# On the Expressive Power of Cost Logics over Infinite Words*

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

[1] LIAFA/CNRS/Université Paris 7, Denis Diderot, France
denis.kuperberg@liafa.jussieu.fr
[2] Department of Computer Science, University of Oxford, England
michael.vandenboom@cs.ox.ac.uk

**Abstract.** Cost functions are defined as mappings from a domain like words or trees to $\mathbb{N} \cup \{\infty\}$, modulo an equivalence relation $\approx$ which ignores exact values but preserves boundedness properties. Cost logics, in particular cost monadic second-order logic, and cost automata, are different ways to define such functions. These logics and automata have been studied by Colcombet et al. as part of a "theory of regular cost functions", an extension of the theory of regular languages which retains robust equivalences, closure properties, and decidability. We develop this theory over infinite words, and show that the classical results FO = LTL and MSO = WMSO also hold in this cost setting (where the equivalence is now up to $\approx$). We also describe connections with forms of weak alternating automata with counters.

## 1 Introduction

The theory of regular cost functions is a quantitative extension to the theory of regular languages introduced by Colcombet [4]. Instead of languages being the centrepiece, functions from some set of structures (words or trees over some finite alphabet) to $\mathbb{N} \cup \{\infty\}$ are considered, modulo an equivalence relation $\approx$ which allows distortions but preserves boundedness over all subsets of the domain. Such functions are known as cost functions. This theory subsumes the classical theory of regular languages since a language can be associated with its characteristic function which maps every word (or tree) in the language to 0 and everything else to $\infty$; it is a strict extension since cost functions can count some behaviour within the input structure.

This theory grew out of two main lines of work: research by Hashiguchi [9], Kirsten [12], and others who were studying problems which could be reduced to whether or not some function was bounded over its domain (the most famous being the star height problem); and research by Bojańczyk and Colcombet [1,2]

* The full version of the paper can be found at http://www.liafa.jussieu.fr/~dkuperbe/. The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement 259454.

A. Czumaj et al. (Eds.): ICALP 2012, Part II, LNCS 7392, pp. 287–298, 2012.

on extensions of monadic second-order logic (MSO) with a quantifier $U$ which can assert properties related to boundedness.

Building on this work, this theory provides a framework which retains the robust closure properties, equivalences, and decidability results that regular languages enjoy. For instance, over finite words regular cost functions can be defined in terms of a cost logic (called cost monadic second-order logic, or CMSO), non-deterministic automata with counters (called $B/S$-automata), and algebra (called stabilisation semigroups). These relationships can be used to prove that it is decidable whether regular cost functions over finite words are equivalent, up to $\approx$ [4,2]. This decidability is also known over infinite words [3] and finite trees [6]. It is an important open problem on infinite trees: decidability of regular cost functions would imply decidability of the non-deterministic parity index [5].

In this paper, we develop this theory further by studying the expressivity of various cost logics over infinite words, namely the cost versions of linear temporal logic, first-order logic, monadic second-order logic, and weak monadic second-order logic, abbreviated CLTL, CFO, CMSO, and CWMSO, respectively. We also show connections with forms of weak alternating automata with counters.

### 1.1 Related Work and Motivation

Understanding the relationship between these cost logics and automata is desirable for application in model checking and other verification purposes. For instance, LTL can express "eventually some good condition holds (and this is true globally)". Unfortunately, it is also natural to want to bound the wait time before this good event occurs, but LTL provides no way to express this. Prompt LTL (introduced in [15]) can express this bounded wait time, and already gave rise to interesting decidability and complexity results. CLTL introduced in [13], is a strictly more expressive logic which can also count other types of events (like the number of occurences of a letter), while still retaining nice properties.

This research was also motivated by recent work which cast doubt as to whether the classical equivalences between logics would hold. For instance, the standard method for proving that MSO = WMSO on infinite words relies on McNaughton's Theorem, which states that deterministic Muller automata capture all regular languages of infinite words (WMSO can describe acceptance in terms of partial runs of the deterministic Muller automaton). However, no deterministic model is known for regular cost functions (even over finite words) [4], so this route for proving CMSO = CWMSO was closed to us.

In [18,14], similar logics were explored in the context of infinite trees rather than infinite words. There it was shown that CMSO is strictly more expressive than CWMSO, and that Rabin's famous characterization of WMSO (in terms of definability of the language and its complement using non-deterministic Büchi automata) fails in the cost setting. Based on this previous work, the relationship between these various cost logics over infinite words was not clear.

### 1.2 Notation

We fix a finite alphabet $\mathbb{A}$, writing $\mathbb{A}^*$ (respectively, $\mathbb{A}^\omega$) for the set of finite (respectively, infinite) words over $\mathbb{A}$. For $a \in \mathbb{A}$, $|u|_a$ denotes the number of $a$-labelled positions in $u$, and $|u|$ denotes the length of the word (the length function is noted $|\cdot|$). We write $\mathbb{N}_\infty$ for $\mathbb{N} \cup \{\infty\}$. By convention, $\inf \emptyset = \infty$ and $\sup \emptyset = 0$.

We briefly define $\approx$, see [4] for details. Let $E$ be a set (usually $\mathbb{A}^\omega$) and let $f, g : E \to \mathbb{N}_\infty$. For $X \subseteq E$, $f(X) := \{f(e) : e \in E\}$. We say $f(X)$ is *bounded* if there is $n \in \mathbb{N}$ such that $\inf f(X) \leq n$ (in particular the set $\{\infty\}$ is unbounded). We say $f \approx g$ if for all $X \subseteq E$, $f(X)$ is bounded if and only if $g(X)$ is bounded. For example, $|\cdot|_a \approx 2|\cdot|_a$ but $|\cdot|_a \not\approx |\cdot|_b$. A *cost function* $F$ is an equivalence class of $\approx$, but in practice will be represented by one of its elements $f : E \to \mathbb{N}_\infty$.

### 1.3 Contributions

In this paper, we show that the classical equivalences of FO = LTL and MSO = WMSO hold in this cost setting. This supports the idea that the cost function theory is a coherent quantitative extension of language theory. We state the full theorems now, and will introduce the precise definitions in later sections.

The first set of results shows that CFO and CLTL are equivalent, up to $\approx$:

**Theorem 1.** *For a cost function $f$ over infinite words, it is effectively equivalent for $f$ to be recognized by a:*

- *CFO sentence;*
- *very-weak B-automaton;*
- *very-weak B-automaton with one counter;*
- *CLTL formula.*

The second set of results shows that CMSO (which is strictly more expressive than CFO) is equivalent to CWMSO (again, up to $\approx$):

**Theorem 2.** *For a cost function $f$ over infinite words, it is effectively equivalent for $f$ to be recognized by a:*

- *CMSO sentence;*
- *non-deterministic B/S-Büchi automaton;*
- *quasi-weak B-automaton;*
- *weak B-automaton;*
- *CWMSO sentence.*

## 2 Cost Logics

### 2.1 Cost First-Order Logic

We extend first-order logic in order to define cost functions instead of languages. It is called *cost first-order logic*, or CFO. Formulas are defined by the grammar

$$\varphi := a(x) \mid x = y \mid x < y \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \forall x.\varphi \mid \forall^{\leq N} x.\varphi$$

where $a \in \mathbb{A}$, and $N$ is a unique free variable ranging over $\mathbb{N}$. The new predicate $\forall^{\leq N} x.\varphi$ means that $\varphi$ has to be true for all $x$, except for at most $N$ "mistakes".

In all of the cost logics, we want to preserve the intuition that increasing the value for $N$ makes it easier to satisfy the formula. In order to make this clear, we will define the logics without negation. An equivalent definition with negation could be given, with the restriction that quantifiers $\forall^{\leq N}.\varphi$ always appear positively (within the scope of an even number of negations); the grammar above could then be viewed as the result of pushing these negations to the leaves.

Given a word $u \in \mathbb{A}^\omega$, an integer $n \in \mathbb{N}$, and a closed formula $\varphi$, we say that $(u, n)$ is a model of $\varphi$ (noted $(u, n) \models \varphi$) if $\varphi$ is true on the structure $u$, with $n$ as value for $N$. If $x$ is a free variable in $\varphi$, then we also need to provide a value for $x$, and we can write $(u, n, i) \models \varphi$, where $i \in \mathbb{N}$ is the valuation for $x$. Note that $\forall^{\leq N} x.\varphi(x)$ is true with $n$ for $N$ if and only if there exists $X \subseteq \mathbb{N}$ such that the cardinality of $X$ is at most $n$ and for all $i \in \mathbb{N} \setminus X$, $(u, n, i) \models \varphi(x)$. We then associate a cost function $[\![\varphi]\!] : \mathbb{A}^\omega \to \mathbb{N}_\infty$ to a closed CFO-formula $\varphi$ by

$$[\![\varphi]\!](u) := \inf \{n \in \mathbb{N} : (u, n) \models \varphi\}.$$

We say $[\![\varphi]\!]$ is the cost function *recognized* by $\varphi$.

For instance for $\varphi = \forall^{\leq N} x. \bigvee_{b \neq a} b(x)$, we have $[\![\varphi]\!](u) = |u|_a$.

### 2.2 Cost Monadic Second-Order Logic

We define *cost monadic second-order logic* (CMSO) as an extension of CFO, where we can quantify over sets of positions. The syntax of CMSO is therefore

$$\varphi := a(x) \mid x = y \mid x < y \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \forall x.\varphi \mid \forall^{\leq N} x.\varphi \mid \exists X.\varphi \mid \forall X.\varphi \mid x \in X \mid x \notin X.$$

The semantic of CMSO-formulas is defined in the same way as for CFO: if $u \in \mathbb{A}^\omega$ and $n \in \mathbb{N}$, we say that $(u, n) \models \varphi$ if $\varphi$ is true on the structure $u$, with $n$ as value for $N$. We then define $[\![\varphi]\!](u) := \inf \{n \in \mathbb{N} : (u, n) \models \varphi\}$.

CMSO was introduced in [4,18] in a slightly different way, as an extension of MSO with predicates $|X| \leq N$ (for a second-order variable $X$) which appeared positively. The two definitions are equivalent in terms of expressive power.

### 2.3 Cost Weak Monadic Second-Order Logic

*Cost weak monadic second-order logic* (CWMSO) was introduced in [18] over infinite trees. The syntax of CWMSO is defined as in CMSO, but the semantics are changed so second-order quantifications range only over finite sets.

CWMSO retains nice properties of WMSO, such as easy dualization, translation to non-deterministic automata models, and equivalence with a form of weak alternating automata with counters. We will only be interested here in cost functions on infinite words, which are a particular case of infinite trees.

### 2.4 Link with Languages

We can remark that in particular, any FO (resp. MSO) formula $\varphi$ can be considered as a CFO (resp. CMSO) formula. In this case if $L$ was the language defined by $\varphi$, then as a cost formula $\varphi$ recognizes the cost function $[\![\varphi]\!] = \chi_L$, defined by $\chi_L(u) = \begin{cases} 0 & \text{if } u \in L \\ \infty & \text{if } u \notin L \end{cases}$.

**Lemma 1.** *If $\varphi$ is a CFO (resp. CMSO) formula such that $[\![\varphi]\!] \approx \chi_L$ for some language $L$, then $L$ is definable in FO (resp. MSO).*

*Proof.* $[\![\varphi]\!] \approx \chi_L$ means that there is a $n \in \mathbb{N}$ such that for all $u \in L$, $[\![\varphi]\!](u) \leq n$, and for all $u \notin L$, $[\![\varphi]\!](u) = \infty$. In particular, all predicates $\forall^{\leq N} x.\psi$ in $\varphi$ must be verified with $N = n$, when the word is in the language. So we can replace these predicates by the formula $\exists x_1, x_2, \ldots, x_n. \forall x. (\psi \vee \bigvee_{i \in [1,n]} x = x_i)$, expressing that we allow $n$ errors, marked by the $x_i$'s. The resulting formula will be pure FO (resp. MSO), and will recognize $L$.

**Corollary 1.** *CMSO is strictly more expressive than CFO.*

*Proof.* Choose $L$ which is MSO-definable but not FO-definable, like $(aa)^* b^\omega$. By Lemma 1, $\chi_L$ is not CFO-definable, but by the first remark it is CMSO-definable.

### 2.5 Cost Linear Temporal Logic

We now define a cost version of a linear temporal logic, CLTL. This was first introduced in [13] over finite words (and with a slightly different syntax). Formulas are defined by the grammar

$$\varphi := a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \mathbf{R} \varphi \mid \varphi \mathbf{U} \varphi \mid \varphi \mathbf{U}^{\leq N} \varphi$$

where $a$ ranges over $\mathbb{A}$, and $N$ is a unique free variable ranging over $\mathbb{N}$. We say that $(u, n, i) \models \varphi$ if $u$ is a model for $\varphi$, where $n \in \mathbb{N}$ is the value for $N$ and $i \in \mathbb{N}$ is the position of $u$ from which the formula is evaluated. If $i$ is not specified, the default value is 0. The semantics are defined as usual, with $(u, n, i) \models \psi_1 \mathbf{U} \psi_2$ if there exists $j > i$ such that $(u, n, j) \models \psi_2$ and for all $i < k < j$, $(u, n, k) \models \psi_1$. Similarly, $(u, n, i) \models \psi_1 \mathbf{U}^{\leq N} \psi_2$ if there exists $j > i$ such that $(u, n, j) \models \psi_2$ and $(u, n, k) \models \psi_1$ for all but $n$ positions $k$ in $[i+1, j-1]$. Likewise, $(u, n, i) \models \psi_1 \mathbf{R} \psi_2$ if either $(u, n, j) \models \psi_1$ for all $j > i$ or $(u, n, i) \models \psi_2 \mathbf{U} (\psi_1 \wedge \psi_2)$. We define $[\![\varphi]\!](u) := \inf \{n \in \mathbb{N} : (u, n) \models \varphi\}$.

Notice that we write $\mathbf{U}$ for the next-until operator. From this, the next operator $\mathbf{X}$ can be defined, and the "large" variants of $\mathbf{U}$, $\mathbf{R}$, $\mathbf{U}^{\leq N}$ operators, which take into account the current position, can be defined as well (see [7] for more information), and will be noted $\overline{\mathbf{U}}$, $\overline{\mathbf{R}}$, $\overline{\mathbf{U}^{\leq N}}$. We also use the standard abbreviations $\mathbf{F}, \mathbf{G}$ for "Eventually" and "Always". We can define the quantitative release $\mathbf{R}^{\leq N}$ by $\psi \mathbf{R}^{\leq N} \varphi \equiv \varphi \mathbf{U}^{\leq N} (\psi \vee \mathbf{G} \varphi)$, and the quantitative always $\mathbf{G}^{\leq N} \varphi \equiv \mathsf{false} \mathbf{R}^{\leq N} \varphi$.

In Sect. 4, we will also use the past variants $\mathbf{S}, \mathbf{Q}, \mathbf{Y}, \mathbf{P}, \mathbf{H}$ of $\mathbf{U}, \mathbf{R}, \mathbf{X}, \mathbf{F}, \mathbf{G}$, respectively (and their quantitative extensions).

# 3 Cost Automata on Infinite Words

## 3.1 $B$-Valuation

Similar to the logic, the automata considered in this paper define functions from $\mathbb{A}^\omega$ to $\mathbb{N}_\infty$. The valuation is based on the classical Büchi acceptance condition and a finite set of counters $\Gamma$.

A counter $\gamma$ is initially assigned value 0 and can be *incremented and checked* $\mathtt{ic}$, left unchanged $\varepsilon$, or *reset* $\mathtt{r}$ to 0. Given an infinite word $u_\gamma$ over the alphabet $\mathbb{B} := \{\mathtt{ic}, \varepsilon, \mathtt{r}\}$, we define $val_B(u_\gamma) \in \mathbb{N}_\infty$, which is the supremum of all checked values of $\gamma$. For instance $val_B((\mathtt{ic}\varepsilon\mathtt{icr})^\omega) = 2$ and $val_B(\mathtt{icric}^2\mathtt{ric}^3\mathtt{r}\dots) = \infty$. In the case of a finite set of counters $\Gamma$ and a word $u$ over the alphabet $\{\mathtt{ic}, \varepsilon, \mathtt{r}\}^\Gamma$, $val_B(u) := \max_{\gamma\in\Gamma} val_B(pr_\gamma(u))$ ($pr_\gamma(u)$ is the $\gamma$-projection of $u$).

The set $\mathbb{C} := \mathbb{B}^\Gamma$ is the alphabet of counter actions, that describe the actions on every counter $\gamma \in \Gamma$.

## 3.2 $B$- and $S$-Automata

An *alternating B-Büchi automaton* $\mathcal{A} = \langle Q, \mathbb{A}, F, q_0, \Gamma, \delta\rangle$ on infinite words has a finite set of states $Q$, alphabet $\mathbb{A}$, initial state $q_0 \in Q$, a set of Büchi states $F$, a finite set $\Gamma$ of $B$-counters, and a transition function $\delta : Q \times \mathbb{A} \to \mathcal{B}^+(\mathbb{C} \times Q)$ (where $\mathcal{B}^+(\mathbb{C} \times Q)$ is the set of positive boolean combinations, written as a disjunction of conjunctions, of elements $(c_i, q_i) \in \mathbb{C} \times Q$).

A run of $\mathcal{A}$ on $u = a_0a_1 \cdots \in \mathbb{A}^\omega$ is an infinite labelled tree $R = (r, c)$ with $dom(R) \subseteq \mathbb{N}^*$ (words over $\mathbb{N}$), $r : dom(R) \to Q$ and $c : (dom(R) \setminus \{\epsilon\}) \to \mathbb{C}$ with

- $r(\epsilon) = q_0$;
- if $x \in dom(R)$ and $\delta(r(x), a_{|x|}) = \varphi$, then there is some disjunct $(c_1, q_1) \wedge \dots \wedge (c_k, q_k)$ in $\varphi$ such that for all $1 \le j \le k$, $x \cdot j \in dom(R)$, $r(x \cdot j) = q_j$ and $c(x \cdot j) = c_j$, and for all $j > k$, $x \cdot j \notin dom(R)$.

We say a run is accepting if for every branch $\pi$ in $R = (r, c)$, there are infinitely many positions $x$ such that $r(x) \in F$.

The behaviour of an alternating automaton can be viewed as a game between two players: Min and Max. Min is in charge of the disjunctive choices and Max chooses a conjunct in the clause picked by Min. Therefore, a run tree fixes all of Min's choices and the branching corresponds to Max's choices.

A *play* is a particular branch of the run: it is the sequence of states and transitions taken according to both players' choices. A *strategy* for some player is a function that gives the next choice of this player, given the history of the play. Notice that a run describes exactly a strategy for Min.

We assign values to runs as follows. Given a branch $\pi = x_0x_1 \dots$ in a run $R$, the value of $\pi$ is $val_B(\pi) := val_B(c(x_1)c(x_2)\dots)$. Then $val_B(R)$ is the supremum over all branch values. We call a run of value at most $n$ an $n$-run.

The *B-semantic* of a $B$-automaton $\mathcal{A}$ is $[\![\mathcal{A}]\!]_B : \mathbb{A}^\omega \to \mathbb{N}_\infty$ defined by

$$[\![\mathcal{A}]\!]_B(u) := \inf\{val_B(R) : R \text{ is an accepting run of } \mathcal{A} \text{ on } u\}.$$

The $B$-semantic minimizes the value over $B$-accepting runs.

If $\delta$ only uses disjunctions, then we say the automaton is *non-deterministic*. In this case, a run tree is just a single branch, and only Min has choices to make. The run is accepting if there are infinitely many Büchi states on its unique branch $\pi$, and its value is the value of $\pi$. In this case, it can be useful to look at the labelled run-DAG (for Directed Acyclic Graph) $G$ of the automaton over some word $u$: the set of nodes is $Q \times \mathbb{N}$, and the edges are of the form $(p, i) \xrightarrow{c} (q, i+1)$,where $(c, q)$ is a disjunct in $\delta(p, a_i)$. Runs of $\mathcal{A}$ over $u$ are in bijection with paths in $G$.

There exists a dual version of $B$-automata, namely $S$-automata, where the semantic is reversed: a run remembers the lowest checked value, and this is maximized over all runs. Switching between non-deterministic $B$- and $S$-automata corresponds to a complementation procedure on languages. See [4] for details.

Intuitively, non-deterministic $B$-automata are used to formulate boundedness problems (like star-height), while non-deterministic $S$-automata are used to answer these problems because they can more easily witness unboundedness.

We will be particularly interested here in non-deterministic $B$-Büchi (resp. $S$-Büchi) automata, abbreviated $B$-NBA (resp. $S$-NBA).

**Weak Automata.** We will say that a $B$-automaton $\mathcal{A} = \langle Q, \mathbb{A}, F, q_0, \Gamma, \delta \rangle$ is a *weak alternating $B$-automaton* ($B$-WAA) if it is an alternating $B$-Büchi automaton such that the state-set $Q$ can be partitioned into $Q_1, \ldots, Q_k$ and there is a partial order $<$ on these partitions satisfying:

- for all $q, q' \in Q_i$, $q \in F$ if and only if $q' \in F$;
- if $q_j \in Q_j$ is reachable from $q_i \in Q_i$ via $\delta$, then $Q_j \leq Q_i$.

This means no cycle visits both accepting and rejecting partitions, so an accepting run must stabilize in an accepting partition on each path in the run tree.

**Theorem 3 ([18]).** *On infinite trees, $B$-WAA and CWMSO recognize the same class of cost functions, namely weak cost functions.*

This theorem holds in particular on infinite words. Notice that unlike in the classical case, WCMSO does not characterize the cost functions recognized by both non-deterministic $B$-Büchi and non-deterministic $S$-Büchi automata. The class that enjoy this Rabin-style characterization is the quasi-weak class, which strictly contains the weak class, see [14] for more details.

We will show that as in the case of languages, CMSO and CWMSO have the same expressive power on infinite words. It means that the regular class, the quasi-weak class, and the weak class collapse on infinite words. The cost functions definable by any of the automata or logics in Theorem 2 are called *regular cost functions over infinite words*.

**Very-Weak Automata.** A *very-weak alternating $B$-automaton* ($B$-VWAA) is a $B$-WAA with the additional requirement that each partition is a singleton. That is, there can be no cycle containing 2 or more states. The name follows [7], but these automata are also sometimes known as linear alternating automata, since the condition corresponds to the existence of a linear ordering on states.

## 4 First-Order Fragment

In this section, we aim to prove Theorem 1.

The classical equivalence of FO and LTL is known as Kamp's Theorem [11]. Converting from CLTL to CFO is standard, since we can describe the meaning of the CLTL operators in CFO, so we omit this part. However, a number of new issues arise in the translation from CFO to CLTL, so we concentrate on this translation in Sect. 4.1.

We then show the connection with $B$-VWAA. Again, one direction (from CLTL to $B$-VWAA) is straightforward and only requires one counter (this is adapted from [17]). Moving from $B$-VWAA (potentially with multiple counters) to CLTL is more interesting, so we describe some of the ideas behind that construction in Sect. 4.2. It uses ideas from [7] but requires some additional work to structure the counter operations in a form that is easily expressible using CLTL.

*Example 1.* We give an example of a cost function recognizable by a CLTL formula, CFO sentence, and $B$-VWAA.

Let $\mathbb{A} = \{a, b, c\}$ and $f(u) = \begin{cases} |u|_a & \text{if } |u|_b = \infty \\ \infty & \text{if } |u|_b < \infty \end{cases}$.

Then $f$ is recognized by the CLTL-formula $\varphi = (\mathbf{G}^{\leq N}(b \vee c)) \wedge (\mathbf{GF}b)$ and by the CFO-sentence $\psi = [\forall^{\leq N} x.(b(x) \vee c(x))] \wedge [\forall x.\exists y.(x < y \wedge b(x))]$. $f$ is also recognized by a 3-state $B$-VWAA: it deterministically counts the number of $a$'s, while Player Max has to guess a point where there is no more $b$ in the future, in order to reject the input word. If the guess is wrong and there is one more $b$, or if the guess is never made, then the automaton stabilizes in an accepting state.

### 4.1 CFO to CLTL

Instead of trying to translate CFO directly into CLTL, we first translate a CFO formula into a CLTL formula extended with past operators $\mathbf{Q}, \mathbf{S}, \mathbf{S}^{\leq N}$ (the past versions of $\mathbf{R}, \mathbf{U}, \mathbf{U}^{\leq N}$) and then show how to eliminate these past operators. Let $\text{CLTL}_\text{P}$ be CLTL extended with these past operators.

A $\text{CLTL}_\text{P}$-formula is *pure past* (resp. *pure future*) if it uses only temporal operators $\mathbf{Q}, \mathbf{S}, \mathbf{S}^{\leq N}$ (resp. $\mathbf{R}, \mathbf{U}, \mathbf{U}^{\leq N}$) and all atoms are under the scope of at least one of these operators. A formula is *pure present* if it does not contain temporal operators. Hence, a pure past (resp. present, future) formula depends only on positions before (resp. equal to, after) the current position in the word. A formula is *pure* if it is pure past, pure present, or pure future. It turns out any $\text{CLTL}_\text{P}$ formula can be translated into a boolean combination of pure formulas.

**Theorem 4 (Separation Theorem).** *$\text{CLTL}_\text{P}$ has the separation property, i.e. every formula is equivalent to a boolean combination of pure formulas.*

The proof is technical and requires an analysis of a number of different cases of past operators nested inside of future operators (and vice versa). It uses ideas from [8], but new behaviours arise in the cost setting, and have to be treated carefully. The proof proceeds by induction on the *junction depth*, the maximal

number of alternations of nested past and future operators, and on the quantifier rank. Each induction step introduces some distortion of the value (the number of mistakes can be squared), but because the junction depth and quantifier rank are bounded in the formula, we end up with an equivalent formula.

We illustrate the idea with an example.

*Example 2.* Let $\mathbb{A} := \{a, b, c, d\}$. Consider the $\mathrm{CLTL_P}$ formula $\varphi = (b\mathbf{U}^{\leq N}c)\mathbf{S}a$. Then $\varphi$ is equivalent to

$$[(b\mathbf{S}^{\leq N}(c \vee a))\mathbf{S}a] \wedge [(b\overline{\mathbf{U}^{\leq N}}c) \vee (\mathbf{Y}a)]$$

which is a boolean combination of pure formulas. This formula factorizes the input word into blocks separated by $c$'s, since the last $a$ in the past. The first conjunct checks that each block is missing at most $N$ $b$'s. The second conjunct checks that at the previous position, we had either $a$ or $b\mathbf{U}^{\leq N}c$.

We can now prove the desired translation from CFO to CLTL. The proof is adapted from [10]. It proceeds by induction on the quantifier rank of the formula, and makes use of the Separation Theorem. We have to take care of the new quantitative quantifiers, but no problem specific to cost functions arises here.

**Proposition 1.** *Every CFO-formula can be effectively translated into an equivalent CLTL-formula.*

### 4.2 *B*-VWAA to CLTL

We use ideas from [17, Theorem 6]. Unlike the classical setting, we must first convert the $B$-VWAA into a more structured form. In this section, we write $\mathbb{C}$ for the set of *hierarchical counter actions* on counters $[1, k]$ such that $\mathtt{ic}_j$ (resp. $\mathtt{r}_j$) performs $\mathtt{ic}$ (resp. $\mathtt{r}$) on counter $j$, resets counters $j' < j$, and leaves $j' > j$ unchanged. We say a $B$-VWAA is *CLTL-like* if the counters are hierarchical, $\delta(q, a)$ is in disjunctive normal form, each disjunct has at most one conjunct with state $q$, and all conjuncts with state $q' \neq q$ have counter action $\mathtt{r}_k$.

**Lemma 2.** *Let $\mathcal{A}$ be a $B$-VWAA. Then there is a $B$-VWAA $\mathcal{A}'$ which is CLTL-like and satisfies $[\![\mathcal{A}]\!] \approx [\![\mathcal{A}']\!]$.*

We can then describe a low value run using a CLTL-formula.

**Proposition 2.** *Let $\mathcal{A}$ be a $B$-VWAA with $k$ counters which is CLTL-like. For all $\varphi \in \mathcal{B}^+(Q)$, there is a CLTL formula $\theta(\varphi)$ such that $[\![\mathcal{A}_\varphi]\!] \approx [\![\theta(\varphi)]\!]$ where $\mathcal{A}_\varphi$ is the automaton $\mathcal{A}$ starting from the states described in $\varphi$.*

*Proof.* The proof is by induction on $|Q|$. The case $|Q| = 0$ is trivial.

Let $|Q| > 0$ and let $q$ be the highest state in the very-weak ordering. Given some $\varphi \in \mathcal{B}^+(\mathbb{C} \times Q)$, we can treat each element separately and then combine using the original boolean connectives. For elements $q' \neq q$, we can immediately apply the inductive hypothesis.

For an element $q$, we first write formulas $\theta_{q,c}$ for $c \in \mathbb{C}$ which express the requirements when the automaton selects a disjunct which has one copy which stays in state $q$ and performs operation $c$ (there is only one such operation since $\mathcal{A}$ is CLTL-like). Likewise, we write a formula $\theta_{q,\text{exit}}$ which describes the requirements when $\mathcal{A}$ chooses a disjunct which leaves $q$. These formulas do not involve $q$ so can be obtained by applying the inductive hypothesis.

While the play stays in state $q$, we must ensure that transitions with increments are only taken a bounded number of times before resets. For a particular counter $\gamma$, this behaviour is approximated by

$$\theta_{q,\text{cycle},\gamma} := (\bigvee_{\gamma' \geq \gamma} \theta_{q,\mathtt{r}_{\gamma'}} \vee \theta_{q,\text{exit}})\ \overline{\mathbf{R}^{\leq N}}\ (\bigvee_{c < \mathtt{ic}_\gamma} \theta_{q,c}).$$

Putting this together for all $\gamma \in [1,k]$, $\theta_{q,\text{cycle}} := \bigvee_{c\in\mathbb{C}} \theta_{q,c} \wedge \bigwedge_{\gamma\in[1,k]} \theta_{q,\text{cycle},\gamma}$. Finally, this gets wrapped into a statement which ensures correct behaviour in terms of accepting states (i.e. if $q \notin F$ then the play cannot stay forever in $q$):

$$\theta(q) := \begin{cases} \theta_{q,\text{cycle}}\ \overline{\mathbf{U}}\ \theta_{q,\text{exit}} & \text{if } q \notin F \\ \theta_{q,\text{exit}}\ \overline{\mathbf{R}}\ \theta_{q,\text{cycle}} & \text{if } q \in F \end{cases}.$$

By combining the translation from a $B$-VWAA with multiple counters to CLTL, and then the translation to a $B$-VWAA (which uses only one counter) we see that adding counters to $B$-VWAA does not increase expressivity.

**Corollary 2.** *Every $B$-VWAA with $k$ counters is equivalent to a $B$-VWAA with one counter.*

## 5 Expressive Completeness of CWMSO

We aim in this part at proving Theorem 2.

The translation from CWMSO to CMSO is standard (since finiteness is expressible in MSO). Likewise, the connection between CMSO and $B$- and $S$-automata was proven in [4] for finite words, and its extension to infinite words (and $B$-NBA and $S$-NBA automata) is known [3].

As a result, we concentrate on the remaining translations. As mentioned in the introduction, because there is no deterministic model for cost automata, we could not prove that CWMSO = CMSO using the standard method. In this section, we describe an alternative route, which goes via weak automata. Using ideas from [16], we show how to move from $B$-NBA to $B$-WAA. This gives an idea about the issues involved in analyzing alternating cost automata over infinite words. We can then use Theorem 3 to move from $B$-WAA to CWMSO.

### 5.1 $B$-NBA to $B$-WAA

**Theorem 5.** *For all $B$-NBA $\mathcal{A}$ with $n$ states and $k$ counters, we can construct an equivalent $B$-WAA $W$ with $O(n^2 4^k)$ states and $k$ counters.*

*Proof.* We first transform the $B$-NBA $\mathcal{A}$ into an equivalent $B$-NBA $\mathcal{B}$ in the following normal form: every transition leaving a Büchi state resets all counters of $\mathcal{B}$. The principle is the following: because we work on infinite words, $\mathcal{B}$ can choose an $n$-run of $\mathcal{A}$ and guess for each counter if there will be infinitely many resets, or finitely many increments. In the first case, $\mathcal{B}$ always delays its Büchi states until the next reset. In the second case, $\mathcal{B}$ waits until the last increment, and then add resets on Büchi states. This results in a slightly different function, but still equivalent up to $\approx$. Notice that this transformation cannot be achieved on infinite trees, which is why this result does not hold for trees (see [14]).

Then we use ideas from [16]: we analyze the run DAG of $\mathcal{B}$ and assign ranks to its nodes. Intuitively, these ranks describe how far each node is from a Büchi run. More precisely, for any $n \in \mathbb{N}$ and $u \in \mathbb{A}^\omega$, it is possible to assign a finite rank to every node if and only if there is no $n$-run of $\mathcal{B}$ on $u$.

Therefore, we can design a $B$-WAA that allows Player Min to play transitions of $\mathcal{B}$, and the opponent to guess ranks in order to prove that there is no low value run. This way, if there is an $n$-run of $\mathcal{B}$ on $u$, playing this run is a strategy of value $n$ for Player Min in $W$. Conversely, if there is no $n$-run of $\mathcal{B}$ on $u$, then we can prove that Player Min cannot have a strategy of value $n$ in $\mathcal{B}$: if he plays in a way that counters stay below $n$, then the run will not be Büchi, and Player Max can guess ranks to prove this. The automaton $W$ is defined so that such a play stabilizes in a rejecting partition of $W$. This shows that $[\![W]\!] = [\![\mathcal{B}]\!] \approx [\![\mathcal{A}]\!]$.

A normal form is needed to make it possible to look for runs of $\mathcal{B}$ that are simultaneously Büchi and low valued. If the automaton is not in normal form, the independence of these two conditions prevents us from defining ranks properly.

## 6 Conclusion

We lifted various equivalence results on infinite words from languages to cost functions. The proofs needed to take care of new behaviours specific to this quantitative setting. These results show that the classical definitions of logics and automata have been extended in a coherent way to the cost setting, and provide further evidence that the theory of regular cost functions is robust.

We showed that the weak cost functions on infinite words enjoy the same nice properties as in the case of languages. This is in contrast to the case of trees (see [14]), where some classical properties of weak languages only held for the larger class of quasi-weak cost functions.

We also studied the first-order fragment which gave rise to an unexpected result: very-weak $B$-automata need only one counter to reach their full expressivity. We did not develop here the algebra side of the first-order fragment as it was done in [13], but if this result can be lifted to infinite words (which we think is the case), it would imply algebraic characterization by aperiodic stabilization semigroups, and hence decidablity of membership for the first-order fragment.

**Acknowledgments.** We would like to thank the referees for their comments, and Thomas Colcombet for making this joint work possible.

## References


1. Bojańczyk, M.: A Bounding Quantifier. In: Marcinkowski, J., Tarlecki, A. (eds.) CSL 2004. LNCS, vol. 3210, pp. 41–55. Springer, Heidelberg (2004)
2. Bojańczyk, M., Colcombet, T.: Bounds in w-regularity. In: LICS, pp. 285–296. IEEE Computer Society (2006)
3. Colcombet, T.: Personal communication
4. Colcombet, T.: The Theory of Stabilisation Monoids and Regular Cost Functions. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part II. LNCS, vol. 5556, pp. 139–150. Springer, Heidelberg (2009)
5. Colcombet, T., Löding, C.: The Non-deterministic Mostowski Hierarchy and Distance-Parity Automata. In: Aceto, L., Damgaard, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 398–409. Springer, Heidelberg (2008)
6. Colcombet, T., Löding, C.: Regular cost functions over finite trees. In: LICS, pp. 70–79. IEEE Computer Society (2010)
7. Diekert, V., Gastin, P.: First-order definable languages. In: Flum, J., Grädel, E., Wilke, T. (eds.) Logic and Automata. Texts in Logic and Games, vol. 2, pp. 261–306. Amsterdam University Press (2008)
8. Gabbay, D.M., Hodkinson, I., Reynolds, M.: Temporal logic: mathematical foundations and computational aspects. Oxford Logic Guides, Clarendon Press (1994)
9. Hashiguchi, K.: Limitedness theorem on finite automata with distance functions. J. Comput. Syst. Sci. 24(2), 233–244 (1982)
10. Hodkinson, I.M., Reynolds, M.: Separation - past, present, and future. In: We Will Show Them!, vol. 2, pp. 117–142 (2005)
11. Kamp, H.W.: Tense Logic and the Theory of Linear Order. PhD thesis, Computer Science Department, University of California at Los Angeles, USA (1968)
12. Kirsten, D.: Distance desert automata and the star height problem. RAIRO - Theoretical Informatics and Applications 3(39), 455–509 (2005)
13. Kuperberg, D.: Linear temporal logic for regular cost functions. In: Schwentick, T., Dürr, C. (eds.) STACS. LIPIcs, vol. 9, pp. 627–636. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)
14. Kuperberg, D., Vanden Boom, M.: Quasi-weak cost automata: A new variant of weakness. In: Chakraborty, S., Kumar, A. (eds.) FSTTCS. LIPIcs, vol. 13, pp. 66–77. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)
15. Kupferman, O., Piterman, N., Vardi, M.Y.: From liveness to promptness. Formal Methods in System Design 34(2), 83–103 (2009)
16. Kupferman, O., Vardi, M.Y.: Weak alternating automata are not that weak. ACM Trans. Comput. Log. 2(3), 408–429 (2001)
17. Löding, C., Thomas, W.: Alternating Automata and Logics over Infinite Words (Extended Abstract). In: Watanabe, O., Hagiya, M., Ito, T., van Leeuwen, J., Mosses, P.D. (eds.) IFIP TCS 2000. LNCS, vol. 1872, pp. 521–535. Springer, Heidelberg (2000)
18. Vanden Boom, M.: Weak Cost Monadic Logic over Infinite Trees. In: Murlak, F., Sankowski, P. (eds.) MFCS 2011. LNCS, vol. 6907, pp. 580–591. Springer, Heidelberg (2011)