On the Expressive Power of Cost Logics over Infinite Words*

Denis Kuperberg[1], 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 ∞; 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.

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 φ 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 φ, we say that (u, n) is a model of φ (noted $(u, n) \models \varphi$) if φ is true on the structure u, with n as value for N. If x is a free variable in φ, 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 φ by

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

We say $[\![\varphi]\!]$ is the cost function *recognized* by φ.
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 φ 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 φ can be considered as a CFO (resp. CMSO) formula. In this case if L was the language defined by φ, then as a cost formula φ 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 φ 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 φ 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, χ_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 φ, 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 Γ.

A counter γ is initially assigned value 0 and can be *incremented and checked* $\mathtt{ic}$, left unchanged ε, or *reset* $\mathtt{r}$ to 0. Given an infinite word u_γ 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 γ. 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 Γ 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 γ-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 Γ 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 φ 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 π 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 π 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 n-run a run of value at most n.

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 δ 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 π, and its value is the value of π. 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 δ, 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 CLTL_P be CLTL extended with these past operators.

A CLTL_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 CLTL_P formula can be translated into a boolean combination of pure formulas.

Theorem 4 (Separation Theorem). *CLTL_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 CLTL_P formula $\varphi = (b\mathbf{U}^{\leq N}c)\mathbf{S}a$. Then φ 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 uses 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 φ.*

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 γ, 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. Lecture Notes in Computer Science, vol. 3210, pp. 41–55. Springer (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.E., Thomas, W. (eds.) ICALP (2). Lecture Notes in Computer Science, vol. 5556, pp. 139–150. Springer (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 (2). Lecture Notes in Computer Science, vol. 5126, pp. 398–409. Springer (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! (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. In: van Leeuwen, J., Watanabe, O., Hagiya, M., Mosses, P.D., Ito, T. (eds.) IFIP TCS. Lecture Notes in Computer Science, vol. 1872, pp. 521–535. Springer (2000)
18. Vanden Boom, M.: Weak cost monadic logic over infinite trees. In: Murlak, F., Sankowski, P. (eds.) MFCS. Lecture Notes in Computer Science, vol. 6907, pp. 580–591. Springer (2011)

Appendix

A More on Cost Functions

Non-decreasing functions $\mathbb{N} \to \mathbb{N}$ will be denoted by letters $\alpha, \beta, \ldots$, and will be extended to $\mathbb{N}_\infty$ by $\alpha(\infty) = \infty$. We call these *correction functions*.

Let E be any set, and $\mathcal{F}_E$ be the set of functions : $E \to \mathbb{N}_\infty$. For $f, g \in \mathcal{F}_E$ and α a correction function, we write $f \preccurlyeq_\alpha g$ if $f \leq \alpha \circ g$ (or if we are comparing single values $n, m \in \mathbb{N}$, $n \preccurlyeq_\alpha m$ if $n \leq \alpha(m)$). We write $f \approx_\alpha g$ if $f \preccurlyeq_\alpha g$ and $g \preccurlyeq_\alpha f$. Finally, $f \approx g$ (respectively, $f \preccurlyeq g$) if $f \approx_\alpha g$ (respectively, $f \preccurlyeq_\alpha g$) for some α. The idea is that the *boundedness relation* $\approx$ does not pay attention to exact values, but does preserve the existence of bounds. Remark that $f \not\preccurlyeq g$ if and only if there exists a set $D \subseteq E$ such that g is bounded on D but f is unbounded on D.

A *cost function over* E is an equivalence class of $\mathcal{F}_E/\approx$. In practice, a cost function (denoted $f, g, \ldots$) will be represented by one of its elements in $\mathcal{F}_E$. In this paper, E will usually be $\mathbb{A}^\omega$. The functions defined by cost automata or cost logics will always be considered as cost functions, i.e. only considered up to $\approx$.

Lemma 3. *This definition of $\approx$ is equivalent to the one in the body*

Proof. We will note $\approx_\alpha$ for this definition with correction function α, and $\approx$ the equivalence relation as defined in Section 1.2 of the body. Let f and g be two cost functions on a set E. Assume there exists α such that $f \approx_\alpha g$. If f is bounded by M on some set $X \subseteq E$, then g is bounded by $\alpha(M)$ on X. By symmetry, for all $X \subseteq E$ if $g(X)$ is bounded then $f(X)$ is bounded. We can conclude that $f \approx g$.

Conversely, assume $f \approx g$, i.e. f and g are bounded on the same sets. We want to first build α such that $f \preccurlyeq_\alpha g$. For all $n \in \mathbb{N}$, we set

$$\alpha(n) := \sup \{f(x) : x \in E, g(x) \leq n\},$$

and $\alpha(\infty) = \infty$ as usual. We have to show that for all $n \in \mathbb{N}$, $\alpha(n) < \infty$. Let $n \in \mathbb{N}$ and $X := \{x \in E : g(x) \leq n\}$. Since $g(X)$ is bounded by n and $f \approx g$, $f(X)$ is bounded. But $\alpha(n)$ is defined as $\sup f(X)$, so $\alpha(n) < \infty$.

Let $x \in E$ and $n = g(x)$, by definition of α we have $f(x) \leq \alpha(n) = \alpha(g(x))$. This is true for all x so $f \preccurlyeq_\alpha g$. By symmetry, we can get a correction function β such that $g \preccurlyeq_\beta f$, and finally $f \approx_{\max(\alpha,\beta)}$.

Notice that the correction functions built here are increasing, in practice this will always be the case.

B On the First-Order Fragment

Lemma 4. *The two definitions of CMSO are equivalent in terms of expressive power.*

Proof. Let CMSO' be MSO with an extra predicate $|X| \leq N$, required to appear positively. We can inductively translate any CMSO-formula into a CMSO'-formula by replacing all occurences of "$\forall^{\leq N} x.\varphi(x)$" by "$\exists X.(|X| \leq N \wedge \forall x.(x \in X \vee \varphi(x)))$". Conversely, we can go from CMSO' to CMSO by replacing all occurences of "$|X| \leq N$" by "$\forall^{\leq N} x.x \notin X$". Both translations preserve exact values of the semantics (i.e. semantics are equal and not just equivalent up to $\approx$).

Proof of Theorem 1

B.1 CLTL to CFO

Let φ be a CLTL-formula and $\psi(x)$ be a CFO-formula with one free variable. Let $\alpha : \mathbb{N}_\infty \to \mathbb{N}_\infty$ be a correction function (so $\alpha(\infty) = \infty$). We say that φ and $\psi(x)$ are α-equivalent at x if for all word u and integers n, i,

- If $(u, n, i) \models \psi(x)$ then $(u, \alpha(n), i) \models \varphi$,
- If $(u, n, i) \models \varphi$ then $(u, \alpha(n), i) \models \psi(x)$,

Where n is the valuation for N, and i instantiates either x (for $\psi(x)$) or the position where the CLTL-formula is evaluated (for φ).

For every CLTL-formula φ and variable x, we want to build an equivalent CFO-formula $\lambda_x(\varphi)$ with only x as free variable. We define the function λ_x by induction on the CLTL-formula:

- $\lambda_x(a) = a(x)$,
- $\lambda_x(\varphi \wedge \psi) = \lambda_x(\varphi) \wedge \lambda_x(\psi), \quad \lambda_x(\varphi \vee \psi) = \lambda_x(\varphi) \vee \lambda_x(\psi)$,
- $\lambda_x(\varphi \mathbf{R} \psi) = \forall z > x.(\lambda_z(\psi) \vee (\exists y.x < y < z \wedge \lambda_y(\varphi))$,
- $\lambda_x(\varphi \mathbf{U} \psi) = \exists z > x.(\lambda_z(\psi) \wedge \forall y.x < y < z \Rightarrow \lambda_y(\varphi))$,
- $\lambda_x(\varphi \mathbf{U}^{\leq N} \psi) = \exists z > x.(\lambda_z(\psi) \wedge \forall^{\leq N} y.x < y < z \Rightarrow \lambda_y(\varphi))$.

It is straightforward to verify that for all φ, φ and $\lambda_x(\varphi)$ are equivalent at x, with $\alpha = id$. Finally if φ is a CLTL-formula, we build the equivalent CFO-formula "$\exists x.\lambda_x(\varphi) \wedge x = 0$", where "$x = 0$" is an abreviation for "$\forall y.x \leq y$". This expresses that φ is true at position 0.

B.2 Proof of the Separation Theorem

We need to eliminate the nesting of future operators under the scope of past operators. The converse is symmetric.

We use notations $\overline{\mathbf{S}}, \overline{\mathbf{S}^{\leq N}}, \overline{\mathbf{U}}, \overline{\mathbf{U}^{\leq N}}, \overline{\mathbf{R}}$ for the "large" versions of the corresponding operators (i.e. shifted by one position: the current position is also taken into account). For instance $A \overline{\mathbf{U}} B \equiv B \vee (A \wedge \mathbf{X}(A \mathbf{U} B))$.

Lemma 5 ([8]).

- $c\mathbf{U}(a \vee b) \Leftrightarrow (c\mathbf{U}a) \vee (c\mathbf{U}b)$;

– $c\mathbf{S}(a \vee b) \Leftrightarrow (c\mathbf{S}a) \vee (c\mathbf{S}b)$;
– $(a \wedge b)\mathbf{U}c \Leftrightarrow (a\mathbf{U}c) \wedge (b\mathbf{U}c)$;
– $(a \wedge b)\mathbf{S}c \Leftrightarrow (a\mathbf{S}c) \wedge (b\mathbf{S}c)$;

Moreover, this lemma remains true with the quantitative variants $\mathbf{U}^{\leq N}$ and $\mathbf{S}^{\leq N}$, with a correction function identity for the first two (involving disjunctions), and $\alpha(n) = 2n$ for the last two (involving conjunctions). There is also the analog lemma for the Release variants $\mathbf{R}$ and $\mathbf{Q}$.

The classical cases, already treated in [8] are

1. $q\mathbf{S}(a \wedge (A\mathbf{U}B))$
2. $q\mathbf{S}(a \wedge (B\mathbf{R}A))$
3. $(q \vee (A\mathbf{U}B))\mathbf{S}a$
4. $(q \vee (B\mathbf{R}A))\mathbf{S}a$
5. $(q \vee (A\mathbf{U}B))\mathbf{S}(a \wedge (A\mathbf{U}B))$
6. $(q \vee (B\mathbf{R}A))\mathbf{S}(a \wedge (B\mathbf{R}A))$

Notice that despite the use of negations to treat these cases in [8], positivity of every subformula is kept. So these translations are still valid in our context.

In [8], two more cases are treated: $(q \vee (A\mathbf{U}B))\mathbf{S}(a \wedge (B\mathbf{R}A))$ and $(q \vee (B\mathbf{R}A))\mathbf{S}(a \wedge (A\mathbf{U}B))$. In fact we will not need to treat these cases here, because we will consider that $A\mathbf{U}B$ is not a subformula of $B\mathbf{R}A$, whereas in [8], $B\mathbf{R}A$ is replaced by $\neg(A\mathbf{U}B)$ (that we cannot do here).

Keeping track of the positivity allows us to treat $A\mathbf{U}B$ and $B\mathbf{R}A$ as two different subformulas, and hence reduce the number of ways an $\mathbf{U}$ formula can appears under a Since.

We now need to take care of the quantitative variants, i.e.

	i	ii	iii
1	$q\mathbf{S}(a \wedge (A\mathbf{U}^{\leq N}B))$	$q\mathbf{S}^{\leq N}(a \wedge (A\mathbf{U}B))$	$q\mathbf{S}^{\leq N}(a \wedge (A\mathbf{U}^{\leq N}B))$
2	$q\mathbf{S}^{\leq N}(a \wedge (B\mathbf{R}A))$		
3	$(q \vee (A\mathbf{U}^{\leq N}B))\mathbf{S}a$	$(q \vee (A\mathbf{U}B))\mathbf{S}^{\leq N}a$	$(q \vee (A\mathbf{U}^{\leq N}B))\mathbf{S}^{\leq N}a$
4	$(q \vee (B\mathbf{R}A))\mathbf{S}^{\leq N}a$		
5	$(q \vee (A\mathbf{U}^{\leq N}B))\mathbf{S}(a \wedge (A\mathbf{U}^{\leq N}B))$	$(q \vee (A\mathbf{U}B))\mathbf{S}^{\leq N}(a \wedge (A\mathbf{U}B)$	$(q \vee (A\mathbf{U}^{\leq N}B))\mathbf{S}^{\leq N}(a \wedge (A\mathbf{U}B))$
6	$(q \vee (B\mathbf{R}A))\mathbf{S}^{\leq N}(a \wedge (B\mathbf{R}A))$		

Cases 1.i to 2.i are quite straightforward: the behaviour is the same as the corresponding classical cases, except that some intervals are cut in two pieces, so the number of mistakes can double.

1.i: $q\mathbf{S}(a \wedge (A\mathbf{U}^{\leq N}B))$

Let t be the present position and y be the position reached by the until.

Formula 1.i is equivalent to

$$\begin{array}{rll} & (q\mathbf{S}a) \wedge (A\mathbf{S}^{\leq N}a) \wedge (A\mathbf{U}^{\leq N}B) & : \ t < y \\ \bigvee & (q\mathbf{S}a) \wedge (A\mathbf{S}^{\leq N}a) \wedge B & : \ t = y \\ \bigvee & q\mathbf{S}(B \wedge q \wedge (A\mathbf{S}^{\leq N}a) \wedge (q\mathbf{S}a)) & : \ y < t \end{array}$$

Notice that we can contract the present case B and the future case $A\mathbf{U}^{\leq N}B$ into a single case $A\overline{\mathbf{U}^{\leq N}}B \equiv B \vee (A\mathbf{U}^{\leq N}B)$. We will do this in the next cases to simplify the reading, but notice that it is always easy to break the formula into a boolean combination of pure past, pure present, and pure future formula.

Correction $\alpha(n) = 2n + 1$. The $+1$ is due to the first case, where we do not ask A to be true in position t.

1.ii: $q\mathbf{S}^{\leq N}(a \wedge (A\mathbf{U}B))$

Equivalent to

$$\begin{array}{c}(q\mathbf{S}^{\leq N}a) \wedge (A\mathbf{S}a) \wedge (A\overline{\mathbf{U}}B)\\ \bigvee q\mathbf{S}^{\leq N}(B \wedge (A\mathbf{S}a) \wedge (q\mathbf{S}^{\leq N}a))\end{array}$$

Correction $\alpha(n) = 2n + 1$.

1.iii: $q\mathbf{S}^{\leq N}(a \wedge (A\mathbf{U}^{\leq N}B))$

Equivalent to

$$\begin{array}{c}(q\mathbf{S}^{\leq N}a) \wedge (A\mathbf{S}^{\leq N}a) \wedge (A\overline{\mathbf{U}^{\leq N}}B)\\ \bigvee q\mathbf{S}^{\leq N}(B \wedge (A\mathbf{S}^{\leq N}a) \wedge (q\mathbf{S}^{\leq N}a))\end{array}$$

Correction $\alpha(n) = 2n + 1$.

2.i: $q\mathbf{S}^{\leq N}(a \wedge (B\mathbf{R}A))$

Equivalent to

$$\begin{array}{c}(q\mathbf{S}^{\leq N}a) \wedge (A\mathbf{S}a) \wedge (B\overline{\mathbf{R}}A)\\ \bigvee q\mathbf{S}^{\leq N}(B \wedge (A\mathbf{S}a) \wedge (q\mathbf{S}^{\leq N}a))\end{array}$$

Correction $\alpha(n) = 2n + 1$.

3.i: $(q \vee (A\mathbf{U}^{\leq N}B))\mathbf{S}a$

This formula expresses that there is some position y in the past where a is true, and since y, the word is a concatenation of blocks of the form: $qqqqqqqAAAAAxAAxAAB$ until the current position. The number of mistakes x is bounded by N in each A segment.

Let $\varphi = [A\mathbf{S}^{\leq N}(q\overline{\mathbf{S}}(B \vee a)]\mathbf{S}a$. Then φ is a pure past formula which verifies this pattern.

In addition to φ, two cases can occur in the present position: either $A\mathbf{U}^{\leq N}B$ is satisfied at $t-1$, or the last block contains only q's. So the formula 3.i is equivalent to $\varphi \wedge [(A\overline{\mathbf{U}^{\leq N}}B)) \vee (q\overline{\mathbf{S}}(B \vee a))]$.

The last block may be cut in two, so the correction is $\alpha(n) = 2n$.

3.ii: $(q \vee (A\mathbf{U}B))\mathbf{S}^{\leq N}a$

We now count the mistakes globally, and distinguish two kinds of blocks: "good block" are of the form $BqqqqqqqqqqAAAAAAAAAAB$, and "wrong blocks" can make mistakes on the q segments (the end of the q segment is defined by the

first place where $A\mathbf{U}B$ is true). We sum the number of missing q on the whole word.

We can approximate the number of total mistakes by bounding the number of mistakes in a wrong block, together with the total number of wrong blocks.

A "good block" must start with B or a, contain a segment of q, then a segment of A, until the next B (or the present).

The number of mistakes in a wrong block is the number of q missing before the first position where $A\mathbf{U}B$ is true.

Let $\varphi_1 = [A\mathbf{S}(q\overline{\mathbf{S}^{\leq N}}(B \vee a))]\mathbf{S}a$ be the formula counting the maximum number of mistakes in a wrong block.

We also need to bound the total number of wrong blocks. To do so, we would like to use a formula such as $[B \Rightarrow A\mathbf{S}(q\overline{\mathbf{S}}(B \vee a))]\mathbf{S}^{\leq N}a$, but it is not allowed since B occurs in the left side of an implication, so it is in fact negated, which is not allowed if B contains quantitative operators.

Instead, we will verify that the block is good locally. The total number of local mistakes will be between the number of wrong blocks and twice the number of total mistakes in the original formula.

Let $\varphi_2 = ([(q \vee B) \wedge \mathbf{Y}(q \vee B)] \vee [(A \vee B) \wedge (\mathbf{Y}(A \vee q \vee B))])\,\mathbf{S}^{\leq N}a$. This describes the local condition that good blocks must verify, and φ_2 counts the number of positions where this condition fails. Each mistake in q segments is reponsible for at most two failures of this local condition, and each wrong block fails it at least once, so the desired inequality is verified. Notice that to shorten the formula, we did not mention a in the local conditions. This can just add one error, immediatly after the a we start from.

Formula 3.ii is equivalent to $\varphi' = \varphi_1 \wedge \varphi_2 \wedge [(A\overline{\mathbf{U}}B) \vee (q\mathbf{S}^{\leq N}(B \vee a))]$. Notice that we allow here the current block to be a wrong one, by only asking $(q\mathbf{S}^{\leq N}(B \vee a))$ at the current position, in the case where $A\overline{\mathbf{U}}B$ where is not satisfied. This induces an extra n in the correction function.

Correction $\alpha(n) = 2n^2 + n + 1$.

Example 3. We illustrate this case with an example. To keep the notations, we will use alphabet $\mathbb{A} = \{a, q, A, B, e\}$, where e is a new letter we will use for mistakes. Let $u = BeaAaqqAAABeeeqqAqAABAAeAA|A|ABAq$. The vertical lines in u delimit the position t where we want to evaluate the formula. For instance $[\![A]\!](u) = 0$ because there is indeed an A at position t.

We now evaluate the formula 3.ii: what is $[\![q \vee (A\mathbf{U}B))\mathbf{S}^{\leq N}a]\!](u)$? The last a before t occurs at position 4 in the word (the first letter is at position 0). The total number of mistakes is then counted, we mark them here in boldface:

$$BeaAaqqAAAB\mathbf{eee}qq\mathbf{A}qAAB\mathbf{AAe}AA|A|ABAq,$$

hence $[\![q \vee (A\mathbf{U}B))\mathbf{S}^{\leq N}a]\!](u) = 7$.

We now evaluate the translated formula. First $[\![\varphi_1]\!](u) = 4$: it is the maximal number of mistakes in the same block. We now mark the places where the local condition of φ_2 fails: $BeaAa\mathbf{q}qAAAB\mathbf{eeeq}qA\mathbf{q}AABAA\mathbf{eA}A|A|ABAq$. This gives us $[\![\varphi_2]\!](u) = 8$. Finally the last formula is verified with $N = 0$, since $A\overline{\mathbf{U}}B$ is true at position t.

The conjunction resolves in a maximum, so we end up with $[\![\varphi']\!](u) = 8$.

3.iii: $(q \vee (A\mathbf{U}^{\leq N}B))\mathbf{S}^{\leq N}a$

A good block is now allowed to have N mistakes in segments of A's. We want to count the sum of all mistakes in q segments, but the mistakes in A segments are only bounded on each segment and not globally.

So we now define $\varphi_1 = [A\mathbf{S}^{\leq N}(q\overline{\mathbf{S}^{\leq N}}(B \vee a))]\mathbf{S}a$, bounding locally the number of mistakes of each wrong block.

The difficulty here is to express a local condition on good blocks, because they are still allowed mistakes in the A segment. The trick is that we can always consider that the segments of q end with a q (or a B if it is empty), because up to a factor 2 on the mistakes in A segments, it is better to count mistakes as "local" ones than as "global" ones.

The local condition will then express "we are in the A segment (including mistakes), or we are at q and the preceding position is q"

Let $\varphi_2 = [(A\overline{\mathbf{S}^{\leq N}}(q \vee B)) \vee ((q \vee B) \wedge \mathbf{Y}(q \vee B))]\mathbf{S}^{\leq N}a$.

As before, the mistakes of the main $\mathbf{S}^{\leq N}$ operator of φ_2 will be between the number of wrong blocks and twice the total number of mistakes.

Formula 3.iii is equivalent to $\varphi_1 \wedge \varphi_2 \wedge [(A\overline{\mathbf{U}^{\leq N}}B)) \vee (q\mathbf{S}^{\leq N}(B \vee a))]$: as before the last block can be considered to be wrong.

Correction $\alpha(n) = (2n)^2 + n + 1$. We still have a $+1$ from the first a which causes a local mistake immediatly after.

4.i: $(q \vee (B\mathbf{R}A))\mathbf{S}^{\leq N}a$

Same as 3.ii except that the last B is allowed to never occur (and in this case A is always true in the future). φ_1 and φ_2 are identical, and formula 4.i is equivalent to $\varphi_1 \wedge \varphi_2 \wedge [(B\overline{\mathbf{R}}A)) \vee (q\mathbf{S}^{\leq N}(B \vee a))]$

5.i: $(q \vee (A\mathbf{U}^{\leq N}B))\mathbf{S}(a \wedge (A\mathbf{U}^{\leq N}B))$

The first B can be before or after (in the large sense) the current position. If it is before, then the situation looks like 3.i: let $a' = B \wedge (A\mathbf{S}^{\leq N}a)$, then we must verify $\psi = (q \vee (A\mathbf{U}^{\leq N}B))\mathbf{S}a'$, which is exactly 3.i with a' instead of a. But a' here is a past formula, so the solution of 3.i is also a solution for ψ.

If it is after (or equal), then the situation is very simple, we just need to verify $\psi' = (A\mathbf{S}^{\leq N}a) \wedge (A\overline{\mathbf{U}^{\leq N}}B)$, which is already separated. Finally, 5.i is equivalent to $\psi \vee \psi'$.

5.ii: $(q \vee (A\mathbf{U}B))\mathbf{S}^{\leq N}(a \wedge (A\mathbf{U}B))$

If we call $a' = B \wedge (A\mathbf{S}a)$, we must verify either $\psi = (q \vee (A\mathbf{U}B))\mathbf{S}^{\leq N}a'$, which corresponds to case 3.ii, or $\psi' = (A\mathbf{S}a) \wedge (A\overline{\mathbf{U}}B)$. Finally 5.ii is equivalent to $\psi \vee \psi'$.

In the same way, case 5.iii is reduced to case 3.iii with $a' = B \wedge (A\mathbf{S}^{\leq N}a)$, and $\psi' = (A\mathbf{S}^{\leq N}a) \wedge (A\overline{\mathbf{U}^{\leq N}}B)$.

In each case, the correction function we get can be taken to be $\alpha'(n) = \alpha(n) + n$, where $\alpha(n)$ comes from the case 3.x we reuse, and the $+n$ is due to the errors commited by a'.

Case 6.i: $(q \vee (B\mathbf{R}A))\mathbf{S}^{\leq N}(a \wedge (B\mathbf{R}A))$
As it was done in case 4.i, it suffices to take case 5.ii and change the Until to Release in the future formulas. We need to apply case 4.i with $a' = B \wedge (A\mathbf{S}^{\leq N}a)$, and do the conjunction with $\psi' = (A\mathbf{S}^{\leq N}a) \wedge (A\overline{\mathbf{U}^{\leq N}}B)$.

We will explicit the formula we get in this case: as in Case 3.ii, let

$$\varphi_1 = [A\mathbf{S}(q\overline{\mathbf{S}^{\leq N}}(B \vee a'))]\mathbf{S}a',$$
$$\varphi_2 = ([(q \vee B) \wedge \mathbf{Y}(q \vee B)] \vee [(A \vee B) \wedge (\mathbf{Y}(A \vee q \vee B))])\,\mathbf{S}^{\leq N}a'.$$

Formula 6.i is equivalent to $\varphi_1 \wedge \varphi_2 \wedge [(B\overline{\mathbf{R}}A)) \vee (q\mathbf{S}^{\leq N}(B \vee a'))]$.

Cases with Q instead of S Finally, replacing $\mathbf{S}$ with $\mathbf{Q}$ in all the precedent cases is not problematic, and we can separate the formulas in the same way: the only difference is that we do not require the presence of a.

Notice that the quantitative variant of $B\mathbf{Q}A$ can be described as $A\overline{\mathbf{S}^{\leq N}}(B\mathbf{Q}A)$, so we do not need a new operator in the syntax.

Termination of the denesting procedure

We adapt the proof of [8], which almost works as it is. The main idea is to consider that $\mathbf{U}^{\leq N}$ and $\mathbf{R}$ behave similarly as $\mathbf{U}$ (and same for the past connectives), so the principle of the classical proof can be applied to each of these operators separately.

We reproduce it here for completeness.

Lemma 6. *Let $\mathbf{U}'$ be an operator in $\{\mathbf{R}, \mathbf{U}, \mathbf{U}^{\leq N}\}$, and $\mathbf{S}'$ be an operator in $\{\mathbf{Q}, \mathbf{S}, \mathbf{S}^{\leq N}\}$. If A and B are CLTL$_P$-formulas without temporal operators, and C and F are such that any appearance of future operator is as $A\mathbf{U}'B$, and is not nested under any past operator.*

Then $C\mathbf{S}'F$ is equivalent to a separated formula.

Proof. If $A\mathbf{U}'B$ does not appear then we are done. Otherwise, repeated use of Lemma 5 allows us to rewrite $C\mathbf{S}F$ as a boolean combination of past formulas $C_1\mathbf{S}'C_2$, and formulas of the form either:

- $C_1\mathbf{S}'(C_2 \wedge (A\mathbf{U}'B))$
- $(C_1 \vee (A\mathbf{U}'B))\mathbf{S}'C_2$
- $(C_1 \vee (A\mathbf{U}'B))\mathbf{S}'(C_2 \vee (A\mathbf{U}'B))$

where C_1 and C_2 are past formulas. Then the preceding transformations (from 1.i to 6.i) allow us to get from this an equivalent boolean combination of formulas of the three following forms:

– $a\mathbf{Q}b$, $a\mathbf{S}b$, $a\mathbf{S}^{\leq N}b$, where a and b are past formulas,
– C_1, C_2, A and B,
– $A\mathbf{U}'B$.

Thus we have a separated equivalent.

We then begin the inductive process of removing future operators $\mathbf{U}'$ from under the scope of past operators $\mathbf{S}'$.

Lemma 7. *If A and B are without temporal operators, and the only appearance of future operators in an CLTL$_P$-formula D is as $A\mathbf{U}'B$, for $\mathbf{U}' \in \{\mathbf{R}, \mathbf{U}, \mathbf{U}^{\leq N}\}$.*

Then D is equivalent to a separated formula, where the only appearance of future operators is as $A\mathbf{U}'B$, for $\mathbf{U}' \in \{\mathbf{R}, \mathbf{U}, \mathbf{U}^{\leq N}\}$.

Proof. By induction on the maximum number k of nested $\mathbf{S}'$ above any $A\mathbf{U}'B$.

If $k = 0$ then D is already separated. If $k > 0$, we apply Lemma 6 to each of the most deeply nested $C\mathbf{S}'F$ in which $A\mathbf{U}'B$ appears. We get an equivalent formula where the maximum depth of $A\mathbf{U}'B$ under an $\mathbf{S}'$ is reduced. Moreover future operators still only appear as $A\mathbf{U}'B$.

Lemma 8. *For each $i \in [1, n]$, let A_i and B_i be CLTL-formulas without temporal operators, and $\mathbf{U}_i \in \{\mathbf{R}, \mathbf{U}, \mathbf{U}^{\leq N}\}$. Suppose that the only appearances of future operators in D are in the form $A_i\mathbf{U}_iB_i$. Then D is equivalent to a separated formula.*

Proof. By induction on n.

If $n = 1$, Lemma 7 allows us to conclude. If $n > 1$ we want to separate only for $A_n\mathbf{U}_nB_n$. To do so, we replace in D every $A_i\mathbf{U}_iB_i$ by a new atom q_i, to obtain a formula D'. Using lemma 7, we can obtain a separated E' equivalent to D', using atoms $(q_i)_{1\leq i\leq n-1}$ and with $A_n\mathbf{U}_nB_n$ being the only form of occurrences of future operators.

E' is a boolean combination of pure present formulas, pure future formulas $A_n\mathbf{U}_nB_n$, and pure past formulas D_j with atoms $(q_i)_{1\leq i\leq n-1}$ in addition to the normal ones. We can now substitute in the D_j's each q_i by $A_i\mathbf{U}_iB_i$. By induction, each D_j is equivalent to a separated formula, and get the wanted separated formula.

The next step is to allow nesting of $\mathbf{U}'$ beneath $\mathbf{S}'$, but not $\mathbf{S}'$ within a $\mathbf{U}'$.

Lemma 9. *Let D be an CLTL$_P$-formula with no $\mathbf{S}'$ between a $\mathbf{U}'$. Then D is equivalent to a separated formula.*

Proof. By induction on the maximum depth n of nesting of $\mathbf{U}'$ beneath an $\mathbf{S}'$.

Lemma 8 treats the case $n = 1$.

Assume $n > 1$, and let $A_i\mathbf{U}_iB_i$ for $i \in [1, k]$ be subformulae of D that gather every appearances of appearances of future operator (with $\mathbf{U}_i \in \{\mathbf{R}, \mathbf{U}, \mathbf{U}^{\leq N}\}$). Each A_i and B_i is a boolean combination of atoms and formulas $X_{ij}\mathbf{U}_{ij}Y_{ij}$. We replace each of this formula by a new atom x_{ij}, to obtain A'_i and B'_i.

We now replace in D each occurence of $A_i \mathbf{U}_i B_i$ which is not contained within another $A\mathbf{U}'B$ by $A'_i \mathbf{U}_i B'_i$, to obtain D'. By Lemma 8, D' is equivalent to a separated formula E' with new atoms x_{ij}. Substituting back x_{ij} by $A_i \mathbf{U}_i B_i$ in E' gives us a formally with maximum nesting depth of $\mathbf{U}'$ equal to $n-1$, so we can separate it by induction.

We are now ready to show the separation theorem, i.e. any CLTL$_{\text{P}}$-formula D with past and future operators is equivalent to a separated formula.

For this we will reuse notation of [8]: If B is a subformula of A, we call the *junction depth* of B in A the maximal number of alternations of nested past and future operators to reach B in A.

For instance the junction depth of a in formula 1.i is 1 but the junction depth of B is 2.

The junction depth of a formula is the maximal junction depth of its subformulas.

We now proceed by induction on the junction depth of D. If it zero or one then D is already separated. We assume that it is at least two.

D is a boolean combination of atoms and formulas of the form $D_1 \mathbf{S}' D_2$, $D_1 \mathbf{U}' D_2$. We just want to separate the latter two forms. By symmetry, it suffices to show the result for formulas of the form $D = D_1 \mathbf{S}' D_2$.

Let $A_i \mathbf{U}_i B_i$ for $i \in [1, k]$ be the subformulas covering appearances of maximal future operators in D (i.e. every future operator of D occurs in one of these formulas).

We now replace in D every subformula $E_{ij} \mathbf{S}_{ij} F_{ij}$ occuring in some $A_i \mathbf{U}_i B_i$ by a new atom x_{ij}. This gets us a formula satisfying hypothesis of Lemma 9, so we can separate it and get an equivalent formula E'. By replacing each x_{ij} with $E_{ij} \mathbf{S}_{ij} F_{ij}$ in E', we get a new formula with strictly smaller junction depth, and we can conclude by induction hypothesis.

Notice that each denesting step can square the correction function, so the final correction function will be of the form $\alpha(n) = O(n^{2^{nd(D)}})$, where $nd(D)$ is the nesting depth of D.

□

B.3 Proof of Proposition 1

Now that we have the Separation Theorem, we want to use it to prove that every CFO-formula can be effectively translated into an equivalent CLTL-formula.

Proof. As before, we will in fact translate a CFO-formula $\varphi(x)$ with one free variable x and an arbitrary set of unary predicates $P_1, \ldots, P_n$ (which can be letter predicates for instance), into a CLTL-formula that evaluates the word from position x, using the same unary predicates. To translate a closed formula φ, it then suffices to translate the formula $\varphi \wedge x = 0$.

The proof is by induction on $\mathrm{qr}(\varphi(x))$. If $\mathrm{qr}(\varphi(x)) = 0$, then $\varphi(x)$ is a boolean combination of atoms of the form $x \leq x$ and $a(x)$, so it is clearly equivalent to a CLTL-formula.

We assume the result for rank at most k. It suffices to show the result for formulas of the form $\exists y.\varphi(x,y)$, $\forall y.\varphi(x,y)$, $\forall^{\leq N} y.\varphi(x,y)$ to be able to conclude by induction, since a formula of rank $k+1$ is a boolean combination of such formulas, and formulas of lower rank.

The principle is to remove the variable x, from $\varphi(x,y)$, and use the induction hypothesis on the resulting formula.

For this, we first start by defining for each $S \subseteq [1,n]$ a formula $\varphi^S(x,y)$, which replaces in $\varphi(x,y)$ each $P_i(x)$ by $\top$ if $i \in S$ and by $\bot$ if $i \notin S$.

Then $\varphi(x,y)$ is equivalent to

$$\bigwedge_{S \subseteq [1,n]} ((\bigwedge_{i \in S} P_i(x) \wedge \bigwedge_{i \notin S} \neg P_i(x)) \Rightarrow \varphi^S(x,y)).$$

Since the $P_i(x)$ and their negations are available in CLTL-formulas, and quantifications on y commute with conjunctions and implication in this formula, it suffices to show the result for formulas $\varphi^S(x,y)$, which does not contain predicates $P_i(x)$. Let $\varphi'(x,y)$ be such a formula.

We then replace predicates $z < x$ by $\mathrm{Pos}_<(z)$, $z = x$ by $\mathrm{Pos}_=(z)$ and $z > x$ by $\mathrm{Pos}_>(z)$ in $\varphi'(x,y)$.

This gets us a formula using new predicates: $\varphi''(y, \mathrm{Pos}_<, \mathrm{Pos}_=, \mathrm{Pos}_>)$, where x is no longer used.

We will restrict ourselves to structures that interpret these predicates as we defined them, i.e. $\mathrm{Pos}_<(z)$ is true iff $z < x$, and so on.

By induction hypothesis, we obtain a CLTL-formula ψ equivalent to φ'' at y (on these structures), and using predicates $\mathrm{Pos}_<, \mathrm{Pos}_=, \mathrm{Pos}_>$.

We can now remark that

- $\exists y.\varphi'(x,y)$ is equivalent to $\psi' = \mathbf{P}\psi \vee \psi \vee \mathbf{F}\psi$;
- $\forall y.\varphi'(x,y)$ is equivalent to $\psi' = \mathbf{H}\psi \wedge \psi \wedge \mathbf{G}\psi$;
- $\forall^{\leq N} y.\varphi'(x,y)$ is equivalent to $\psi' = \mathbf{H}^{\leq N}\psi \wedge \mathbf{G}^{\leq N}\psi$, with a correction function $\alpha(m) = 2m+1$.

In all cases ψ' is evaluated at position x, and still uses additional predicates $\mathrm{Pos}_<, \mathrm{Pos}_=, \mathrm{Pos}_>$ that compare the current position with position x.

But according to the Separation Theorem, ψ' can be translated into a boolean combination of pure formulas. We can replace $(\mathrm{Pos}_<, \mathrm{Pos}_=, \mathrm{Pos}_>)$ by $(\top, \bot, \bot)$ (resp. $(\bot, \top, \bot), (\bot, \bot, \top)$) in pure past formulas (resp. pure present, pure future), so indeed we get an CLTL-formula ψ'', using only predicates $P_1, \ldots, P_n$, and equivalent to the original CFO-formula at x. This closes the induction and the proof of the theorem.

Since we are ultimately interested in a formula which is evaluated at the first position (where pure past formulas can be trivially evaluated to true or false for all words), we end up with an equivalent CLTL-formula without past operators.

B.4 CLTL to *B*-VWAA

Given an CLTL sentence θ, we can design a very-weak B-automata $\mathcal{A}_\theta$ which tries to prove that $[\![\theta]\!](u)$ is bounded (it is a simple adaptation of [7, Proposition 13.1]). It is defined as follows:

– $Q := \{\varphi : \varphi \text{ is a subformula of } \theta\}$, $q_0 := \theta$, $\Gamma := \{\gamma\}$
– $\delta : Q \times \mathbb{A} \to \mathcal{B}^+(\mathbb{C} \times Q)$ is defined by

$$\delta(a,b) := \begin{cases} \mathsf{true} & \text{if } b = a \\ \mathsf{false} & \text{otherwise} \end{cases} \qquad \begin{aligned} \delta(\varphi \vee \psi, b) &:= \delta(\varphi, b) \vee \delta(\psi, b) \\ \delta(\varphi \wedge \psi, b) &:= \delta(\varphi, b) \wedge \delta(\psi, b) \end{aligned}$$

$$\delta(\varphi \mathbf{R} \psi, b) := (\varepsilon, \psi) \wedge ((\varepsilon, \varphi) \vee (\varepsilon, \varphi \mathbf{R} \psi))$$

$$\delta(\varphi \mathbf{U} \psi, b) := (\varepsilon, \psi) \vee ((\varepsilon, \varphi) \wedge (\varepsilon, \varphi \mathbf{U} \psi))$$

$$\delta(\varphi \mathbf{U}^{\leq N} \psi, b) := (\mathtt{r}, \psi) \vee (\mathtt{ic}, \varphi \mathbf{U}^{\leq N} \psi) \vee ((\mathtt{r}, \varphi) \wedge (\varepsilon, \varphi \mathbf{U}^{\leq N} \psi))$$

– $F := \{\varphi : \varphi \text{ is a subformula of } \theta \text{ of the form } \psi_1 \mathbf{R} \psi_2\}$

Note that only one counter is required, since the automaton is only ever checking a single $\mathbf{U}^{\leq N}$ formula at a time. When a copy of the automaton moves to another $\mathbf{U}^{\leq N}$ subformula, it can reset the counter and reuse it in that context.

We now show the correctness of this construction.

The proof is by induction on the structure of θ. The base case, when $\theta = a$ is trivial.

Now consider when θ is of the form $\varphi \mathbf{U}^{\leq N} \psi$. We prove that $[\![\theta]\!] = [\![\mathcal{A}_\theta]\!]$. Fix some $u \in \mathbb{A}^\omega$ and let $u^i := u(i)u(i+1)\ldots$ be the suffix of u starting at position i.

Assume $[\![\theta]\!](u) \leq N$. Then there is some $i > 0$ and some set $I \subseteq [1, i-1]$ such that $[\![\psi]\!](u^i) \leq N$, $|I| \leq N$, and for all $1 \leq j < i$ with $j \notin I$, $[\![\varphi]\!](u^j) \leq N$. Now consider the strategy for $\mathcal{A}_\theta$ on u which selects the first disjunct when reading position $i-1$, the second disjunct when reading position $j-1$ for $j \in I$, and the third disjunct otherwise. Because $|I| \leq N$, any play in the strategy selects the second disjunct which increments the counter at most N times. Moreover, by the inductive hypothesis, any partial play starting from ψ or φ will have value at most N (and the counter is reset when moving to these states). Hence, $[\![\mathcal{A}_\theta]\!](u) \leq N$.

Now assume that $[\![\mathcal{A}_\theta]\!](u) \leq N$ so there is a strategy σ for Eve with $val(\sigma) \leq N$. There must be a play π in σ which visits $(\mathtt{r}, \psi)$ (otherwise, there is a play which stabilizes in a next-until formula, contradicting the fact that $val(\sigma) \leq N$). Consider the least i such that there is a $\pi \in \sigma$ which selects $(\mathtt{r}, \psi)$ for position i.

Omitting the moves up position i from all plays in σ results in a strategy σ_i for $\mathcal{A}_\psi$ which witnesses $[\![\mathcal{A}_\psi]\!](u^i) \leq N$. By the inductive hypothesis $[\![\psi]\!](u^i) \leq N$.

The remaining positions $j < i$ on π must correspond to $(\mathtt{ic}, \varphi \mathbf{U}^{\leq N} \psi)$ or $(\varepsilon, \varphi \mathbf{U}^{\leq N} \psi)$ (if $(\mathtt{r}, \varphi)$ were selected at such a position j then it is not possible to reach $(\mathtt{r}, \psi)$ since ψ is not a subformula of φ). Let $I \subseteq [1, i-1]$ be the set of positions for which $(\mathtt{ic}, \varphi \mathbf{U}^{\leq N} \psi)$ is chosen by π. Since $val(\pi) \leq N$ and there is no reset action for the counter in positions $j < i$, $|I| \leq N$. For all positions $j \notin I$, there must a play $\pi_j \in \sigma$ that selects $(\mathtt{r}, \varphi)$ (since δ forces the conjunction of $(\mathtt{r}, \varphi)$ and $(\varepsilon, \varphi \mathbf{U}^{\leq N} \psi)$). Let σ_j be the set of plays in σ which select $(\mathtt{r}, \varphi)$ for position j, with this prefix up to position j removed. Then σ_j is a strategy witnessing that $[\![\mathcal{A}_\varphi]\!](u^j) \leq N$, so by the inductive hypothesis, $[\![\varphi]\!](u^j) \leq N$.

Since we have shown that there is some $i > 0$ and some set $I \subseteq [1, i-1]$ such that $[\![\psi]\!](u^i) \leq N$, $|I| \leq N$, and for all $1 \leq j < i$ with $j \notin I$, $[\![\varphi]\!](u^j) \leq N$, we have $[\![\theta]\!](u) \leq N$ as desired.

The other cases are similar.

Alternatively, this construction could be done with k counters and using only actions $\mathtt{ic}$ and ε.

B.5 B-VWAA to CLTL

Let $\mathcal{A} := \langle Q, \mathbb{A}, q_0, \delta, F\rangle$ be a B-VWAA, u a word in $\mathbb{A}^\omega$, and R a tree representing a run of $\mathcal{A}$ over u. We remind that R fixes the choices of Player Min, and the branching correspond to the choices of Player Max. However, at each depth, several vertices can be labelled by the same state in Q : this happens when Max can choose between several transitions to the same state. If we merge all such vertices (that have same depth in R and labelled by the same state q), we obtain a direct acyclic graph of width at most $|Q|$, describing completely the run R. This graph will be called a *run DAG* of $\mathcal{A}$ over u.

Proof of Lemma 2. By [4], we can assume that $\mathcal{A} := \langle Q, \mathbb{A}, q_0, \delta, F\rangle$ uses k hierarchical counters. We write $\mathbb{C}$ for the set of hierarchical counter actions for k counters, ordered by $\mathtt{R}_k > \mathtt{IC}_k > \cdots > \mathtt{R}_1 > \mathtt{IC}_1 > \mathtt{R}_0 = \varepsilon$.

Each $\delta(q, a)$ can be written as a disjunction of conjunctions of elements of $\mathbb{C} \times Q$. A single disjunct can be written in the form $(c_1, q) \wedge (c_2, q) \wedge \ldots \wedge (c_j, q) \wedge \psi$ where ψ is a conjunction of elements (c, q') for $c \in \mathbb{C}$ and $q' \neq q$.

Let $\delta'(q, a)$ be the result of performing the following transformations on each disjunct in $\delta(q, a)$:

1. replace each conjunct (c, q') for $q' \neq q$ in ψ with $(\mathtt{R}_k, q')$.
2. contract $(c_1, q) \wedge (c_2, q) \wedge \ldots \wedge (c_j, q)$ to a single conjunct (c, q) such that
   (a) if there any c_i which is an increment, then $c := \mathtt{IC}_j$ for j the highest counter which is incremented;
   (b) if there are no increments, then $c := \mathtt{R}_j$ for j the lowest counter which is reset.

Let $\mathcal{A}' := \langle Q, \mathbb{A}, q_0, \delta', F\rangle$.

It is clear that $[\![\mathcal{A}']\!] \leq [\![\mathcal{A}]\!]$ since the new automaton has only added resets and removed conjuncts (making it easier for there to be a run of low value).

Now assume that $[\![\mathcal{A}']\!]$ is bounded by N on some set U of input words. We prove that $[\![\mathcal{A}]\!](U) \leq \alpha(N)$ where $\alpha(n) = |Q| \cdot (N+1)^k$.

We proceed by induction on $|Q|$, proving that given a run DAG of $\mathcal{A}'$ on u of value N, we can construct a run DAG of $\mathcal{A}$ on u of value at most $\alpha(N)$.

If $|Q| = 1$, then the run DAG witnessing $[\![\mathcal{A}']\!](u) \leq N$ has a (possibly infinite) spine π' which stays in q, along with side branches that may terminate in true. Consider a single node $y = (c, q)$ with parent x and child z on π'. It has a corresponding expansion $(c_1, q), (c_2, q), \ldots, (c_j, q)$ from $\mathcal{A}$. We can replace this single node in the run DAG with the expansion, creating an edge from x to each element of $\{(c_1, q), \ldots, (c_j, q)\}$, and from each of these elements to the child z.

We do this sort of replacement simultaneously for all nodes on π'. This results in a run DAG for $\mathcal{A}$ on u.

We claim that this run DAG has value at most $(N+1)^k$. Notice that by the contraction rules, π' still describes a path through this new run DAG. Now consider another possible path π in the run DAG for $\mathcal{A}$ on u and assume for the sake of contradiction that π has value at least $(N+1)^k+1$. Then there is some segment of π witnessing at least $(N+1)^k+1$ increments for some counter j, no resets for counter j, and no counter higher than j is touched. Let v denote the sequence of counter actions on this segment in π, and let v' denote the corresponding sequence of counter actions on this same segment in π'.

If $v(i) = \mathtt{IC}_l$ then $v'(i)$ must be $\mathtt{IC}_{l'}$ for some $l' \geq l$ by contraction rule (a). This means that there are at least $(N+1)^k+1$ increments for counters $j' \geq j$ on v'.

If some counter j' for $j' \geq j$ is explicitly reset at $v'(i)$, then there cannot be an increment at $v(i)$ by (a), so there must be a reset. But by the choice of v, this reset must be of some counter lower than j. But then contraction rule (b) would imply that $v'(i)$ is a reset for some counter less than j, which is a contradiction.

Hence, there are at least $(N+1)^k+1$ increments for counters $j' \geq j$ and there is no explicit reset of counters $j' \geq j$ on v'. This means that the only way a counter j' for $j' \geq j$ is reset on this segment is when a counter $j'' > j'$ is incremented. But the maximum number of increments possible across all counters on v' without using explicit resets and while still keeping the overall value at most N is $(N+1)^k$.[3] Hence some counter must be incremented more than N times on v', contradicting the fact that $[\![\mathcal{A}']\!](u) \leq N$ (since π' is a path in the run DAG for $\mathcal{A}'$ on u).

If $|Q| > 1$, then consider the run DAG witnessing $[\![\mathcal{A}']\!](u) \leq N$. Let G' be the partial run DAG of $\mathcal{A}'$ on u consisting entirely of nodes labelled with (c, q). As in the previous case, we can expand the nodes in G' to yield a partial run DAG G for $\mathcal{A}$ on u of value at most $(N+1)^k$. By the inductive hypothesis, we can construct a run DAG for each successor of the terminal nodes of G' which starts in some state $q' \neq q$ and witnesses cost at most $(|Q|-1)\cdot(N+1)^k$ (this is taking advantage of the fact that $\mathcal{A}'$ is very-weak). Combining the partial run DAG G with the run DAGs from the inductive hypothesis results in a run DAG for $\mathcal{A}$ on u where every path has value at most $(N+1)^k + (|Q|-1)\cdot(N+1)^k \leq \alpha(N)$ as desired.

Proof of Proposition 2. The proof is by induction on $|Q|$. We prove that $[\![\mathcal{A}_\varphi]\!] \leq [\![\theta(\varphi)]\!]$ and $[\![\theta(\varphi)]\!] \preccurlyeq_\alpha [\![\mathcal{A}_\varphi]\!]$ for $\alpha(n) = (n+1)^k$.

[3] Technically, the maximum number M_k of increments across k counters without using explicit resets and while still keeping the overall value below N is defined inductively to be $M_1 = N$ for one counter and $M_k = N \cdot M_{k-1} + N$. This can be achieved by nesting the increments, i.e. for two counters, there would be N groups consisting of N increments of counter 1 followed by an increment of counter 2, accounting for a total of $N^2 + N$ increments. Note that the value M_k is bounded by $(N+1)^k$ for all $k \geq 1$.

If $|Q| = 0$, then the only possibilities for φ are positive boolean combinations of true or false and we let $\theta(\mathsf{true}) = \mathsf{true}$, $\theta(\mathsf{false}) = \mathsf{false}$, $\theta(\psi_1 \wedge \psi_2) = \theta(\psi_1) \wedge \theta(\psi_2)$, and $\theta(\psi_1 \vee \psi_2) = \theta(\psi_1) \vee \theta(\psi_2)$.

Otherwise, let $|Q| > 0$ and let q be the highest state in the very-weak ordering. Given some φ, 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 consider $\delta(q, a)$ for $a \in \mathbb{A}$, which can be written in disjunctive normal form as $\bigvee_{c \in \mathbb{C}} \varphi_{q,a,c} \vee \varphi_{q,a,\text{exit}}$ such that

- $\varphi_{q,a,c}$ for $c \in \mathbb{C}$ is a (possibly empty) disjunction of conjunctions such that in each disjunct there is a conjunct of the form (c, q) (there is only one such conjunct by Lemma 2);
- $\varphi_{q,a,\text{exit}}$ is a (possibly empty) disjunction of conjunctions of elements $(\mathtt{R}_k, q')$ for $q' \neq q$.

This partitions the disjuncts into sets depending on whether the disjunct includes a conjunct which remains in q, or includes only conjuncts which go to a lower state in the ordering. Within the disjuncts which have a conjunct which remains in q, we partition these based on the operation associated with q (there is only one such operation because of Lemma 2).

We let $\tilde{\varphi}_{q,a,\text{exit}}$ be false if $\varphi_{q,a,\text{exit}}$ is empty; otherwise, it is $\varphi_{q,a,\text{exit}}$ with elements $(\mathtt{R}_k, q')$ replaced by q'. We let $\tilde{\varphi}_{q,a,c}$ be false if $\varphi_{q,a,c}$ is empty; otherwise, it is $\varphi_{q,a,c}$ with all conjuncts (c, q) replaced by true and all conjuncts $(\mathtt{R}_k, q')$ replaced by q'. Because these $\tilde{\varphi}$ versions do not contain q, the inductive hypothesis can be applied to them. We can use this to define

- $\theta_{q,c} := \bigvee_{a \in \mathbb{A}} a \wedge \mathbf{X}\theta(\tilde{\varphi}_{q,a,c})$
- $\theta_{q,\text{exit}} := \bigvee_{a \in \mathbb{A}} a \wedge \mathbf{X}\theta(\tilde{\varphi}_{q,a,\text{exit}})$

which says that when reading $a \in \mathbb{A}$, a disjunct in $\delta(q, a)$ is chosen which falls into one of the sets described above.

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 γ, this behaviour is approximated by

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

This is only an approximation of the value since it requires an explicit reset of some counter $\gamma' \geq \gamma$; it does not recognize when the counter is reset due to an increment of a higher counter. This is to prevent this formula from achieving an artificially low value for some γ and γ' with $\gamma' > \gamma$ by using $\theta_{q,\mathtt{IC}_{\gamma'}}$ formula to avoid making a mistake on some $\theta_{q,\text{cycle},\gamma}$, and simultaneously using $\theta_{q,\mathtt{IC}_\gamma}$ to avoid making a mistake on $\theta_{q,\text{cycle},\gamma'}$. Up to a correction function of $\alpha(n) = (N+1)^k$, this is equivalent.

Putting this together for all counters, we have

$$\theta_{q,\text{cycle}} := \bigvee_{c \in \mathbb{C}} \theta_{q,c} \wedge \bigwedge_{\gamma \in [1,k]} \theta_{q,\text{cycle},\gamma}.$$

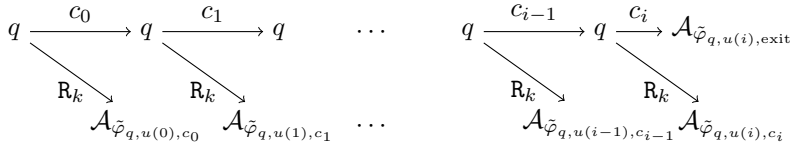
Finally, this gets wrapped into a statement which ensures correct behaviour in terms of accepting states (i.e. if $q \notin F$ then the play cannot stay forever in q):

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

Now assume that $[\![\theta(q)]\!](u) \leq N$. We must prove that $\mathcal{A}_q$ has an accepting run DAG starting from q with value at most N. Assume that $q \notin F$. Then there is some i such that $[\![\theta_{q,\text{exit}}]\!](u^i) \leq N$ and for all $j < i$, $[\![\theta_{q,\text{cycle}}]\!](u^j) \leq N$. Using the inductive hypothesis, this means that $[\![\mathcal{A}_{\tilde{\varphi}_{q,u(i),\text{exit}}}]\!](u^{i+1}) \leq N$. It remains to show the existence of a partial run DAG on $u(0)\dots u(i)$ which terminates in $\tilde{\varphi}_{q,u(i),\text{exit}}$.

In order to define this run DAG, we must select an action c_j to associate to each position $j < i$. We do this based on the values of formulas $\theta_{q,c}$ at those positions. If there is some counter γ such that $[\![\theta_{q,\mathtt{R}_\gamma}]\!](u^j) \leq N$, then let c_j be $\mathtt{R}_\gamma$ for the highest such γ (note that in this case, the subformulas $\theta_{q,\text{cycle},\gamma'}$ can avoid making any mistake at that position by using $\theta_{q,\mathtt{R}_\gamma}$). Otherwise, let c_j be $\mathtt{IC}_\gamma$ for γ the lowest counter such that $[\![\theta_{q,\mathtt{IC}_\gamma}]\!](u^j) \leq N$ (there must be some γ like this since $[\![\theta_{q,\text{cycle}}]\!](u^j) \leq N$ implies $[\![\bigvee_{c\in\mathbb{C}} \theta_{q,c}]\!](u^j) \leq N$).

Applying the inductive hypothesis, this means that $[\![\mathcal{A}_{\tilde{\varphi}_{q,u(j),c_j}}]\!](u^{j+1}) \leq N$. Consider the run DAG consisting of a spine of transitions from q to q with action c_j when reading $u(j)$ and terminating in the run DAG for $\mathcal{A}_{\tilde{\varphi}_{q,u(i),\text{exit}}}$ after reading $u(i)$, and with edges at each j to the run DAGs from $\mathcal{A}_{\tilde{\varphi}_{q,u(j),c_j}}$ (see diagram below). The run DAGs from the subautomata witness value at most N, and reaching these subautomata resets the counter, so it remains to prove that the path which stays on the spine until position i has at most N increments between resets. Assume not. Then there is some segment starting at position j' which has more than N increments for some counter γ and no resets for γ. By the choice of c_j, $[\![\theta_{q,\mathtt{R}_{\gamma'}}]\!](u^l) > N$ for all positions l on this segment and for all $\gamma' \geq \gamma$. Likewise, at the positions l corresponding to increments for γ on this segment, it must be the case that $[\![\theta_{q,\mathtt{R}_{\gamma'}}]\!](u^l) > N$ for all γ' (otherwise there would be a reset at this position by the definition of c_l), and $[\![\theta_{q,\mathtt{IC}_{\gamma'}}]\!](u^l) > N$ for all $\gamma' \leq \gamma$. This means that $\theta_{q,\text{cycle},\gamma}$ must make more than N mistakes on this segment starting from j'. This contradicts the fact that $[\![\theta_{q,\text{cycle}}]\!](u^{j'}) \leq N$.



In the other direction, we prove that if $[\![\mathcal{A}_q]\!](u) \leq N$ then $[\![\theta(q)]\!](u) \leq \alpha(N)$. Assume that there is a run DAG of $\mathcal{A}_q$ of value N, and assume that $q \notin F$. There must be some position i where there are no transitions from q to q (otherwise the run DAG would have value ∞). This can be viewed as a run of $\mathcal{A}_{\tilde{\varphi}_{q,u(i),\text{exit}}}$

on u^{i+1}, so by the inductive hypothesis, $[\![\theta(\tilde{\varphi}_{q,u(i),\mathrm{exit}})]\!](u^{i+1}) \leq (N+1)^k$. For $j < i$, let c_j denote the action in the run DAG from q to q when reading $u(j)$. We must prove $[\![\theta_{q,\mathrm{cycle}}]\!](u^j) \leq (N+1)^k$ for all $j < i$.

By the definition of c_j and the fact that the run DAG witnesses value at most N, $\mathcal{A}_{\tilde{\varphi}_{q,u(j),c_j}}(u^{j+1}) \leq N$. By the inductive hypothesis, this implies that $[\![\theta(\tilde{\varphi}_{q,u(j),c_j})]\!](u^{j+1}) \leq (N+1)^k$, which means that $[\![\theta_{q,c_j}]\!](u^j) \leq (N+1)^k$ for all $j < i$. Any subword of $c(0)c(1)\dots c(i)$ can have at most N increments for a given counter between resets (either explicit, or because a higher counter has been touched). For a particular counter γ, this can translate into at most $(N+1)^k$ errors in $\theta_{q,\mathrm{cycle},\gamma}$ (since resets coming from increments of higher counters do not allow the number of mistakes to be reset, but the number of such increments of higher counters is bounded). Hence, $[\![\theta_{q,\mathrm{cycle}}]\!](u^j) \leq (N+1)^k$ holds for each $j < i$.

The proof when $q \in F$ is similar.

B.6 *S*-Automata

We briefly recall the definition of s-automata, see [4] for more details.

As before, we use counters from a set Γ, that are initially assigned value 0.

This time, the atomic actions are $\mathbb{S} := \{\mathtt{i}, \varepsilon, \mathtt{r}, \mathtt{cr}\}$. We still collect all checked values of a word of $u \in \mathbb{S}^\omega$ in a set $C(u)$, but in the S-semantic, we will be interested in the infimum of these values: $val_S(u) := \inf C(u)$.

The global counter actions are defined as $\mathbb{C}_S := \mathbb{S}^{\Gamma_S}$, acting simultaneously on all counters.

An *alternating S-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 Γ of S-counters, and a transition function $\delta : Q \times \mathbb{A} \to \mathcal{B}^+(\mathbb{C}_S \times Q)$ (where $\mathcal{B}^+(\mathbb{C} \times Q)$ is the set of positive boolean combinations, written as a disjunction of conjunctions of elements $(c_i, q_i) \in \mathbb{C}_S \times Q$).

A run of $\mathcal{A}$ on $(a_i)_{i\in\mathbb{N}} \in \mathbb{A}^\omega$ is a labelled tree $R = (r, c)$ with $r : dom(R) \to Q$ and $c : (dom(R) \setminus \{\epsilon\}) \to \mathbb{C}$ such that

- $r(\epsilon) = q_0$;
- 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 φ such that for all $1 \leq j \leq k$, $x \cdot j \in dom(R)$, $r(x \cdot j) = q_j$ 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 π in $R = (r, c)$, there are infinitely-many positions x such that $r(x) \in F$. This means that no matter

We assign S-values to runs as follows. Given a branch $\pi = (x_i)_i \in \omega$ in a run R, the value of π is $val_S(\pi) = val_S(c(x_1)c(x_2)\dots)$ and $val_S(R)$ is the infimum over the values of the branches.

Notice that this time, Player Or tries to maximize the value of the run. That is why if the run do not satisfy the Büchi condition, the value is 0 and this Player has failed.

If δ 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. Then the run is accepting if there are infinitely many Büchi states on its unique branch π, and its value is the value of π.

The *S-semantic* of an S-automaton $\mathcal{A}$ is $[\![\mathcal{A}]\!]_S : \mathbb{A}^\omega \to \mathbb{N}_\infty$

$$[\![\mathcal{A}]\!]_S(u) := \sup \{val_S(R) : R \text{ is an accepting run of } \mathcal{A} \text{ on } u\}$$

The idea is that the S-semantic maximizes the value over S-accepting runs.

C From B-NBA to S-NBA

Let $\mathcal{B} = \langle Q, \mathbb{A}, F, q_0, \Gamma, \Delta \rangle$ be a B-NBA, and $f = [\![\mathcal{S}]\!]$. We want to build a S-NBA $\mathcal{S}$ recognizing f.

C.1 Semigroup of actions

Let $\mathbb{B}$ be the set of atomic B-actions. We define $\mathbb{C}$ to be an extension of $\mathbb{B}^\Gamma$, describing the global action of an arbitrary part of a run.

Let $\mathbb{C}_1 := \{\mathtt{r}, \varepsilon, \mathtt{ic}, \bot\}$, and $\mathbb{C} = \mathbb{C}_1^\Gamma$.

Action $\bot$ corresponds to a lot of increments, i.e. a fail of the run.

We give to $\mathbb{C}_1$ a semigroup structure by the following product operation, which represents the concatenation of two global actions. The left column representing the left operand. We also define a unary operation ω on $\mathbb{C}_1$.

$\cdot$	$\mathtt{r}$	ε	$\mathtt{ic}$	$\bot$	$\cdot^\omega$
$\mathtt{r}$	$\mathtt{r}$	$\mathtt{r}$	$\mathtt{r}$	$\bot$	$\mathtt{r}$
ε	$\mathtt{r}$	ε	$\mathtt{ic}$	$\bot$	ε
$\mathtt{ic}$	$\mathtt{r}$	$\mathtt{ic}$	$\mathtt{ic}$	$\bot$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

We also define an order by $\mathtt{r} \leq \varepsilon \leq \mathtt{ic} \leq \bot$.

The product, order and ω-power of $\mathbb{C}$ are then defined component-wise, relative to each counter.

C.2 Types

We define the set of *types* $\mathcal{T}$ to be the $\leq$-closed subsets of $Sig := Q^2 \times \mathbb{C} \times \{0, 1\}$, where $\leq$ is defined on Sig by $(p, q, c, b) \leq (p', q', c', b')$ if $(p, q, b) = (p', q', b')$ and $c \leq c'$. Elements of Sig will be called *signatures*.

Intuitively, for $n \in \mathbb{N}$, the n-type of a word $u \in \mathbb{A}^*$, noted $T_n(u) \in \mathcal{T}$, will contain signature (p, q, c, b) iff the automaton $\mathcal{B}$ can go from p to q by reading u, doing global action at most $c \in \mathbb{C}$, where action $\bot$ corresponds to values bigger than n, and seeing a Büchi state iff $b = 1$.

Note than for any $u \in \mathbb{A}^*$ and $n \leq m$, we have $T_n(u) \subseteq T_m(u)$.

$(\mathcal{T}, \cdot, \omega, \mathcal{T}^\omega)$ has the structure of an ω-semigroup. The product is defined as $t_1 \cdot t_2 := \{(p, r, c_1 c_2, b_1 \vee b_2) : \exists q, (p, q, c_1, b_1) \in t_1 \text{ and } (q, r, c_2, b_2) \in t_2\}$. And the ω-power on idempotents: $E(\mathcal{T}) \to \mathcal{T}^\omega = 2^{Q \times \mathbb{C} \times \{0,1\}}$ by

$$t^\omega = \{(p, c_1 c^\omega, b) : \exists q, b_1 \in Q \times \{0,1\}, (p, q, c_1, b_1) \in t \text{ and } (q, q, c, b) \in t\}.$$

We also define a mixed product $\cdot : \mathcal{T} \times \mathcal{T}^\omega \to \mathcal{T}^\omega$ by
$s \cdot t = \{(p, c \cdot c', b) : (p, q, c, b_0) \in s \text{ and } (q, c', b) \in t\}$.

Elements of $\mathcal{T}^\omega$ describe sets of infinite runs, each one described by some (p, c, b): p is the starting state, c describes the global action performed by the run, and b is 1 if infinitely many accepting state are seen.

We finally define *accepting types* Acc $\subseteq \mathcal{T} \times E(\mathcal{T})$ to be the set of pairs (s, t) such there is $c \in \mathbb{C}$ with $(q_0, c, 1) \in s \cdot t^\omega$ and for all $\gamma \in \Gamma$, $c(\gamma) \neq \bot$. Intuitively, (s, t) is accepting means that if $u = u_0 u_1 u_2 \dots$ with $s \subseteq T_n(u_0)$ and for all $i \geq 1$, $t \subseteq T_n(u_i)$, then $f(u) \leq n$ because the types describe a valid n-run of $\mathcal{B}$ on u. We call Ref the complement of Acc in $\mathcal{T} \times E(\mathcal{T})$.

Notice that Ref is $\subseteq$-closed, in the sense that if $(s, t) \in$ Ref and $s' \subseteq s$, $t' \subseteq t$ then $(s', t') \in$ Ref.

C.3 Back to finite words

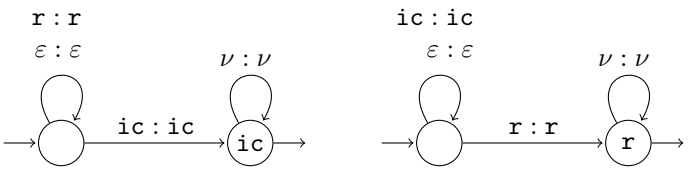
If $t \in \mathcal{T}$, let g_t be the cost function on $\mathbb{A}^*$ defined by

$$g_t(u) = \inf \{n : T_n(u) \not\subseteq t\}.$$

We show that for all $t \in \mathcal{T}$, g_t is a regular cost function on finite words, by building a B-automaton $\mathcal{B}_t$ for g_t. It suffices to design a B-automaton for one $(p, q, c, b) \notin t$, that checks that $(p, q, c, b) \in T_n(u)$. We can then do the union of all those automata. Given $s = (p, q, c, b) \notin t$, the B-automaton $\mathcal{B}_{(p,q)}$ has the same states and transitions as $\mathcal{B}$, with p as initial state, q as final state. We also need to verify that global action at least c is performed. For each $c \in \mathbb{C}_1$, we design an automaton $\mathcal{A}_c$ on finite words, which accepts a sequence of action with value n if the global action is at most c, with threshold n for switching form `ic` to $\bot$. Automata $\mathcal{A}_c$ with $c \in \mathbb{C}$ are then simply products of automata for atomic actions.

We also define a two-state automaton $\mathcal{A}_b$ remembering if a Büchi state has been seen. The desired B automaton $\mathcal{S}_s$ can finally be defined as the composition of $\mathcal{B}_{(p,q)}$ and $\mathcal{B}_c \times \mathcal{A}_b$, which will analyse the output of $\mathcal{B}_{(p,q)}$.

Here are the automata $\mathcal{B}_{\mathtt{ic}}$ and $\mathcal{B}_{\mathtt{r}}$ for one counter, where $\nu \in \{\mathtt{ic}, \varepsilon, \mathtt{r}\}$.



Since the g_t are regular cost functions on finite words, for each $t \in \mathcal{T}$ we can build a S-automaton $\mathcal{A}_t$ recognizing g_t.

Finally, let $g'_t(u) := \sup\{n : T_n(u) \subseteq t\}$, and remark that for all $t \in \mathcal{T}$, $g'_t \leq g_t \leq g'_t + 1$. It means that $g_t \approx g'_t$, and so $[\![\mathcal{A}_t]\!] \approx g'_t$.

C.4 From finite to infinite words

If $t \in \mathcal{T}$ and $n \in \mathbb{N}$, let $L^n_t := \{u \in \mathbb{A}^* : T_n(u) \subseteq t\}$.

Lemma 10. *Let $n \in \mathbb{N}$, then*

$$\mathbb{A}^\omega = \bigcup_{(s,t) \in \mathcal{T} \times E(\mathcal{T})} L^n_s (L^n_t)^\omega. \quad \textbf{(1)}$$

Moreover there is α such that for all n,

- *if $(s,t) \in$ Acc and $u \in L^n_s(L^n_t)^\omega$, then $f(u) \leq \alpha(n)$* ***(2)****,*
- *if $(s,t) \in$ Ref and $u \in L^n_s(L^n_t)^\omega$, then $f(u) > n$* ***(3)****.*

Proof. **(1)**: Let $n \in \mathbb{N}$ and $u \in \mathbb{A}^\omega$. We have to show that there exists $(s,t) \in \mathcal{T} \times E(\mathcal{T})$ such that $u \in L^n_s(L^n_t)^\omega$. $T_n : \mathbb{A}^* \to \mathcal{T}$ is a semigroup morphism, so by a Ramsey theorem we can conclude that there is $s \in \mathcal{T}$ and t idempotent in $\mathcal{T}$ such that $u = u_0u_1u_2\dots$ with $T_n(u_0) = s$ and for all $i \geq 1$, $T_n(u_i) = t$. We get that $u \in L^n_s(L^n_t)^\omega$.

(2): We assume $u \in L^n_s(L^n_t)^\omega$ for some $(s,t) \in$ Acc. It means that there is c such that $(q_0, c, 1) \in s \cdot t^\omega$ and for all $\gamma \in \Gamma$, $c(\gamma) \neq \bot$.

By definition of products and ω-power on $\mathcal{T}$, we have the existence of $(p_0, q, c_0, b_0) \in s$ and $(q, q, c', 1) \in t$ with $c_0 \cdot c'^\omega = c$.

But $u \in L^n_s(L^n_t)^\omega$, so u can be written $u_0u_1u_2\dots$ with $T_n(u_0) = s$ and for all $i \geq 1$, $T_n(u_i) = t$. This means that there is a n-run of $\mathcal{B}$ from p_0 to q on $u_0u_1\dots u_k$ for some k doing global action c_0, and moreover $\mathcal{B}$ can read each u_i while doing a n-loop around q containing a Büchi state, with action $c' < \texttt{ic}$. By definition of the action product, this describes an accepting $2n$-run of $\mathcal{B}$ on u. The 2 factor comes from the gluing of two n-paths containing an $\texttt{r}$. We conclude that $f(u) \leq 2n$.

(3): We show the result by contraposition. Assume $f(u) \leq n$. Let $u = u_0u_1u_2\dots$ be any Ramsey decomposition of u with respect to the semigroup morphism T_n. Let $s = T_n(u_0)$ and $t = T_n(u_i)$ for all $i \geq 1$, with t idempotent.

There is an accepting n-run ρ of $\mathcal{B}$ over u. Let q be a state that appear infinitely many often as a frontier between the u_i's in ρ.

Since t is an idempotent and ρ is accepting with value at most n, we must have some $(p_0, p, c_0, b_0) \in s$, $(p, q, c_1, b_1) \in t$ and $(q, q, c, 1) \in t$ such that for all $\gamma \in \Gamma$, $c_0 \cdot c_1 \cdot c^\omega(\gamma) \neq \bot$. This implies $(s,t) \in$ Acc.

We showed that if $f(u) \leq n$, any Ramsey decomposition (s,t) with respect to T_n is in Acc. So by contraposition, if $(s,t) \in$ Ref and $u \in L^n_s(L^n_t)^\omega$, then $f(u) > n$.

□

Corollary 3. *Let g be the cost function on infinite words defined by*

$$g(u) := \sup \{n \in \mathbb{N} : \exists (s,t) \in \mathrm{Ref}, u \in L_s^n (L_t^n)^\omega\}.$$

Then $f \approx g$.

Proof. Let $u \in \mathbb{A}^\omega$ such that $f(u) > \alpha(n)$. By Lemma 10 there is $(s,t) \in \mathcal{T} \times E(\mathcal{T})$ such that $u \in L_s^n (L_t^n)^\omega$.

If $(s,t) \in \mathrm{Acc}$, we get by Lemma 10 that $f(u) \leq \alpha(n)$ which is absurd, so $(s,t) \in \mathrm{Ref}$ and $g(u) \geq n$.

Conversely, if $g(u) \geq n$, let $(s,t) \in \mathrm{Ref}$ such that $u \in L_s^n (L_t^n)^\omega$. Then by Lemma 10 $f(u) > n$. □

C.5 Construction of S-NBA $\mathcal{S}$

Let $s, t \in \mathcal{T}$, we define $g_{s,t}(u) = \sup \{n \in \mathbb{N} : u \in L_s^n (L_t^n)^\omega\}$

Lemma 11. *For all $s, t \in \mathcal{T}$, there is an S-NBA $\mathcal{S}_{s,t}$ recognizing $g_{s,t}$.*

Proof. For all $r \in \mathcal{T}$, we had $[\![\mathcal{A}_r]\!] \approx_{\alpha_r} g'_r$ for some α_r. Let $\alpha_{s,t} = \max(\alpha_s, \alpha_t)$. The automaton $\mathcal{S}_{s,t}$ just needs to guess the factorization of u witnessing $g_{s,t}(u) \geq n$, and independently run $\mathcal{A}_s$ on u_0 and $\mathcal{A}_t$ on each u_i, checking that they accept with value at least n. The Büchi states are at the end of each u_i, forcing the automaton to guess infinitely many of them.

If $[\![\mathcal{S}_{s,t}]\!](u) \geq \alpha_{s,t}(n)$, it means that there is a factorisation $u = u_0 u_1 u_2 \dots$ with $[\![\mathcal{A}_s]\!](u_0) \geq \alpha_s(n)$ and $[\![\mathcal{A}_t]\!](u_i) \geq \alpha_t(n)$ for all $i \geq 1$.

It means that $g'_s(u_0) \geq n$ and $g'_t(u_i) \geq n$ for all $i \geq 1$. Hence $g_{s,t}(u) \geq n$. We got $[\![\mathcal{S}_{s,t}]\!] \preccurlyeq g_{s,t}$.

Conversely, assume $g_{s,t}(u) \geq n$. Then there is a factorisation $u = u_0 u_1 u_2 \dots$ with $g_s(u_0) \geq n$ ans $g_t(u_i) \geq n$ for all $i \geq 1$. By guessing this factorisation, $\mathcal{S}_{s,t}$ has a run of value at least n on u.

Finally, $[\![\mathcal{S}_{s,t}]\!] \approx g_{s,t}$.

□

It now suffices to define $\mathcal{S} = \bigcup_{(s,t) \in \mathrm{Ref}} \mathcal{S}_{s,t}$. Since $g = \max_{(s,t) \in \mathrm{Ref}} g_{s,t}$, we can conclude that $\mathcal{S}$ recognizes f by Corollary 3.

C.6 From S-NBA to B-NBA

The converse follows the same proof scheme, but it gets a little more complicated.

Indeed, a partial run can have more different effects on a counter, and therefore the possible types are $\mathbb{C}_1 := \{\Omega, \mathtt{i}, \varepsilon, \mathtt{r}, \mathtt{cr}\Omega, \mathtt{cr}, \bot\}$, where Ω means "a lot of increments".

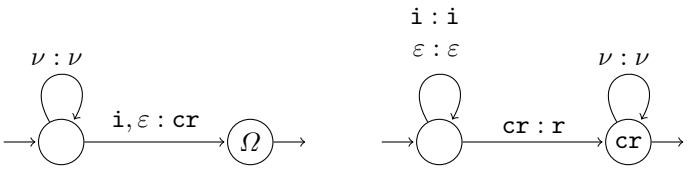
The composition law is the following:

$\cdot$	Ω	$\mathtt{i}$	ε	$\mathtt{r}$	$\mathtt{cr}\Omega$	$\mathtt{cr}$	$\bot$	$\cdot^\omega$
Ω	Ω	Ω	Ω	$\mathtt{r}$	Ω	$\mathtt{r}$	$\bot$	Ω
$\mathtt{i}$	Ω	$\mathtt{i}$	$\mathtt{i}$	$\mathtt{r}$	$\mathtt{cr}\Omega$	$\mathtt{cr}$	$\bot$	Ω
ε	Ω	$\mathtt{i}$	ε	$\mathtt{r}$	$\mathtt{cr}\Omega$	$\mathtt{cr}$	$\bot$	ε
$\mathtt{r}$	Ω	$\mathtt{r}$	$\mathtt{r}$	$\mathtt{r}$	$\bot$	$\bot$	$\bot$	$\mathtt{r}$
$\mathtt{cr}\Omega$	$\mathtt{cr}\Omega$	$\mathtt{cr}\Omega$	$\mathtt{cr}\Omega$	$\mathtt{cr}$	$\mathtt{cr}\Omega$	$\mathtt{cr}$	$\bot$	$\mathtt{cr}\Omega$
$\mathtt{cr}$	$\mathtt{cr}\Omega$	$\mathtt{cr}$	$\mathtt{cr}$	$\mathtt{cr}$	$\bot$	$\bot$	$\bot$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

The set of accepting types $\mathrm{Acc} \subseteq \mathcal{T} \times E(\mathcal{T})$ becomes the set of pairs (s,t) such there is $c \in \mathbb{C}$ with $(q_0, c, 1) \in s \cdot t^\omega$ and for all $\gamma \in \Gamma$, $c(\gamma) \notin \{\mathtt{cr}, \mathtt{cr}\Omega, \bot\}$: a check at the beginning of the word is not allowed.

We also need to design S-automata on finite words that recognize each of these types.

Types starting with $\mathtt{cr}$ or ending with Ω are a little puzzling: here are the automata $\mathcal{A}_\Omega$ and $\mathcal{A}_{\mathtt{cr}}$ for one counter, where $\nu \in \{\mathtt{i}, \varepsilon, \mathtt{r}, \mathtt{cr}\}$.



Remark 1.

- If c ends with Ω, we cannot use the states to verify that we reached a high value. However, $\mathcal{A}_c$ can perform action $\mathtt{cr}_\gamma$ at the end of the word, as it done in $\mathcal{A}_\Omega$. This way, the run will be accepting if and only if a high value is reached.
- Likewise, if c begins with $\mathtt{cr}$, then when see the first $\mathtt{cr}$ in $\mathcal{S}$, we do not perform $\mathtt{cr}$ in $\mathcal{A}_s$, but we remember that we saw a $\mathtt{cr}$ in the input, as it is done in $\mathcal{A}_{\mathtt{cr}}$.

Then again, we can use the result of [4] on finite words: B and S-automata have same expressive power. This gets us a B-automaton $\mathcal{B}_t$ for any type t. Combining these automata with respect to accepting types finally gets us the wanted B-automaton.

The proof of correctnes is almost identical to the previous one, we just need to reverse some inequalities. Indeed going from B to S switches between inf and sup.

For instance Lemma 10 becomes

Lemma 12. *Let $n \in \mathbb{N}$, then*

$$\mathbb{A}^\omega = \bigcup_{(s,t) \in \mathcal{T} \times E(\mathcal{T})} L_s^n (L_t^n)^\omega. \quad \textbf{(1)}$$

Moreover there is α such that for all n,

- *if $(s,t) \in \mathrm{Acc}$ and $u \in L_s^n (L_t^n)^\omega$, then $f(u) > n$* ***(2)****,*
- *if $(s,t) \in \mathrm{Ref}$ and $u \in L_s^n (L_t^n)^\omega$, then $f(u) \leq \alpha(n)$* ***(3)****.*

D Equivalence between B-NBA and CMSO

In this part we will use for convenience the alternative definition of CMSO, with predicate "$|X| \leq N$".

D.1 From B-NBA to CMSO

Let $\mathcal{B} = \langle Q, \mathbb{A}, F, q_0, \Gamma, \Delta \rangle$ be a B-NBA. We want to build a CMSO-formula $\varphi_\mathcal{B}$ recognizing $[\![\mathcal{B}]\!]$.

Since both the semantics (CMSO and B-automata) are defined as an infimum, the formula $\varphi_\mathcal{B}$ just needs to express that there is a run of $\mathcal{B}$ on u of value at most N.

Let $\Delta = \{\delta_1, \dots, \delta_k\} \subseteq Q \times \mathbb{A} \times \mathbb{C} \times Q$. For each $\delta = (p, a, \nu, q) \in \Delta$, we will use a second-order variable X_δ to describe the set of positions where δ is used. We will note $\pi_1, \pi_\mathbb{A}, \pi_\mathbb{C}, \pi_2$ the projections of δ onto its components. For each $\gamma \in \Gamma$, we also define $\pi_\gamma(\delta)$ to be the projection of $\pi_\mathbb{C}(\gamma)$ on the counter γ.

We then define the following auxiliary formula, having $X_{\delta_1}, \dots, X_{\delta_k}$ as free variables :

- **Partition** $:= \forall x, \bigvee_{1 \leq i \leq k} (x \in X_{\delta_i} \wedge (\bigwedge_{j \neq i} x \notin X_{\delta_j}))$.
  This formula expresses that the X_{δ_i}'s form a partition of the set of positions.
- **Word** $:= \forall x, \bigvee_{a \in \mathbb{A}} (a(x) \wedge \bigvee_{\pi_\mathbb{A}(\delta) = a} x \in X_\delta))$.
  This formula ensure the coherence between the input word and the sequence of transitions described by the X_δ's.
- **Init** $:= \bigvee_{\pi_1(\delta) = q_0} (\exists x, x \in X_\delta \wedge \forall y, x \leq y)$.
  This formula ensures that the first transition starts from the initial state q_0.
- **Büchi** $:= \bigvee_{\pi_2(\delta) \in F} (\forall x, \exists y, y \in X_\delta \wedge x \leq y)$.
  This formula expresses that there is a transition towards a Büchi state that occurs infinitely often.
- For each $\gamma \in \Gamma$, we define a formula
  $\mathbf{Measure}_\gamma(X) := (\forall x, x \in X \Rightarrow \bigvee_{\pi_\gamma(\delta) = \mathtt{ic}} x \in X_\delta) \wedge$
  $\forall x, y, z (x \in X \wedge y \in X \wedge x \leq z \leq y) \Rightarrow \bigvee_{\pi_\gamma(\delta) \in \{\mathtt{ic}, \varepsilon\}} z \in X_\delta)$
  expressing that X marks consecutive increments (possibly separated by ε's) for counter γ.
- **Cost** $:= \bigwedge_{\gamma \in \Gamma} (\forall X, \mathbf{Measure}_\gamma(X) \Rightarrow |X| \leq N)$.
  This formula expresses that for each counter, there are at most N consecutive increments in the run.

We can finally define the formula

$$\varphi_\mathcal{B} := \exists X_{\delta_1}, \dots, X_{\delta_k}, \mathbf{Partition} \wedge \mathbf{Word} \wedge \mathbf{Init} \wedge \mathbf{Büchi} \wedge \mathbf{Cost}.$$

It is easy to see that $[\![\varphi_\mathcal{B}]\!] = [\![\mathcal{B}]\!]$, since for any $u \in \mathbb{A}^\omega$, runs of value at most n are in bijection with instanciations of $X_{\delta_1}, \dots, X_{\delta_k}$ witnessing $(u, n) \models \varphi_\mathcal{B}$.

D.2 From CMSO to B-Büchi

Here we want to show that for any CMSO formula φ, we can build a B-Büchi automaton $\mathcal{B}_\varphi$. This is done by using closure properties of the class of cost functions recognized by B- and S-Büchi automata, and the equivalence between these two formalisms.

Going back to the definition of CMSO, we had two types of formulas : type φ, where predicates "$|X| \leq N$" appear positively. We can also define the dual logic : $\overline{\text{CMSO}}$, where predicates "$|X| \leq N$" appear only negatively. We will note $\overline{\varphi}$ the formulas of $\overline{\text{CMSO}}$.

We can introduce the negation in the syntax, and define the two dual logics in the following way :

$$\varphi := a(x) \mid x \leq y \mid x \in X \mid \varphi \wedge \varphi \mid \exists x, \varphi \mid \exists X, \varphi \mid \neg\overline{\varphi} \mid |X| \leq N,$$
$$\overline{\varphi} := a(x) \mid x \leq y \mid x \in X \mid \overline{\varphi} \wedge \overline{\varphi} \mid \exists x, \overline{\varphi} \mid \exists X, \overline{\varphi} \mid \neg\varphi.$$

We saw that formulae φ naturally correspond to B-automata, since in both cases, the semantic is defined in term of an infimum.

In the same manner, we can give a semantic to a formula $\overline{\varphi}$, by $[\![\overline{\varphi}]\!](u) := \sup\{n \in \mathbb{N}, (u,n) \models \overline{\varphi}\}$. The natural automata model for formulas $\overline{\varphi}$ of $\overline{\text{CMSO}}$ is now S-automata, since the semantics is defined as a supremum in both cases.

Lemma 13. *If φ is a CMSO or $\overline{\text{CMSO}}$ formula, then $[\![\varphi]\!] \approx [\![\neg\varphi]\!]$.*

Proof. First notice that if φ is a CMSO formula, then $\overline{\varphi}$ is a $\overline{\text{CMSO}}$ formula, and vice-versa.

Assume φ is a CMSO formula.Let u be a word of $\mathbb{A}^\omega$. We recall that $[\![\varphi]\!](u) = \inf\{n \in \mathbb{N}, (u,n) \models \varphi\}$. Since predicates $|X| \leq N$ only appear positively in φ, we get that for all $k \leq n \in \mathbb{N}$, $(u,k) \models \varphi \implies (u,n) \models \varphi$. Two cases can now occur :

- If $[\![\varphi]\!](u) > 0$, the largest k such that $(u,k) \not\models \varphi$ is exactly $[\![\varphi]\!](u) - 1$. But notice that $(u,k) \not\models \varphi \iff (u,k) \models \neg\varphi$. Hence $[\![\neg\varphi]\!](u) = [\![\varphi]\!](u) - 1$.
- If $[\![\varphi]\!](u) = 0$, then $[\![\neg\varphi]\!](u) = \sup\emptyset = 0$.

In all cases $[\![\neg\varphi]\!](u) \leq [\![\varphi]\!](u) \leq [\![\neg\varphi]\!](u) + 1$, and this is true for all $u \in \mathbb{A}^\omega$, so $[\![\varphi]\!] \approx [\![\neg\varphi]\!]$.

The proof when φ is a $\overline{\text{CMSO}}$ formula is similar. □

Theorem 6. *If f is a cost function on infinite words, the following statements are equivalent :*

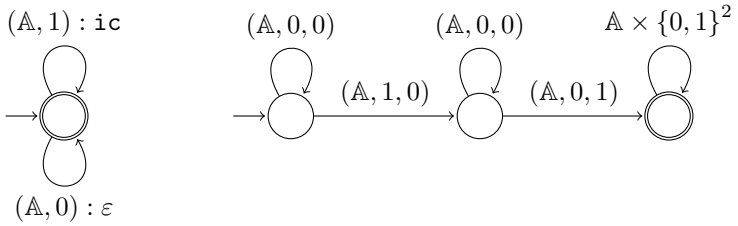
- *f is recognized by a B-Büchi automaton,*
- *f is recognized by an S-Büchi automaton,*
- *f is recognized by a formula of CMSO,*
- *f is recognized by a formula of $\overline{\text{CMSO}}$.*

Proof. We do this proof by induction on formulas. We start by using B-automata for CMSO formulae, and S-automata for $\overline{\text{CMSO}}$ formulae. When necessary, we can use equivalence between the two kinds of automata.

The automata may represent formulae with free variables, so an automaton $\mathcal{A}_\varphi$ will read words on alphabet $\mathbb{A}\times\{0,1\}^k$, where k is the number of free variable in φ. A word in $\{0,1\}^\omega$ describe a valuation for a set of positions, by marking with 1 the positions belonging to the set. First-order variables can be considered as singletons.

In this setting, it is easy to design classical Büchi automata for predicates $a(x)$, $x \leq y$, $x \in X$. Considered as B-automata, they recognize the same cost function as the corresponding formulae. We can also design Büchi automata for the complement of these properties. Considered as S-automata, they recognize the same cost function as the corresponding formulae.

The following B-automaton recognize $[\![|X| \leq N]\!]$, followed by a B-automaton for $x \leq y$ (as an example):



It remains to deal with the constructors $\neg, \wedge, \exists x, \exists X$. We showed that B- and S-automata have same expressive power on infinite words, so this settles the case of the negation. The other constructors correspond to the closure of the class of cost functions recognized by B- or S-automata by the operations $\min, \max,$ inf-projection and sup-projection. More precisely, existential quantificators in CMSO corresponds to inf-projections in CMSO, and to sup-projections in $\overline{\text{CMSO}}$, which explains the choice of B-automata for CMSO and S-automata for $\overline{\text{CMSO}}$. See [4, 18] for the closure of B-automata by $\min, \max,$ inf-projection and closure of S-automata by $\min, \max,$ sup-projection.

Using these constructions, we finally get a B-automaton $\mathcal{B}_\varphi$ for all formula φ of CMSO. □

E From B-NBA to B-WAA

Let $\mathcal{A} = \langle Q_\mathcal{A}, \mathbb{A}, I^\mathcal{A}, F_\mathcal{A}, \Gamma, \Delta_\mathcal{A}\rangle$ be a B-NBA, and $f = [\![\mathcal{B}]\!]$. We want to build a B-WAA recognizing f.

Here $F_\mathcal{A}$ is the set of Büchi states, so a run is accepting if infinitely many states from F are seen. If it is not the case the value of the run is ∞.

The proof scheme is the following: we start by building another B-NBA $\mathcal{B}$ in some normal form, recognizing f. Indeed the problem we encounter when dealing with cost automata is the interplay between counters and acceptance condition. Automaton $\mathcal{B}$ will avoid some of these problem, by establishing a link between counter actions and Büchi states.

We then extend the proof of [16]: we describe strategies proving that there is no n-run of $\mathcal{B}$ on some input. This allows us to build a B-WAA for f, where the opponent is able to play these strategies.

E.1 Normal form for B-NBA

A B-NBA $\mathcal{B} = \langle Q, \mathbb{A}, I, F, \Gamma, \Delta \rangle$ is in normal form if for any run ρ of $\mathcal{B}$, and for any counter $\gamma \in \Gamma$, all transitions leaving a Büchi state in ρ perform a reset on γ.

We will build a B-NBA $\mathcal{B}$ in normal form, with $[\![\mathcal{B}]\!] \approx f$.

The principle is that for all $n \in \mathbb{N}$ and $\gamma \in \Gamma$, any n-run of $\mathcal{A}$ must either perform infinitely many $\mathtt{r}_\gamma$, either perform a finite number of $\mathtt{ic}_\gamma$. The automaton $\mathcal{B}$ will be forced to guess which of these cases is occuring, and to verify it. In case 1, this can be done by waiting a reset before acknowledging the next Büchi state. States $1'$ will carry the information that a reset has been seen. Moreover, the transitions leaving acknowledged Büchi states are changed to resets. In the case 2 (finite number of $\mathtt{ic}_\gamma$), the automaton must guess the moment after which no more $\mathtt{ic}_\gamma$ will be seen, we call the part after this phase $2'$. Büchi states will only occur in phase $2'$, and it does not change the cost of the play to replace ε by $\mathtt{r}$ in this phase.

Formally, $\mathcal{B} = \langle Q, \mathbb{A}, I, F, \Gamma, \Delta \rangle$, with

- $Q = Q_\mathcal{A} \times \{1, 1', 2, 2'\}^\Gamma$
- $I = I_\mathcal{A} \times \{1, 2\}^\Gamma$
- $F = F_\mathcal{A} \times \{1', 2'\}^\Gamma$
- $\Delta = \{((p, \boldsymbol{x}), a, g'(\boldsymbol{x}, \tau, b), (q, \boldsymbol{x}')) : (p, a, \tau, q) \in \Delta_\mathcal{A},$
  $\boldsymbol{x}' \in g(\boldsymbol{x}, \tau, b), b = B$ if $(p, \boldsymbol{x}) \in F$ and $\not{B}$ otherwise$\}$,
  where g and g' are defined below. Notice that $g(p, \boldsymbol{x}, \tau)$ is a set, so it is possible that several (or zero) transitions in $\mathcal{B}$ correspond to one in $\mathcal{A}$.

The aim of g is to update the $\boldsymbol{x}$ component, depending on the action τ. It suffices to define g for one particular γ, the general case is obtained from this by using this g on each component. We also define $g'(\boldsymbol{x}, \tau, b)$ for each component: it is equal to $\mathtt{r}$ if $\boldsymbol{x} \in \{1', 2'\}$ and $b = B$, and to τ otherwise. The b component (in $\{B, \not{B}\}$) is used to remember whether the current state is Büchi in the new automaton. We can now define, for all $b \in \{B, \not{B}\}$:

- $g(1, \mathtt{ic}, b) = g(p, 1, \varepsilon) = \{1\}$,
- $g(1, \mathtt{r}, b) = g(p, 1', \mathtt{r}) = \{1'\}$,
- $g(1', \mathtt{ic}, B) = g(1', \varepsilon, B) = \{1\}$,
- $g(1', \mathtt{ic}, \not{B}) = g(1', \varepsilon, \not{B}) = \{1'\}$
- $g(2, \tau, b) = \{2, 2'\}$ for all $\tau \in \{\mathtt{ic}, \varepsilon, \mathtt{r}\}$,
- $g(2', \varepsilon, b) = g(2', \mathtt{r}, b) = \{2'\}$
- $g(2', \mathtt{ic}, b) = \emptyset$.

It is easy to verify that $\mathcal{B}$ is in normal form. Notice that $\mathcal{B}$ has the same set of counters as $\mathcal{A}$, and $|Q| = |Q_\mathcal{A}| \times 4^{|\Gamma|}$.

Lemma 14. $[\![\mathcal{B}]\!] \approx [\![\mathcal{A}]\!]$

Proof. Let $u \in \mathbb{A}^\omega$, and $n \in \mathbb{N}$.

Assume $[\![\mathcal{B}]\!](u) \leq n$. Let ρ be a n-run of $\mathcal{B}$ over u. Each transition $((p, \boldsymbol{x}), a, \tau, (q, \boldsymbol{x}') \in \Delta$ corresponds to a transition $(p, a, \tau', q) \in \Delta_{\mathcal{A}}$. This means that we can define a run ρ' of $\mathcal{A}$ over u. Moreover, the action sequence is the same, except that for each counter, we added at most one reset in ρ between two resets in ρ', or we added some resets after the last increment. This means that $val_B(\rho') \leq 2 * val_B(\rho) = 2n$, hence $[\![\mathcal{A}]\!](u) \leq 2n$. This is true for all u so $[\![\mathcal{A}]\!] \leq 2[\![\mathcal{B}]\!]$.

Conversely, assume $[\![\mathcal{A}]\!](u) \leq n$, and let ρ be a n-run of $\mathcal{A}$ over u, starting in state $q_0 \in I$.

We want to associate a run ρ' of $\mathcal{B}$ to ρ.

First, we need to choose an initial state. For all counter γ, let $x_\gamma = 1$ if there are infinitely many $\mathtt{ic}_\gamma$ in ρ, and 2 otherwise. x_γ is called the type of γ. Notice that if $x_\gamma = 1$, then there are infinitely many $\mathtt{r}_\gamma$ in ρ, otherwise the value of ρ would not be finite. Let $\boldsymbol{x} = (x_{\gamma_1}, \ldots, x_{\gamma_k})$, and $q'_0 = (q_0, \boldsymbol{x})$.

For counters γ of type 1, we have no more choice, since $g(x, \tau, b)$ is always a singleton when $x \in \{1, 1'\}$, and the component $\{1, 1'\}$ and $\{2, 2'\}$ are disjoint. For counters γ of type 2, we still have to define when to switch to component $2'$. We can do this as soon as we enter the suffix with no more $\mathtt{ic}_\gamma$. This defines a run ρ' , which has value less than n, since the only change is the actions is the replacement of some actions by resets. We still have to show that ρ' is accepting. The Büchi states of $\mathcal{B}$ are defined as $F = F_{\mathcal{A}} \times \{1', 2'\}^\Gamma$, which means that all components must be equal to $1'$ or $2'$. Every γ of type 2 enters the $2'$ component at some point, so these counters are not a problem: they can only delay the appearance of the first Büchi state for a finite amount of time. We look at the behaviour of counters γ of type 1. Each time a $\mathtt{r}_\gamma$ is see, the component switches to $1'$. Then it waits a Büchi state of $\mathcal{B}$ to go back to 1. At any point, it suffices to wait for every component to switch to $1'$, and then to wait for the next Büchi state of $\mathcal{A}$, in order to witness a Büchi state of $\mathcal{B}$. Only then, all components will simultaneously go back to 1 (excepts for the one directly seeing a reset). We showed that ρ' is avvalid n-run of $\mathcal{B}$ over u, so $[\![\mathcal{B}]\!](u) \leq n$.

Finally, we get $[\![\mathcal{B}]\!] \leq [\![\mathcal{A}]\!] \leq 2[\![\mathcal{B}]\!]$, so $[\![\mathcal{B}]\!] \approx [\![\mathcal{A}]\!]$. □

E.2 The B-Büchi game for $\mathcal{B}$

We look here from the point of view of the opponent in the B-Büchi game corresponding to automaton $\mathcal{B}$, i.e. Player Max. His aim is to guarantee either a high counter value, or the failure of the Büchi condition.

In fact Player Max has no role at all in the transitions of $\mathcal{B}$, since all the decisions in a run of $\mathcal{B}$ are taken by Player Min who aims at a low value. However, we will be interested in Player Max proving that Player Min does not have a good strategy. This will be done by assigning ranks to positions in the game, satisfying certain constraints.

E.3 Playing with graphs

Let $\mathbb{C} = \mathbb{B}^\Gamma$ be the set of actions on counters of Γ.

Let $u = a_0 a_1 \cdots \in \mathbb{A}^\omega$. We define the $\mathbb{C}$-labelled DAG $G = (V, E)$ in order to describe all possible behaviours of $\mathcal{B}$ on u. We have $V = Q \times \mathbb{N}$ and $E \subseteq V \times \mathbb{C} \times V$ is the set $\{((p, l), \nu, (q, l+1)) : (p, a_l, \nu, q) \in \Delta\}$.

The rank definition is a generalization of the one in [16]. Let $K := 2|Q| + 1$. Let $n \in \mathbb{N}$. A n-path in G is a path of global value at most n. Let G' be a subgraph of G. We say that a vertex v in G' is *n-endangered* in G' iff there is no infinite n-path in G' starting in v. We say that v is *n-safe* in G' iff no vertex (q, l) of G' with $q \in F$ is reachable from v via a n-path.

For each $n \in \mathbb{N}$, we define the DAGs $(G^n_k)_{k \in \mathbb{N}}$ by

- $G^n_0 = G^n$,
- $G^n_{2i+1} = G^n_{2i} \setminus \{v : v \text{ is } n\text{-endangered in } G^n_{2i}\}$,
- $G^n_{2i+2} = G^n_{2i+1} \setminus \left\{v : v \text{ is } n\text{-safe in } G^n_{2i+1}\right\}$.

We simultaneously define for all $v \in V$, $\mathrm{rank}_n(v) = \left\{k : v \in G^n_k \setminus G^n_{k+1}\right\}$.

Intuitively, $\mathrm{rank}_n(v)$ evaluates how hard it is to be convinced that there is no Büchi n-run from v. For instance $rank_n(v)$ is 0 if there is no infinite path of value at most n starting in v.

Lemma 15. *For all $n < [\![\mathcal{B}]\!](u)$ and $i \geq 0$, there exists l_i such that for all $l \geq l_i$, there are at most $|Q| - i$ vertices of the form (q, l) in G^n_{2i}.*

Proof. We fix $n < [\![\mathcal{B}]\!](u)$, and prove the lemma by induction on i. The case $i = 0$ follows from the fact that $G^n_0 \subseteq G$, and G has at most $|Q|$ vertices of the form (q, l) for any l. We assume the lemma for i, and show it for $i + 1$.

We consider the graph G^n_{2i}. If it is finite, then G^n_{2i+1} is empty and G^n_{2i+2} is empty as well, so the lemma trivially holds for $i + 1$. Otherwise, G^n_{2i} is infinite. We show that there is a n-safe vertex in G^n_{2i+2}. Assume it is not the case. Since we removed all n-endangered vertices from G^n_{2i}, G^n_{2i+1} is infinite and without leaves. Let $q_0 \in I$, the additional assumption that no vertex is n-safe allows us to build a path π in G^n_{2i+1} starting in $(q_0, 0)$ with infinitely many F-states, connected by n-paths. To achieve this, we start in $(q_0, 0)$, and use the fact that it is not n-safe to reach (q_1, l'_1) via a n-path, with $q_1 \in F$. We can then use the fact (q_1, l'_1) is not a leaf, and choose a successor $(q'_1, l'_1 + 1)$. From this we can again find an n-path to reach (q_2, l'_2) with $q_2 \in F$, and so on...

But the automaton $\mathcal{B}$ is in normal form, so for each counter γ, all transitions leaving Büchi states perform a $\mathtt{r}_\gamma$. Since n-paths have value at most n, we get that the value of π is at most n.

This contradicts the fact that $n < [\![\mathcal{B}]\!](u)$, so we can conclude that there is a n-safe vertex $v = (q, l)$ in G^n_{2i+1}. We claim that we can take $l_{i+1} = l$.

Since v is in G^n_{2i+1}, it is not n-endangered in G^n_{2i}, there is an infinite n-path π starting in v in G^n_{2i}, which still exists in G^n_{2i+1}. Moreover the fact that v is n-safe in G^n_{2i+1} implies that all vertices in π are as well. This means that π is absent in G^n_{2i+2}, so the width of the G^n_{2i+2} after depth l is at most $|Q| - i - 1$, by induction hypothesis. □

Corollary 4. *For all $u \in \mathbb{A}^*$ and $n \in \mathbb{N}$, $[\![\mathcal{B}]\!](u) > n$ iff G^n_K is empty.*

Proof. By Lemma 15, if $[\![\mathcal{B}]\!](u) > n$, then there is $l_{|Q|}$ such that $G^n_{2|Q|}$ has 0 vertices of the form (q, l) for all $l \geq l_{|Q|}$. It means that all vertices are n-endangered in $G^n_{2|Q|}$, so $G_K = G_{2|Q|+1}$ is empty.

Conversely, if $[\![\mathcal{B}]\!](u) \leq n$, then there is an n-path π in G with infinitely many F-vertices on it. A straightforward induction shows that for all k, no vertex of π is n-endangered or n-safe in G^n_k, and so π is in all the $(G^n_k)_{k\geq 0}$. This suffices to show that G^n_K is not empty. □

E.4 Definition of weak automaton W

We can now describe the B-WAA W. It is the same as in [16], excepts that it copies counter actions from $\mathcal{B}$.

Formally, let $W := \langle Q_W, \mathbb{A}, Q_{in}, F_W, \Gamma, \delta\rangle$. We set $Q_W = Q \times [0, K-1]$, $Q_{in} = I \times \{K-1\}$ and $F_W = Q \times ([0, K-1] \cap 2\mathbb{N})$. Intuitively, when the automaton is in the state (q, i) at position l in the word, it means that the opponent guessed that $\mathrm{rank}_n(q, l) = i$, where n is the value it aims at. Initial states are an exception to this intuition, because $K-1$ is the upper bound for $\mathrm{rank}_n(q_0, 0)$, for $q_0 \in I$. Recall that $K = 2|Q| + 1$.

We finally define $\delta : Q_W \times \mathbb{A} \to \mathcal{B}^+(\mathbb{C} \times Q_W)$ by

$$\delta(\langle q, i\rangle, a) = \begin{cases} \bigvee_{(q,a,\nu,p)\in\Delta} \bigwedge_{0\leq j\leq i} (\nu, \langle p, j\rangle) & \text{if } q \notin F \text{ or } i \text{ even} \\ \mathit{true} & \text{if } q \in F \text{ and } i \text{ odd} \end{cases}$$

If we call the *color* of a state $col(q, i) = i$, it is easy to verify that the color is decreasing in any run of W. Then accepting runs are those which stabilize in even colors, and W is indeed a weak automaton, with partitions of the form $Q_i := Q \times \{i\}$.

E.5 Correctness of W

Player Or (or Min) tries to minimize the value of W, by choosing a low value run of $\mathcal{B}$. Player And (or Max) tries to maximize the value of W, either by forcing the run to be rejecting (inducing value ∞), or by reaching a high counter value.

Let $u \in \mathbb{A}^\omega$ and $n = [\![\mathcal{B}]\!](u) - 1$. By Corollary 4, G^n_K is empty, so rank_n is a function $V \to [0, K-1]$, and is decreasing along all paths of G. A play of W corresponds to Player Min choosing a path in G, and player Max labelling this path with colors. We can look at the strategy σ_n of player And consisting of labelling every vertex v of G by $\mathrm{rank}_n(v)$.

Let P be a play compatible with σ_n and π be the corresponding path in G. First of all, the automaton never reaches the transition *true*, because if $\mathrm{rank}_n(v)$ is odd, it means that v is n-safe in some G^n_{2i+1}, so in particular v cannot be an F-vertex. It means that σ_n never reaches a state $\langle q, i\rangle$ with i odd and $q \in F$.

If π stabilizes in an odd rank, it has value ∞. Assume π stabilizes in an even rank. It means that after some point, all vertices in π are n-endangered. Since π

is an infinite path in G, it must have value at least $n+1$, otherwise the vertices on it would not be n-endangered.

The strategy σ_n witnesses $[\![W]\!](u) \geq n+1$, so $[\![W]\!] \geq [\![\mathcal{B}]\!]$.

Conversely, if $n = [\![\mathcal{B}]\!](u)$, then we show that no strategy of Player Max can ensure a value at least $n+1$. Player Min can play in W an accepting $n-$run ρ of $\mathcal{B}$, witnessing infinitely many states from F. Recall that the counter actions of W are copied from $\mathcal{B}$, so the only way for Player And to ensure value at least $n+1$ is to make the run rejecting, i.e. stabilize in an odd partition i. However, since there are infinitely many F-states on ρ, the automaton will reach a state $\langle q, i\rangle$ with $q \in F$, and will go to *true*, so the value of the play will still be at most n.

We showed $[\![W]\!] \leq [\![\mathcal{B}]\!]$, so finally $[\![W]\!] = [\![\mathcal{B}]\!]$.

Remark that we get an equality of the functions computed by W and $\mathcal{B}$, and not just a cost function equivalence. Moreover, the number of states of W is quadratic in the number of states of $\mathcal{B}$, and the number of counters is preserved.

By equivalence of CMSO with B-NBA, and of WCMSO with B-WAA (Theorem 3), we get the following result:

Theorem 7. *Any cost function over infinite words definable in CMSO is also definable in WCMSO.*