic space, and is easy to update by implementing the evaluation of the new semantics. The essential idea behind is that the order $\le_{\mathsf{rlex}}$ allows to keep track of only one run of the automaton for each final state.

5 Uniform memoryless winning strategies in hB-games

In this section we study the question of uniformization for the existence of memoryless strategies in hB-games. We first start by providing some definitions concerning games, and develop a generic technique for constructing memoryless strategies. We then apply this technique to hB-games.

5.1 Games and memoryless strategies

We consider two players (Eve and Adam) playing a turn-based game of infinite duration on a possibly infinite arena. An *arena* $\mathcal{A} = (V_\mathcal{A}, E)$ has a set $V_\mathcal{A} = V_{\mathsf{Eve}} \uplus V_{\mathsf{Adam}}$ of *vertices* that is partitioned into those of the Eve and those of Adam, and a set $E \subseteq V_\mathcal{A} \times V_\mathcal{A}$ of *moves*. We assume, as classical, that for all vertices v there exists an outgoing move $(v, w) \in E$. Under this assumption, there are infinite paths in $\mathcal{A}$ (seen as a graph), that are called *plays*. We denote by Π the set of plays.

A *strategy (tree)* for the player Eve from position v is a (possibly infinite) tree Σ_v equipped with a *labeling* ℓ that maps *nodes* to vertices in the arena, and such that (a) the label of the root is v, (b) for all nodes x, if $\ell(x) \in V_{\mathsf{Eve}}$, then x has exactly one child y such that $(\ell(x), \ell(y)) \in E$, and (c) for all nodes x if $\ell(x) \in V_{\mathsf{Adam}}$, then all children have different labels, and the set of labels of the children is $\{v \in V_\mathcal{A} \mid (\ell(x), v) \in E\}$. A *branch* in a strategy is a sequence of nodes $x_0 x_1 \cdots$ where x_0 is the root, and x_{i+1} is a child of x_i. By definition of a strategy, it is a play in the arena.

A *memoryless strategy* is a mapping Σ from vertices x of Eve to vertices such that $(x, \Sigma(x)) \in E$. Given a vertex v, the memoryless strategy induces a unique strategy $\Sigma[v]$ in the above sense, which is the sole one starting in v and such that for all nodes x, if $\ell(x) \in V_{\mathsf{Eve}}$ then x has one child y which is labeled by $\ell(y) = \Sigma[v](\ell(x))$. A *game* is a pair $\mathcal{G} = (\mathcal{A}, \mathbb{W})$, where $\mathcal{A}$ is an arena and $\mathbb{W} \subseteq \Pi$ is a *winning condition*. A strategy is *winning* if all its plays (branches) belong to $\mathbb{W}$. If there is a strategy from v that is winning, then Eve is said to *win* the game from v.

We are interested in results of the form "if Eve wins the game from v, then there is a winning memoryless strategy from v". Not all winning conditions do satisfy these kind of properties. We introduce below the notion of a *signature scheme*. It is a generalization of the signature technique of Emerson and Streett [24] that was originally developed for parity conditions. In the below description, more winning conditions can be seen to have memoryless winning strategies.

Definition 11. *We call a tuple $\mathcal{S} = (S, \sqsubseteq, (\lambda_e)_{e\in E})$ a* signature scheme *for an arena $\mathcal{A} = (V_\mathcal{A}, E)$ if $(S, \sqsubseteq)$ is a well-ordered*[1] *set that has a maximal element, and $\lambda_e : S \to S$ is a monotone function for all $e \in E$.*

Such a signature scheme can be used to evaluate strategies. A *pre-signature* over a strategy (tree) Σ (for the signature scheme $\mathcal{S} = (S, \sqsubseteq, (\lambda_e)_{e\in E})$) is a mapping s from the nodes of Σ to S such that for all nodes x with child y we have $s(x) \sqsupseteq \lambda_{(\ell(x),\ell(y))}(s(y))$. The *value* of the pre-signature is the label of its root. A winning condition $\mathbb{W}$ is $\mathcal{S}$*-compatible* if there exists some $t \in S$ such that for all strategy trees Σ it holds that Σ is winning if and only if there exists a pre-signature over Σ of value strictly smaller than t with respect to $\sqsubseteq$ (i.e. of value $\sqsubset t$).

Theorem 12. *Let $\mathcal{A}$ be an arena and $\mathcal{S}$ be a signature scheme on $\mathcal{A}$. There exists a memoryless strategy Σ such that for all winning conditions $\mathbb{W}$ that are $\mathcal{S}$-compatible and all vertices v we have that if* Eve *wins the game $(\mathcal{A}, \mathbb{W})$ from v, then $\Sigma[v]$ is winning from v.*

Proof. The main point is to construct the memoryless strategy Σ. For all vertices v, we set m_v to be the least value that a pre-signature over some strategy from v can have. Such a value exists since $\mathcal{S}$ is a well-order and has a maximal element. We further set Σ_v to be the strategy witnessing the existence of m_v. Our memoryless strategy is defined for all vertices v of Eve by $\Sigma(v) = w$ where w is the vertex reached by Σ_v after its first move, *i.e.*, it replicates the first move of Σ_v.

Consider now some winning condition $\mathbb{W}$ that is compatible with $\mathcal{S}$ and some vertex v such that Eve wins the game $(\mathcal{A}, \mathbb{W})$ from v. This means that $m_v \sqsubset t$, where t is the threshold witnessing that $\mathbb{W}$ is $\mathcal{S}$-compatible. Recall that $\Sigma[v]$ is the strategy from v that is induced by Σ and let ℓ denote its labeling. We claim that $\Sigma[v]$ is also winning from v. For this, let s be the mapping which to each node x of $\Sigma[v]$ associates $m_{\ell(x)}$. For the sake of contradiction, assume s is not a pre-signature of $\Sigma[v]$. Then there is a node x in $\Sigma[v]$ such that either (i) $\ell(x) \in V_{\mathsf{Eve}}$ and for the unique child y of x we have $s(x) \sqsubset \lambda_{(\ell(x),\ell(y))}(s(y))$, or (ii) $\ell(x) \in V_{\mathsf{Adam}}$ and for some child y of x we have $s(x) \sqsubset \lambda_{(\ell(x),\ell(y))}(s(y))$. We only treat the case (i) since case (ii) can be proven analogously. Consider a pre-signature s' over some strategy Σ' from v of value m_v. By definition of Σ we have that s' and Σ' exist. Let $e = (\ell(x), \ell(y))$. By the definition of pre-signature we have

$$s'(x) \sqsupseteq \lambda_e(s'(y)).$$

It follows that

$$\lambda_e(s'(y)) \sqsubseteq s'(x) \sqsubseteq s(x) \sqsubset \lambda_e(s(y)),$$

hence $\lambda_e(s'(y)) \sqsubset \lambda_e(s(y))$. By monotonicity (Lemma 5) it follows $s(y) \not\sqsubseteq s'(y)$, thus $s'(y) \sqsubset s(y)$ since $\sqsubseteq$ is total. However, the latter contradicts the definition of s. Thus we have shown that s is a pre-signature of $\Sigma[v]$. Finally, since the value of s is $m_v \sqsubset t$ we conclude that $\Sigma[v]$ is winning. □

5.2 Memoryless winning strategies for Eve in safety hB-games

We now use the technique of the previous section for proving the existence of memoryless optimal strategies for Eve in (safety) hB-games (the situation is completely different if Eve has to satisfy simultaneously a reachability condition, say). This results in Theorem 15.

A *(safety) hB-game* is given by a tuple $\mathcal{G} = (\mathcal{A}, k, \mathtt{act})$, where $\mathcal{A} = (V_\mathcal{A}, E)$ is the arena on which the game $\mathcal{G}$ takes place, k is the number of counters, and $\mathtt{act} : E \to A_{\Gamma_h}$ labels each edge with an *action* from $A_{\Gamma_h} = \{\mathtt{R}_0, \mathtt{IC}_1, \mathtt{R}_1, \dots, \mathtt{IC}_k, \mathtt{R}_k\}$. Naturally the mapping $\mathtt{act}$ can be used to translate any infinite play π in $\mathcal{A}$ into an infinite word $\mathtt{act}(\pi) \in A^\omega_{\Gamma_h}$. The idea is that the pair $(k, \mathtt{act})$ defines a family of accepting conditions $(\mathbb{W}_n)_{n\in\mathbb{N}}$, such that $\pi \in \mathbb{W}_n$ if $\mathsf{cost}_{hB}(\mathtt{act}(\pi')) \leq n$ for all finite prefixes π' of π. If Eve wins the game $(\mathcal{A}, \mathbb{W}_n)$ from v, then we say that Eve n*-wins* the game $\mathcal{G} = (\mathcal{A}, k, \mathtt{act})$. Our goal is to prove that there exists a memoryless strategy Σ such that if Eve n-wins from v whatever are n and v, then following $\Sigma[v]$ ensures Eve to n-win from v. This is not true as it stands, and we need to change the family of winning conditions by an "equivalent" one to obtain such a result.

[1] A binary relation $\sqsubseteq$ over S is a *well-order* if $\sqsubseteq$ is a total order (i.e. $\sqsubseteq$ is antisymmetric, transitive and total) and every non-empty subset of S has a minimal element.

Let us define $\mathbb{W}^*_n$ to be the set of plays π such that $\mathsf{cost}^*_{hB}(\mathtt{mirror}(\mathtt{act}(\pi'))) \leq n$ for all π' that are finite prefixes of π (where $\mathtt{mirror}$ reverses the order of letters in a word). If Eve wins the game $(\mathcal{A}, \mathbb{W}^*_n)$ from v, then we say that Eve *n-wins** the game $\mathcal{G} = (\mathcal{A}, k, \mathtt{act})$. Our first observation is that, up to $\approx$, the notions of winning and winning* are equivalent.

Proposition 13. *There exists a correction function α such that for all plays π and all n, if $\pi \in \mathbb{W}_n$, then $\pi \in \mathbb{W}^*_{\alpha(n)}$, and if $\pi \in \mathbb{W}^*_n$ then $\pi \in \mathbb{W}_{\alpha(n)}$.*

Proof. By Lemma 7 it suffices to show that for each $u \in A^*_{\Gamma_h}$ and each $n \in \mathbb{N}$ we have that $\mathsf{cost}_{hB}(u) \geq n$ implies $\mathsf{cost}_{hB}(\mathtt{mirror}(u)) \geq n$ and thus $\mathsf{cost}_{hB} = \mathsf{cost}_{hB} \circ \mathtt{mirror}$. Indeed, if $\mathsf{cost}_{hB}(u) \geq n$, then there exists an infix $(\mathtt{IC_i})^n$ of u which is also an infix of $\mathtt{mirror}(u)$. Hence $\mathsf{cost}_{hB}(\mathtt{mirror}(u)) \geq n$. $\square$

It happens that the notion of n-winning* behaves much better, and in particular one can provide a unique signature scheme that is compatible with $\mathbb{W}^*_n$ for all n. Recall from Section 3 that V denotes the set of all $(k+1)$-tuples $(n_1, \ldots, n_k, N)$ over $\mathbb{N}$ satisfying $n_i \leq N$ for each $i \in [1, k]$ and each $a \in A_{\Gamma_h}$ defines a monotone function λ_a on V by Lemma 5, where $\lambda_a(v) = \mathsf{val}^*(a, v)$. For this we define the *reverse lexicographic signature scheme* as $\mathcal{S}_{\text{rlex}} = (S_{\text{rlex}}, \sqsubseteq_{\text{rlex}}, \{\lambda_a \mid a \in A_{\Gamma_h}\})$, where the domain is $S_{\text{rlex}} = V \cup \{\infty\}$ and the order $\sqsubseteq_{\text{rlex}}$ on S_{rlex} is the extension of $\leq_{\text{rlex}}$ on V by putting $v \sqsubseteq_{\text{rlex}} \infty$ for each $v \in \mathbb{N}^{k+1}$ and lifting the domain of λ_a to S_{rlex} by setting $\lambda_a(\infty) = \infty$. It is indeed a signature scheme according to Lemma 5. Let us now prove compatibility of $\mathcal{S}_{\text{rlex}}$.

Proposition 14. *For each $n \in \mathbb{N}$ we have that $\mathbb{W}^*_n$ is $\mathcal{S}_{rlex}$-compatible.*

Proof. We prove that for all $n \in \mathbb{N}$ and all strategies Σ we have that Σ is n-winning* if and only if there is a pre-signature of value v over Σ, where $v \sqsubset (\bar{0}^k, n+1)$.

Let r denote the root of Σ.

Let us first assume that there is a pre-signature s of value v over Σ, where $v \sqsubset (\bar{0}, n+1)$. For this, let us fix an arbitrary finite branch π in Σ that ends in node x. To show that $\mathsf{cost}^*_{hB}(\mathtt{mirror}(\mathtt{act}(\pi))) \leq n$ we prove $\mathsf{val}^*(\mathtt{mirror}(\mathtt{act}(\pi)), \bar{0}^{k+1}) \sqsubseteq s(r) \sqsubset (\bar{0}, n+1)$ holds. One can easily prove that we have $\mathsf{val}^*(\mathtt{mirror}(\mathtt{act}(\pi)), s(x)) \sqsubseteq s(r)$ by induction on $|\pi|$. We have

$$\begin{array}{rcl}\mathsf{val}^*(\mathtt{mirror}(\mathtt{act}(\pi)), \bar{0}^{k+1}) & \overset{\text{Lemma 5}}{\sqsubseteq} & \mathsf{val}^*(\mathtt{mirror}(\mathtt{act}(\pi)), s(x)) \\ & \sqsubseteq & s(r) \\ & \sqsubset & (\bar{0}^k, n+1)\end{array}$$

Conversely assume that the strategy Σ with labeling ℓ is n-winning*. We have to define a pre-signature s of value v, where $v \sqsubset (\bar{0}^k, n+1)$. To each node x of Σ we assign

$$s(x) \;=\; \sup\{\mathsf{val}^*(\mathtt{mirror}(\mathtt{act}(\pi)), \bar{0}^{k+1}) \mid \pi \text{ is a branch from } x \text{ in } \Sigma\}$$

It remains to show that s is a pre-signature. For this we have to show that for each node x of Σ and for each child y of x we have

$$s(x) \;\sqsupseteq \lambda_{(\ell(x),\ell(y))}(s(y)).$$

In case $\ell(x) \in V_{\mathsf{Eve}}$ we have that y is the only child of x and moreover

$$\begin{array}{rcl}s(x) & \sqsupseteq & \sup\{\mathsf{val}^*(\mathtt{mirror}(\mathtt{act}(\pi)), \bar{0}^{k+1}) \mid \pi \text{ is a branch from } x \text{ in } \Sigma\} \\ & \sqsupseteq & \sup\{\mathsf{val}^*(\mathtt{mirror}(\mathtt{act}(\ell(x), \ell(y)) \cdot \mathtt{act}(\pi')), \bar{0}^{k+1}) \mid \pi' \text{ is a branch from } y \text{ in } \Sigma\} \\ & \sqsupseteq & \mathsf{val}^*\left(\mathtt{act}(\ell(x), \ell(y)), \sup\{\mathsf{val}^*(\mathtt{act}(\pi'), \bar{0}^{k+1}) \mid \pi' \text{ is branch from } y \text{ in } \Sigma\}\right) \\ & = & \mathsf{val}^*(\mathtt{act}(\ell(x), \ell(y)), s(y)) \\ & = & \lambda_{(\ell(x),\ell(y))}(s(y))\end{array}$$

The case $\ell(x) \in V_{\mathsf{Adam}}$ can be shown analogously. $\square$

At this point, by applying Theorem 12, we immediately obtain the following main result of the section.

Theorem 15. *In each safety hB-game $\mathcal{G}$ there exists a memoryless strategy Σ such that for all vertices v and all $n \in \mathbb{N}$, if* Eve *is n-winning* from v then $\Sigma[v]$ is n-winning*.*

This statement is uniform in the sense that this strategy is optimal regardless of both v and n. Previous similar results were known for strategies that depended upon n [12].

6 Uniform history-determinacy of max-automata

History-determinism, introduced in [17], is a notion of sequentiality which sits in between nondeterminism and determinism. An automaton is history-deterministic if it is the homomorphic image of a deterministic automaton with potentially infinitely many states. See [10], Section 5 for a general introduction. The importance of this notion is from the fact that it allows to compose automata with games in a semantically correct way. This is a crucial argument in the proof of decidability of cost monadic logic over finite trees [14].

6.1 History-determinism

In the following, for any B-automaton $\mathcal{B} = (Q, \Sigma, \Gamma, q_0, \Delta, F)$ and any run $\rho \in \Delta^*$ we define $\mathsf{cost}(\rho)$ as an abbreviation for $\mathsf{cost}(\pi_3(\rho))$, i.e. we take the cost of the counter actions that are induced by ρ. Similar remarks apply to cost_{hB}, cost^*_{hB}, val and val^*. We first recall the basic notions related to history-determinism for B-automata. For each $i \geq 1$ let $\mathcal{A}_i = (Q_i, \Sigma, \Gamma, q_i^0, \Delta_i, F_i)$ be a B-automaton where Q_i is countable and possibly infinite. The definitions of run and cost can be naturally extended to the scenario where Q_i is infinite. Given a finite set Q and a map $h : Q_i \to Q$ we define the homomorphic image of $\mathcal{A}_i$ under h to be the automaton $h(\mathcal{A}_i) = (Q_i, \Sigma, \Gamma, h(q_i^0), h(\Delta_i), h(F_i))$ where

$$h(\Delta_i) = \{(h(p), a, \bar{\sigma}, h(q) \mid (p, a, \bar{\sigma}, q) \in \Delta_i\}.$$

Let $\mathcal{B}$ be a finite state B-automaton such that $h(\mathcal{A}_i) = \mathcal{B}$. We observe that the image under h of a run of $\mathcal{A}_i$ on a word w is a a run of $\mathcal{B}$ on w. This implies that $[\![\mathcal{B}]\!](w) \leq [\![\mathcal{A}_i]\!](w)$.

Definition 16 (Variant 1). *We say a B-automaton $\mathcal{B} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$ is a history-deterministic automaton if there is a correction function α, a sequence $(\mathcal{A}_i)_{i\in\mathbb{N}}$ of deterministic B-automata with potentially infinitely many states, and a sequence of maps $(h_i : Q_i \to Q)_{i\in\mathbb{N}}$ such that*

- $h_i(\mathcal{A}_i) = \mathcal{B}$,
- $[\![\mathcal{B}]\!](w) \leq i$ *then* $[\![\mathcal{A}_i]\!](w) \leq \alpha(i)$.

An equivalent way to state the above definition is by using translation strategies, which we will be using later. A *translation strategy* ζ is a function $\zeta : \Delta^* \times \Sigma \to \Delta$ such that for each $w \in \Sigma^*$ a unique run $\tilde{\zeta}(w)$ of $\mathcal{B}$ on w is defined in the following way: We require that the induced mapping $\tilde{\zeta} : \Sigma^* \to \Delta^*$ defined inductively as $\tilde{\zeta}(\varepsilon) = \varepsilon$ and $\tilde{\zeta}(wa) = \tilde{\zeta}(w)\zeta(\tilde{\zeta}(w), a)$ for each $w \in \Sigma^*$ and each $a \in \Sigma$ has only runs of $\mathcal{B}$ in its range. Note that for every $i \in \mathbb{N}$ it is the case that $[\![B]\!](w) \leq \mathsf{cost}(\tilde{\zeta}_i(w))$.

Definition 17 (Variant 2). *The automaton $\mathcal{B}$ is* history-deterministic *automaton if there exists a correction function α and a translation strategy ζ_i for every $i \in \mathbb{N}$ such that for all words w and all $i \in \mathbb{N}$, if $[\![\mathcal{B}]\!](w) \leq i$ then* $\mathsf{cost}(\tilde{\zeta}_i(w)) \leq \alpha(i)$.

Let us verify that these two definitions are equivalent. Assume $\mathcal{B}$ is history-deterministic with respect to the translation strategies $(\zeta_i)_{i\in\mathbb{N}}$. For $\pi \in \Delta^*$ we define $\mathsf{in}(\pi)$ and $\mathsf{out}(\pi)$ to be the start and end state of π respectively with the convention that $\mathsf{target}(\varepsilon) = q_I$. Similarly let $\mathsf{label}(\pi)$ to be the projection of π to the Σ component. Let Π be the set of all runs of $\mathcal{B}$. We define $\mathcal{A}_i$ with the set of states $Q_i = \Pi$, the initial state $q_i^0 = \varepsilon$ (the empty run), and set of final states $F_i = \{\pi \in Q_i \mid \mathsf{target}(\pi) \in F\}$ and the set of transitions

$$\Delta_i = \{(\pi, a, \bar{\sigma}, \pi \cdot (p, a, \bar{\sigma}, q)) \mid \pi, \pi\delta \in \Pi, \zeta_i(\pi, a) = \delta\}.$$

Let $h_i : Q_i \to Q$ be the map which associates to π the state $\mathsf{target}(\pi)$. Assume $[\![\mathcal{B}]\!](w) \leq i$. Then by definition of $\mathcal{A}_i$ it follows that $\mathcal{A}_i(w) \leq \alpha(i)$ (Consider the path in $\mathcal{A}_i$ given by $\tilde{\zeta}_i(w)$). However $h_i(\mathcal{A}_i)$ may not be $\mathcal{B}$ (this happens when ζ_i is not surjective). But one can always add isolated states and transitions to $\mathcal{A}_i$ without affecting the costs to make $h_i(\mathcal{A}_i)$ to be $\mathcal{B}$. The details are left to the reader.

For the other direction let $\mathcal{B}$ be history-deterministic according to Variant 1. Then we define the translation strategy ζ_i inductively as

- $\zeta_i(\varepsilon, a) = h_i(\delta_0)$ where $\delta_0 \in \Delta_i$ is the unique transition $(q_i^0, a, \bar{\sigma}, q)$ for some $q \in Q_i$ and $\bar{\sigma} \in \{\mathtt{ic}, \mathtt{r}, \varepsilon\}^\Gamma$,

– $\zeta_i(\pi, a) = h_i(\delta)$ where δ is the unique transition $(p, a, \bar{\sigma}, q)$ where $p = \Delta_i(q_i^0, \mathsf{label}(\pi))$, $q \in Q_i$ and $\bar{\sigma} \in \{\mathtt{ic}, \mathtt{r}, \varepsilon\}^{\Gamma}$.

Let $[\![\mathcal{B}]\!](w) \leq i$ then we claim that $\mathsf{cost}(\tilde{\zeta_i}(w)) \leq \alpha(i)$. By Variant 1 $[\![\mathcal{A}_i(w)]\!] \leq \alpha(i)$. Let π be the unique accepting run of $\mathcal{A}_i$ witnessing the cost $\alpha(i)$. By definition of ζ and h_i, it follows that $\mathsf{cost}(\tilde{\zeta_i}(w)) \leq \alpha(i)$.

It is known that every B-automaton is equivalent (up to $\approx$) to a history-deterministic B-automaton [11]. On the other hand deterministic and unambiguous B-automata are strictly weaker.

We can define a natural uniform variant of history-determinism as follows.

Definition 18 (Uniform history-determinism). *An automaton $\mathcal{B}$ is* uniform history-deterministic *if there exists a correction function α and a translation strategy $\zeta : \Delta^* \times \Sigma \to \Delta$ such that for all words w if $[\![\mathcal{B}]\!](w) \leq i$ then $\mathsf{cost}(\tilde{\zeta}(w)) \leq \alpha(i)$.*

It is known that in general B-automata are not uniformly history-deterministic [14]. Then arises a natural question, *whether there is a model of automata for regular cost functions which admits uniform history-determinism?* We answer this question positively by showing that uniform history deterministic $\max$-automata accept all (and only) regular cost functions. This is going to be the subject of the rest of the paper.

6.2 $\max$-Automata

We introduce the class of $\max$-automata and $\min$-automata as generalizations of B-automata and S-automata (similar automata, in their deterministic form, were already used in the context of deciding the satisfiability of WMSO+$\mathbb{U}$ and WMSO+$\mathbb{R}$ over infinite words in [4]). We show that $\max$-automata recognize only regular cost functions, and furthermore, every regular cost function is accepted by a uniformly history-deterministic $\max$-automaton.

A *max-automaton* is a finite state automaton equipped with a finite set of counters $\Gamma = \{c_1, \ldots, c_n\}$. *Counter operations* are of the form $(c_1 := e_1, \ldots, c_n := e_n)$ where each e_j is a $\max$ of some subset of $\{c_1, \ldots, c_n, c_1 + 1, \ldots, c_n + 1\}$. Let A_Γ denote the set of all possible counter operations.

Formally a max-automaton is a tuple $(Q, \Sigma, \Gamma, \Delta, q_0, F, c_{out})$, where Q is the finite set of states, Σ is the finite input alphabet, Γ is the set of counters, $\Delta \subseteq (Q \times \Sigma \times A_\Gamma \times Q)$ is the set of *transitions*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *final states* and $c_{out} \in \Gamma$ is the output counter. On a word $w = a_1 \cdots a_n$ a *successful run* of the automaton is a sequence of transitions $\delta_1 \cdots \delta_n$ starting in the initial state, ending in a final state and such that successive transitions share a common state. The run ρ defines the sequence of vectors $v_0, \ldots, v_n \in \mathbb{N}^k$, where v_0 is the zero vector and each v_{i+1} is obtained from v_i by updating according to δ_i. The *cost of the run ρ* is defined to be the value of c_{out} in the final counter configuration v_n. The *cost of the word w* is the infimum of costs of all successful runs on w.

By the following two propositions we prove that $\max$-automata defines exactly the class of cost functions.

Proposition 19. *All regular cost functions are accepted by* $\max$*-automata.*

Proof. To show that all regular cost functions are accepted by max-automata it is enough to observe that B automata as well as hierarchical B-automata are a natural special case of max-automata where the counter actions $\mathcal{A}_\Gamma$ are expressed using max-expressions. Let $\mathcal{B}$ be a hierarchical B-automaton with counters $\Gamma = \{1, \ldots, k\}$. We construct a $\max$-automaton $\mathcal{A}$ with counters $\{c_1, \ldots, c_k, c_{zero}, c_{out}\}$ which has the same set of states as that of $\mathcal{B}$. Moreover the initial state and final states of $\mathcal{A}$ are the same as that of $\mathcal{B}$. The counters $\{c_1, \ldots, c_k\}$ simulate the counters $\{1, \ldots, k\}$ of $\mathcal{B}$. The counter c_{zero} is never incremented and is used to perform the reset operations. The unique output counter c_{out} holds value of the run so far. The transitions of $\mathcal{A}$ are obtained by replacing the counter actions in the transitions of $\mathcal{B}$ by equivalent $\max$-expressions in the following way. To perform the operation $\mathtt{R}_i$ we use the $\max$-expression $c_1 := c_{zero}, \ldots, c_i := c_{zero}$. Similarly $\mathtt{IC}_i$ is performed by the expression $c_1 := c_{zero}, \ldots, c_{i-1} := c_{zero}, c_i := c_i + 1$. On every transition of $\mathcal{A}$, the maximum value obtained by any counter is stored in the counter c_{out} by the expression $c_{out} := \max\{c_{out}, c_1, \ldots, c_n\}$. It is straightforward to see that the automaton $\mathcal{A}$ simulates $\mathcal{B}$ and computes the same cost function as that of $\mathcal{B}$. ☐

Next we want to show the converse, namely all cost functions accepted by $\max$-automata are regular. We show a stronger result. For this purpose we define $\min$-$\max$-automata. These are identical to $\max$-automata except that the counter operations are of the form $(c_1 := e_1, \ldots, c_n := e_n)$ where each e_ℓ is of the form $\min_{i\in[1,k]} \max_{j\in[1,l_i]} f_{ij}$ where $f_{ij} \in \{c_1, \ldots, c_n, c_1{+}1, \ldots, c_n{+}1\}$. For notational convenience we identify expression e_ℓ with the "clause" $\{\{(c, \iota) \mid f_{ij} = c + \iota, j \in [1, l_i]\} \mid i \in [1, k]\}$.

The definition of a run of the automaton and cost are defined exactly in the same way.

Proposition 20. *All cost functions accepted by* $\min$-$\max$-*automata are regular.*

Proof. For a $\min$-$\max$-automaton $\mathcal{A} = (Q, \Sigma, \Gamma, \Delta, q_0, F, c_{out})$ we construct an equivalent B-automaton $\mathcal{B}$. Let $\mathsf{run}(\mathcal{A}) \subseteq \Delta^*$ be the set of all accepting runs of the automaton $\mathcal{A}$. For a run $\rho \in \mathsf{run}(\mathcal{A})$ define $\mathsf{cost}(\rho)$ to be the value of the output counter at the last position. Let $R : \Delta^* \to \mathbb{N}_\infty$ be the cost function

$$R(\rho) = \begin{cases} \infty & \text{if } \rho \notin \mathsf{run}(\mathcal{A}) \\ \mathsf{cost}(\rho) & \text{if } \rho \in \mathsf{run}(\mathcal{A}) \end{cases}$$

We claim that it is enough to show that R is regular. Assume it is the case. Let $h : \Delta \to \Sigma$ be the projection which maps a transition to its Σ component. We can extend h naturally to the morphism $h : \Delta^* \to \Sigma^*$. Observe that the cost function $[\![\mathcal{A}]\!]$ defined by the automaton $\mathcal{A}$ is the $\inf$-projection of R under h in the following sense;

$$[\![\mathcal{A}]\!](w) = \inf\big\{R(\rho) \mid \rho \in \Delta^*, h(\rho) = w\big\}.$$

It is a key property of regular cost functions that they are closed under $\inf$-projections [11]. Hence it follows that $[\![A]\!]$ is regular provided R is regular. We next show that R is regular. Further observe that there is a finite state automaton $\mathcal{A}_1$ which accepts the set $\mathsf{run}(\mathcal{A})$. Seen as a B-automaton $\mathcal{A}_1$ defines the cost function

$$[\![\mathcal{A}_1]\!](\rho) = \begin{cases} \infty & \text{if } \rho \notin \mathsf{run}(\mathcal{A}) \\ 0 & \text{if } \rho \in \mathsf{run}(\mathcal{A}) \end{cases}$$

Since regular cost functions are closed under the operation $\max$ it is enough to show that (†) there is a B-automaton which given a run $\rho \in \mathsf{run}(\mathcal{A})$ outputs $\mathsf{cost}(\rho)$.

Assume $f : \Sigma^* \to \mathbb{N}_\infty$ is a cost function. We define $\mathtt{mirror}(f)$ as the cost function $(\mathtt{mirror}(f))(u) = f(\mathtt{mirror}(u))$. It is easy to see that if f is regular then $\mathtt{mirror}(f)$ is also regular. The reverse of a B-automaton accepting f (obtained by reversing the transitions and exchanging initial and final states) accepts the cost function $\mathtt{mirror}(f)$.

Therefore it suffices to show that there is a B-automaton which reads $\mathtt{mirror}(\rho)$ and outputs $\mathsf{cost}(\rho)$ for every $\rho \in \mathsf{run}(\mathcal{A})$. We will construct an alternating distance automaton (defined below) which accepts the function does the above. Since cost functions computed by alternating distance automata are regular [12] it implies the claim (†).

We recall the definition of alternating distance automata. An alternating distance automaton $\mathcal{D}$ is a tuple $\mathcal{D} = (Q, \Sigma, \kappa, q_0, F)$ where Q is the finite set of states, Σ is the alphabet, q_0 is the initial state, F is the set of final states and $\kappa : Q \times \Sigma \to \mathsf{DNF}^+(Q \times \{0, 1\})$ is the set of transitions where $\mathsf{DNF}^+(Q \times \{0, 1\})$ denotes the set of all positive boolean formulas over the literals $Q \times \{0, 1\}$ in disjunctive normal form. For convenience we identify a formula $\bigvee_{i=1}^{k} \bigwedge_{j=1}^{l_i} e_{ij} \in \mathsf{DNF}^+(Q \times \{0, 1\})$, where $e_{ij} \in (Q \times \{0, 1\})$, with the set $\{\{e_{ij} \mid j \in [1, l_i]\} \mid i \in [1, k]\}$. For a state $p \in Q$ and a word $u \in \Sigma^*$ we define the *distance* of u from p, in notation $d(p, u) \in \mathbb{N}_\infty$, inductively as

$$d(p, u) = \begin{cases} \infty & \text{if } w = \varepsilon, p \notin F \\ 0 & \text{if } w = \varepsilon, p \in F \\ \inf\limits_{i\in[1,k]} \{ \sup\limits_{j\in[1,l_i]} \{d(q, v) + \iota \mid e_{ij} = (q, \iota)\}\} & \text{if } u = av, \kappa(p, a) = \bigvee_{i=1}^{k} \bigwedge_{j=1}^{l_i} e_{ij}. \end{cases}$$

The cost function defined by $\mathcal{D}$ is given by $[\![\mathcal{D}]\!](w) = d(q_0, w)$. It is known that cost functions defined by alternating distance automata are regular. In fact cost functions defined by alternating B-automata are regular [14]).

Recall the min-max-automaton $\mathcal{A} = (Q, \Sigma, \Gamma, \Delta, q_0, F, c_{out})$. Next we construct the alternating distance automaton $\mathcal{D}$ with the set of states Γ, input alphabet Δ, initial state c_{out} and final states Γ. To disambiguate nowonwards we use bold letters (like $\mathbf{c}_\ell$) to denote the states. On a state $\mathbf{c}_\ell$ and input letter $\delta = (p, a, \langle c_1 := e_1, \dots, c_n := e_n\rangle, q) \in \Delta$ where $e_\ell = \{\{(c_{ij}, \iota_{ij}) \mid j \in [1, l_i]\} \mid i \in [1, k]\}$, the transition $\kappa(\mathbf{c}_\ell, \delta)$ is defined as,

$$\kappa(\mathbf{c}_\ell, \delta) = \bigvee_{i\in[1,k]} \bigwedge_{j\in[1,l_i]} (\mathbf{c_{ij}}, \iota_{ij}).$$

Let $\texttt{mirror}(\rho)$, where $\rho \in \mathsf{run}(\mathcal{A})$, be an input word and let ρ' be a prefix of ρ. Then we prove by induction that value of a counter c_ℓ after the sequence of transitions ρ' is same as the distance of $\texttt{mirror}(\rho')$ from the state $\mathbf{c}_\ell$. When ρ' is the empty prefix the claim holds trivially since the value of the counter c_ℓ is zero as well as the distance $d(\mathbf{c}_\ell, \varepsilon)$ since $\mathbf{c}_\ell$ is a final state. For the inductive case assume the claim holds for all counters c and the prefix ρ'. We need to show the claim for the counter c_ℓ and the prefix $\rho\delta$. After executing the transition δ the value of c_ℓ is given by

$$c_\ell = \min_{i\in[1,k]}\{\max_{j\in[1,l_i]}\{c_{ij} + \iota_{ij}\}\}.$$

By induction hypothesis,

$$c_\ell = \min_{i\in[1,k]}\{\max_{j\in[1,l_i]}\{d(\mathbf{c_{ij}}, \rho) + \iota_{ij}\}\},$$

which is by definition of the distance is same as $\mathbf{c}_\ell$. Hence the value of the output counter c_{out} after the sequence of transitions is exactly the same as the distance of $\texttt{mirror}(\rho)$ from the input state $\mathbf{c}_{out}$. Hence the automaton $\mathcal{D}$ computes $\mathsf{cost}(\texttt{mirror}(\rho))$. $\square$

One can define the deterministic variant of max-automata naturally as the subclass of max-automata where for any state $p \in Q$ on a letter $a \in \Sigma$ there is at most one transition in Δ. Like B-automata the deterministic variant of max-automata is also strictly weaker. For instance the function $f : w \in \{a, b\}^* \to \min(|w|_a, |w|_b)$, where $|w|_a$ (resp. $|w|_b$) denotes the number of a's (resp. b's) in w, is not accepted by any deterministic max-automaton. The proof involves an algebraic characterization of these automata using stabilization semigroups and is beyond the scope of this exposition. However, next we show that uniform history-deterministic max-automata accept all regular cost functions.

Theorem 21. *For every hB-automaton $\mathcal{B}$ there is a cost-equivalent uniform history-deterministic max-automaton $\mathcal{B}'$.*

Proof. Let us fix an hB-automaton $\mathcal{B} = (Q, \Sigma, \Gamma, q_0, \Delta, F)$, where $|\Gamma| = k$. We will use a construction similar to the on-the-fly evaluation of the hierarchical B-automata. The automaton $\mathcal{B}'$ on reading a word will do a subset construction. If two configurations with the same state are reachable then it will choose the best configuration according to the new semantics with the help of the translation strategy.

The automaton $\mathcal{B}'$ does the operations in A_{Γ_h} and simulates the new semantics. The semantics of $\mathtt{R}_j$ is the same in both semantics and can be implemented by the max-expression $c_1 := \max\{c_{zero}\}, \dots, c_j := \max\{c_{zero}\}$ (we identity this expression with the action $\overline{\mathtt{R}_j}$). But for the instruction $\mathtt{IC}_j$, since the max-automaton has no way of comparing the values of two counters (necessary for performing addition with carry) it will perform the update of the counters nondeterministically and will use the translation strategy to resolve the nondeterminism. For this let $\overline{\mathtt{IC}_j}$ denote the max-expression which resets all the counters $i < j$ and increments the counter j and leaves all other counters unchanged (implemented by the max-expression $c_1 := \max\{c_{zero}\}, \dots, c_{j-1} := \max\{c_{zero}\}, c_j := \max\{c_j + 1\}$). We denote by $A_{\Gamma_{max}}$ the set of max expressions $\{\overline{\mathtt{R}_1}, \dots, \overline{\mathtt{R}_k}, \overline{\mathtt{IC}_1}, \dots, \overline{\mathtt{IC}_{k+1}}\}$.

Next we give a description of the automaton $\mathcal{B}' = (Q', \Sigma, \Gamma', \Delta', q_0', F')$ whose states Q' are precisely the subsets of Q and furthermore it has counters from the set $\Gamma' = (Q \times [1, k+1]) \cup \{c_{zero}, c_{out}\}$, in other words each counter of $\mathcal{B}'$ stores for each state of $\mathcal{B}$ a $(k+1)$-tuple of counters to be simulated. Intuitively, each counter of the form $(q, v) \in Q \times [1, k+1]$ aims at simulating the new semantics of runs of $\mathcal{B}$ that end in q. In addition there is the output counter c_{out} and the counter c_{zero} which always holds value 0. The initial state q_0' of $\mathcal{B}'$ is the set $q_0' = \{q_0\}$ and the set of final states F' are $F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$.

Recall that a state of $\mathcal{B}'$ is of the form $S \subseteq Q$. A *macro transition* $\Delta_S \subseteq \Delta$ from S on the letter a is a subset of Δ such that

(1) all the transitions $\delta \in \Delta_S$ are on the letter a,
(2) all transitions in Δ_S are transitions from a state in S, and
(3) if the state q is reached from a state in S on letter a (denoted as $q \in \Delta(S, a)$) there is a *unique* transition $\delta \in \Delta_S$ to q.

For a macro transition $\Delta_S = \{\delta_1, \dots, \delta_m\}$ of size m, the set of *macro counter actions* is the set of tuples,

$$A^{\Delta_s}_{\Gamma_{max}} = \left\{ (e_{\delta_1}, \dots, e_{\delta_m}) \in (A_{\Gamma_{max}})^m \;\middle|\; \begin{array}{ll} e_i = \overline{\mathtt{R}_j} & \text{if } \; \delta_i = (p, a, \mathtt{R}_j, q) \\ e_i \in \{\overline{\mathtt{IC}_j}, \dots, \overline{\mathtt{IC}_{k+1}}\} & \text{if } \; \delta_i = (p, a, \mathtt{IC}_j, q) \end{array} \right\}$$

For a macro transition $\Delta_S = \{\delta_1, \dots, \delta_m\}$, a macro counter action $\bar{e} = (e_{\delta_1}, \dots, e_{\delta_m}) \in A^{\Delta_s}_{\Gamma_{max}}$ and a transition $\delta \in \Delta_S$ to a final state $q_f \in F$ we denote by $\sigma_{\Delta_S, \bar{e}, q_f}$ the counter operation (using max-expressions) on Γ' which does the following.

(1) For each $\delta_i = (p, a, \sigma, q) \in \Delta_S$ the counters $\{q\} \times \{1, \dots, k+1\}$ are obtained from the counters $\{p\} \times \{1, \dots, k+1\}$ by the max expression e_{δ_i}. In addition we will also make sure that after update the counter $(q, k+1)$ is holds the maximum value of any of the counters $\{q\} \times \{1, \dots, k+1\}$. One can achieve this easily by modifying the max expression e_{δ_i}. This step is crucial, since this ensures that if the counters hold the least possible value only if it is updated correctly.
(2) Let $\delta = (p, a, \sigma, q_f) \in \Delta_S$ be the transition leading to state q_f in Δ_S. Then the output counter c_{out} set to be the value of the counter $(p, k+1)$ when evaluated by the max expression e_δ.
(3) All remaining counters are kept unchanged.

Now the transitions of $\mathcal{B}'$ (denoted by the set Δ') are of the form $(S, a, \sigma_{\Delta_S, \bar{e}, q_f}, S')$ such that Δ_S is macro transition on S and $\sigma_{\Delta_S, \bar{e}, q_f}$ is as described earlier, and S' is the set of states reachable from S on letter a.

Next we define the translation strategy $\zeta : \Delta'^* \times \Sigma \to \Delta'$. Let $\rho' = \delta'_1 \cdots \delta'_n$ be a run of the automaton $\mathcal{B}'$ ending in the state S. Recall that each transition δ'_i is of the form $(S_i, a, \sigma_{\Delta_{S_i}, \bar{e}, q_f}, S_{i+1})$. To define $\zeta(\rho', a)$ it is enough to choose a macro transition Δ_S of $\mathcal{B}$ on the letter a, a macro counter transition $\bar{e} \in A^{\Delta_s}_{\Gamma_{max}}$ and a final state (if there exists one) $q_f \in \Delta(S, a)$. Choosing a macro transition Δ_S amounts to choosing for each state $q \in \Delta(S, a)$ a unique transition incident on q from a state in S. For each state p in S, the run ρ' defines a unique run $\rho_p = \delta_1 \cdots \delta_n$ of $\mathcal{B}$ such that each $\delta_i \in \Delta_{S_i}$ and δ_n is the unique transition to the state p in Δ_{S_n}. The uniqueness follows from the fact that for each state in S_{i+1} there is exactly one transition in Δ_{S_i} incident on it. Let q be a state that belongs to $\Delta(S, a)$ and let Δ_{Sq} be the set of transitions (p, a, σ, q) in Δ such that $p \in S$. We choose the transition $\delta = (p, a, \sigma, q) \in \Delta_{Sq}$ such that the value of the run $\pi_3(\rho_p \delta)$ with the new semantics (namely $\mathsf{val}^*\,(\pi_3(\rho_p \delta))$) is minimal, to be part of the macro transition Δ_S. The max epxression e_{δ_i} is chosen so that the vector $\mathsf{val}^*(\pi_3(\rho_p))$ is correctly updated. Similarly q_f is chosen to be a $q_f \in F \cap \Delta(S, a)$ such that the value $\mathsf{val}^*(\pi_3(\rho_p \delta))$ of the run $\rho_p \delta$ is minimal according to the new semantics.

We lift the translation strategy $\zeta : \Delta'^* \times \Sigma \to \Delta'$ to $\tilde{\zeta} : \Sigma^* \to \Delta'^*$ as described previously. We need to prove two things, namely

(1) the automaton $\mathcal{B}'$ is history-deterministic, that is to say, there is an α such that $\mathsf{cost}(\tilde{\zeta}(w)) \leq \alpha([\![\mathcal{B}']\!](w))$ for all $w \in \Sigma^*$, and
(2) the cost function computed by $\mathcal{B}$ and the cost function defined by the translation strategy are the same.

We prove (1), namely that $\mathsf{cost}(\tilde{\zeta}(w)) \leq \alpha([\![\mathcal{B}']\!](w))$ for each $w \in \Sigma^*$. In fact we take α to be the identity function. Let us first explain the complication here. Assume the translation strategy yields the run $\tilde{\zeta}(w) = \rho'$ and the value $[\![\mathcal{B}']\!](w)$ corresponds to the run ρ'' witnessing the infimum cost among all runs. From the construction of the automaton it is clear that the projection of the run ρ' and ρ'' yields the same sequence of states (namely the set of reachable states of $\mathcal{B}$ on the corresponding prefix). It is only in the update of the counters that the runs ρ' and ρ'' differ. We know that ρ' updates each set of counters $\{p\} \times \{1, \dots, k+1\}$ correctly according to the new semantics, while ρ'' may not. We want to argue that if ρ'' chooses an incorret update it will not have a smaller output value. For this it is enough to observe that (as we noted before) if the set of counters $\{p\} \times \{1, \dots, k+1\}$ is not updated correctly the resulting tuple will be strictly bigger (with respect to $\leq_{\text{rlex}}$) than the tuple obtained by a correct update. Therefore we can conclude that (we are assuming an implicit induction argument here) the following. Assume ρ' and ρ'' finish in the state S. Let $q \in S$ and $(v'_1, \ldots, v'_{k+1})$ and $(v''_1, \ldots, v''_{k+1})$ be the values of the counters $(p, 1), \ldots, (p, k+1)$ after ρ' and ρ''. Then it is the case that $(v'_1, \ldots, v'_{k+1}) \leq_{\text{rlex}} (v''_1, \ldots, v''_{k+1})$. From this it follows that the output of ρ'' is at least as big as the output of ρ'.

For the correctness of the construction it remains to show that $[\![\mathcal{B}]\!] \approx \mathsf{cost} \circ \tilde{\zeta}$.

Let us first observe that for all $w \in \Sigma^*$ it is the case that

$$[\![\mathcal{B}']\!](w) \leq \mathsf{cost}(\tilde{\zeta}(w)). \tag{$\star$}$$

Let ρ' be the unique run of the max-automaton $\mathcal{B}'$ on the word w according to the translation strategy $\tilde{\zeta}$. The cost of the run ρ' is the value of the counter c_{out} at the end which in turn corresponds to the value of a counter (q_f, c_{k+1}) (by definition). But the value of (q_f, c_{k+1}) is exactly the cost (in the new semantics) of the unique run ρ of $\mathcal{B}$ on w ending in state $q_f \in F$ traced by the run ρ'. Using the correction function α given by the Equivalence Lemma, this shows that

$$\begin{aligned}
[\![\mathcal{B}]\!](w) &\leq \mathsf{cost}_{hB}(\rho) && \text{(By definition of cost of a B-automaton)}\\
&\leq \alpha\left(\mathsf{cost}^*_{hB}(\rho)\right) && \text{(By equivalence lemma)}\\
&\leq \alpha\left(\mathsf{cost}(\tilde{\zeta}(w))\right). && \text{(By ($\star$))}
\end{aligned}$$

Finally it is sufficient to show that $\mathsf{cost}(\tilde{\zeta}(w)) \leq [\![\mathcal{B}]\!](w)$. One can prove by induction on $|w|$ that for the unique run $\rho' = \tilde{\zeta}(w)$ of $\mathcal{B}'$ that ends in state $S \subseteq Q$ the following holds:

$$\text{for any run ρ of $\mathcal{B}$ on w ending in q} : \pi_{k+1}(\mathsf{val}^*(\pi_3(\rho_q))) \leq \pi_{k+1}(\mathsf{val}^*(\pi_3(\rho)))$$

This readily implies that

$$\begin{aligned}
\mathsf{cost}(\tilde{\zeta}(w)) &\leq \inf_{q \in F} \pi_{k+1}(\mathsf{val}^*(\pi_3(\rho_q))) && \text{(By definition of ζ)}\\
&\leq \inf_{q \in F} \pi_{k+1}(\mathsf{val}^*(\pi_3(\rho))) && \text{(By previous claim)}\\
&\leq \inf_{q \in F} \mathsf{cost}_{hB}(\mathsf{val}(\pi_3(\rho))) \quad = [\![\mathcal{B}]\!](w) && \text{(Definition of cost of a B-automaton)}
\end{aligned}$$

This concludes the correctness proof. □

References

1. Sebastian Bala. Regular language matching and other decidable cases of the satisfiability problem for constraints between regular open terms. In *STACS*, volume 2996, pages 596–607, 2004.
2. Achim Blumensath, Martin Otto, and Mark Weyer. Boundedness of monadic second-order formulae over finite words. In *36th ICALP*, pages 67–78, July 2009.
3. Achim Blumensath, Martin Otto, and Mark Weyer. Decidability results for the boundedness problem. Available online, 2012.
4. Mikolaj Bojanczyk. Weak MSO with the unbounding quantifier. *Theory Comput. Syst.*, 48(3):554–576, 2011.
5. Mikolaj Bojańczyk and Thomas Colcombet. Bounds in ω-regularity. In *LICS 06*, pages 285–296, 2006.
6. Mikolaj Bojanczyk and Szymon Torunczyk. Deterministic automata and extensions of weak mso. In *FSTTCS*, pages 73–84, 2009.
7. Mikolaj Bojańczyk and Szymon Toruńczyk. Weak MSO+U over infinite trees. In *STACS*, 2012.
8. Michael Vanden Boom. Weak cost monadic logic over infinite trees. In *MFCS*, pages 580–591, 2011.
9. Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *Automata, languages and programming. Part II*, volume 5556 of *Lecture Notes in Comput. Sci.*, pages 139–150. Springer, Berlin, 2009.
10. Thomas Colcombet. Forms of determinism for automata. In *STACS*, 2012. Invited lecture, to appear.
11. Thomas Colcombet. Regular cost functions, part I: logic and algebra over words. Special issue of ICALP09, to appear, 2012.
12. Thomas Colcombet and Christof Löding. The nesting-depth of disjunctive μ-calculus for tree languages and the limitedness problem. In *Computer science logic*, volume 5213 of *Lecture Notes in Comput. Sci.*, pages 416–430. Springer, Berlin, 2008.

13. Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *Automata, languages and programming. Part II*, volume 5126 of *Lecture Notes in Comput. Sci.*, pages 398–409. Springer, Berlin, 2008.
14. Thomas Colcombet and Christof Löding. Regular cost functions over finite trees. In *LICS*, pages 70–79, 2010.
15. Kosaburo Hashiguchi. Limitedness theorem on finite automata with distance functions. *J. Comput. Syst. Sci.*, 24(2):233–244, 1982.
16. Kosaburo Hashiguchi. Relative star height, star height and finite automata with distance functions. In *Formal Properties of Finite Automata and Applications*, pages 74–88, 1988.
17. Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *CSL*, volume 4207 of *Lecture Notes in Computer Science*, pages 395–410. Springer, 2006.
18. Daniel Kirsten. Desert automata and the finite substitution problem. In *STACS*, volume 2996, pages 305–316, 2004.
19. Daniel Kirsten. Distance desert automata and the star height problem. *ITA*, 39(3):455–509, 2005.
20. Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. In *ICALP*, volume 623 of *Lecture Notes in Computer Science*, pages 101–112. Springer, 1992.
21. Denis Kuperberg and Michael Vanden Boom. Quasi-weak cost automata: A new variant of weakness. In *FSTTCS*, pages 66–77, 2011.
22. Marcel Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2-3):245–270, 1961.
23. Imre Simon. Recognizable sets with multiplicities in the tropical semiring. In *MFCS*, volume 324 of *Lecture Notes in Computer Science*, pages 107–120. Springer, 1988.
24. Robert S. Streett and E. Allen Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Inf. Comput.*, 81(3):249–264, 1989.