Cost Functions Definable by Min/Max Automata[*]

Thomas Colcombet[1], Denis Kuperberg[2], Amaldev Manuel[3], and Szymon Toruńczyk[†][3]

1   CNRS & LIAFA, Université Paris Diderot, Paris 7
2   IRIT/ONERA, Toulouse
3   MIMUW, University of Warsaw

──── **Abstract** ────

Regular cost functions form a quantitative extension of regular languages that share the array of characterisations the latter possess. In this theory, functions are treated only up to preservation of boundedness on all subsets of the domain. In this work, we subject the well known *distance automata* (also called *min-automata*), and their dual *max-automata* to this framework, and obtain a number of effective characterisations in terms of logic, expressions and algebra.

1   Introduction

Regular languages enjoy multiple equivalent characterisations, in terms of regular expressions, automata, monoids, monadic second order (MSO) logic, etc. One of them is purely algebraic: a language $L \subseteq A^*$ is regular if and only if its two-sided Myhill-Nerode congruence on A^* has finite index. These characterisations have been refined further for many subclasses of regular languages. The archetypical example is the Schützenberger-McNaughton-Papert theorem. which equates the class of star-free languages (i.e. expressible by star-free regular expressions with complementation), the class of first-order definable languages, the class of languages accepted by *counter-free* automata, and the class of languages definable by aperiodic monoids (i.e. those that satisfy the equation $x^{n+1} = x^n$ for sufficiently large n). This gives an algebraic and effective characterisation of the class of star-free languages.

The theory of regular languages has been extended in many directions – to infinite words, trees, infinite trees, graphs, linear orders, traces, pictures, data words, nested words, timed words etc. With some effort, some of the above characterisations can be transferred, by finding the right notion of a "regular" language, and by finding algebraic and logical characterisations of certain subclasses of languages among the class of all the "regular" languages.

Whereas languages are qualitative objects, in this paper, we study characterisations of classes of quantitative objects. One of the classes that we study are cost functions defined by *distance automata*. A distance automaton $\mathcal{A}$ is like a nondeterministic finite automaton, where each transition additionally carries a *weight*, i.e., a natural number. The weight of a run is the sum of the weights of the transitions in the run. The distance automaton $\mathcal{A}$

───────

then associates to each input word w a value $[\![\mathcal{A}]\!](w) \in \mathbb{N} \cup \{\infty\}$, defined as the minimal weight of an accepting run over w, and ∞ if there is no accepting run. The central decision problem is the *limitedness problem* – does the function $[\![A]\!]$ have a finite range?

Distance automata were introduced by Hashiguchi in his solution of the star height problem, which he reduced to the limitedness problem for distance automata via a strenuous reduction. As distance automata try to minimize the value of a run, we call them *min-automata* in this paper. *Max-automata* are the dual model, for which the value of the word is the maximal weight of an accepted run. Min-automata and max-automata have appeared in various contexts (see related work below for more on this) under various names.

Distance automata were extended in subtly distinct ways to nested distance-desert automata [12], R-automata [1], B-automata [4, 6] by allowing several counters instead of just one, and allowing these counters to be reset. The limitedness problem is decidable for all these classes. In terms of cost functions all these extended models are equivalent.

Colcombet [6, 8] discovered that functions defined by B-automata enjoy a very rich theory of *regular cost functions*, extending the theory of regular languages. A *cost function* is an equivalence class of functions, where two functions $f, g : A^* \to \mathbb{N} \cup \{\infty\}$ are equivalent if they are bounded over precisely the same subsets of A^*. A *regular cost function* is the equivalence class of a function computed by a B-automaton. Colcombet gave equivalent characterisations of regular cost functions in terms of a quantitative extension of MSO, an extension of monoids called *stabilisation monoids*, Later, analogous characterisations were described, in terms of a quantitative extension of regular expressions, in terms of logics and regular expressions manipulating profinite words (a completion of the set of finite words in a certain metric) [17], and in terms of a finite index property [17, 14]. In [15] a characterisation in terms of a quantitative extension of FO on the Σ-tree is given.

Contributions

In this paper, we propose a Schützenberger-McNaughton-Papert style characterisation of the subclass of the class of regular cost functions, defined by distance automata – in terms of logic, regular expressions and algebra, i.e., by conditions satisfied by the syntactic stabilisation monoid. The last characterisation provides a machine-independent, purely algebraic description of the cost functions defined by distance automata – or min-automata. We also provide similar characterisations for the dual class of max-automata. Although their definition is simply obtained by replacing min by max, the statements and their proofs are quite different for both classes. Our characterisations are effective, i.e., given a B-automaton, it is decidable whether the cost function it defines is recognisable by a min- or max-automaton. Detailed proofs can be found in Appendix.

Related Work

Characterising special classes of regular cost functions in various formalisms has been done in [11, 14]. In [11], the class of temporal cost functions was defined and studied. These cost functions are only allowed to measure *consecutive* events, for instance the function counting the number of occurrences of a letter in an input word is not temporal. Equivalent characterisations of this class were given in terms of cost automata, regular languages, stabilisation monoids. Additionally, in [7], an equivalent fragment of cost MSO was given. In [14], the class of aperiodic cost functions was considered, as a generalisation of star-free languages. It was shown that this class of function can be equivalently characterised by definability via cost linear temporal logic, cost first order logic, or group-trivial stabilisation

monoids, generalising the Schützenberger-McNaughton-Papert theorem to cost functions.

In the papers [5] and [3], min- and max-automata were defined in a different way, as *deterministic* automata with many counters, over which any sequence of instructions could be performed in a single transition, where each instruction is either of the form $c := c + 1$ or $c := \max(d, e)$ (in the case of max-automata) or $c := \min(d, e)$ (in the case of min-automata), where c, d, e are counters. As these models were studied in relationship with logics over infinite words, rather than evaluating a finite word to a number, they were used as acceptors of infinite words. However, their finitary counterparts are equivalent to the models studied in this paper, up to cost function equivalence (see Proposition 2.4). Note that nondeterminism is exchanged for multiple counters with aggregation (min or max).

In the paper [2], Cost Register Automata are studied, and are parametrised by a set of operations. Those too are deterministic automata with many registers (i.e. counters). For the set of operations denoted $(\min, +c)$, one obtains a model equivalent to the min-automata of [5], and for the set of operations denoted $(\max, +c)$, one obtains a model equivalent to the max-automata [3].

Min-automata can be equivalently described as nondeterministic weighted automata over the semiring $(\mathbb{N} \cup \{\infty\}, \min, +)$, where min plays the role of addition and $+$ of multiplication. Similarly, max automata can be equivalently described as nondeterministic weighted automata over the semiring $(\mathbb{N} \cup \{\infty, \bot\}, \max, +)$, where max plays the role of addition and $+$ of multiplication (and $\bot$ is neutral with respect to max and absorbing with respect to $+$). Using this formalism, decidability results about precision of approximation of functions computed by min-automata are shown in [9]. Similar results on max-automata are presented in [10], together with an application to evaluation of time complexity of programs.

2 Preliminaries

In this section, we recall various models of cost automata, and the theory of regular cost functions. For more details, the reader should confer [6, 8, 11]. We write $\mathbb{N}_\infty$ for the set $\mathbb{N} \cup \{\infty\}$. We follow the convention that $\inf \emptyset = \min \emptyset = \infty$ and $\sup \emptyset = \max \emptyset = 0$.

2.1 Automata

We recall the notions of B- and S-automata, and relate them to min- and max-automata.

B-automata and S-automata. *B-* and *S-automata* are nondeterministic automata over finite words, which are moreover equipped with a finite set of *counters*. Each counter admits three basic operations: incrementation by one, denoted i, reset to zero, denoted r, and the idle operation, denoted ε. Initially, all counters are set to 0, and during the run each transition of the automaton performs one operation on each counter separately (formally, a transition is a tuple (p, a, o, q), where p is its source state, q is its target state, a is the label and $o = (o_c)_c$ is a vector of operations, one per each counter c). Additionally, we allow counters to be reset after the run terminates in an accepting state, depending on the state. If in a run ρ some transition resets a counter currently storing a value n, then we say that n is a *reset value* for the considered run ρ.

We now define the value of a word under a B-automaton; the definition for S-automata is dual and will be given later. Let $\mathcal{B}$ be a B-automaton and w be an input word. For a run ρ over the word w, define the *cost* of ρ as the maximum of the set of its reset values:

$$\mathrm{cost}(\rho) = \max\{n \in \mathbb{N} : n \text{ is a reset value for } \rho\}.$$

Recall that the maximum of the empty set is equal to 0. Finally, define $[\![\mathcal{B}]\!](w)$ as the minimal value of an accepting (initial-to-accepting state) run over w:

$$[\![\mathcal{B}]\!](w) \;=\; \min\{\mathrm{cost}(\rho) \mid \rho \text{ is an accepting run of } \mathcal{B} \text{ over } w\}.$$

Note that this value can be infinite, if there is no accepting run of $\mathcal{B}$ over w. In particular, if the set of counters is empty, then $[\![\mathcal{B}]\!](w) = 0$ if $\mathcal{B}$ has an accepting run over w, otherwise $[\![\mathcal{B}]\!](w) = \infty$. In this way, B-automata generalize finite automata.

If $\mathcal{A}$ is an S-automaton, the definitions are obtained by swapping min with max. In particular, the cost of the run is the minimum of the set of its reset values (recall that $\min \emptyset = \infty$) and $[\![\mathcal{A}]\!](w)$ is the maximal value of an accepting run over w. If $\mathcal{A}$ has no counters, then $[\![\mathcal{A}]\!](w) = \infty$ if $\mathcal{A}$ has an accepting run over w, otherwise $[\![\mathcal{A}]\!](w) = 0$.

For a B- or S-automaton $\mathcal{A}$, we call $[\![\mathcal{A}]\!]$ the function *computed* by $\mathcal{A}$.

▶ **Example 2.1.** We construct a B-automaton that computes the smallest length of a block of consecutive a's in an input word consisting of a's and b's. In other words, for an input word w of the form $a^{n_1} b a^{n_2} \dots b a^{n_k}$, the computed value is $f_{\min}(w) = \min_{j=1}^{k} n_j$. The automaton has one counter, and is depicted in Figure 1.
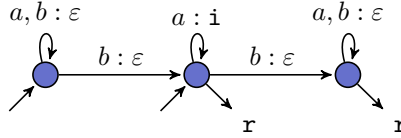
■ **Figure 1** A B-automaton, which is also a min-automaton. Edges with no source state mark initial states, and edges without a target state mark accepting states and may reset the counter.

▶ **Example 2.2.** Another example of a function computed by a B-automaton is the following. For a given word w over the alphabet $\{a, b, c\}$ of the form $w_1 c w_2 \dots c w_k$, where the words $w_1, \dots, w_k$ are over the alphabet $\{a, b\}$, define $f(w) = \max_{j=1}^{k} f_{\min}(w_j)$. The function f is computed by a B-automaton obtained from the automaton in Figure 1 by adding a c-labeled, resetting transition from every accepting state to every initial state.

▶ **Example 2.3.** The automaton in Figure 1 can be interpreted as an S-automaton which, for an input word w of the form $a^{n_1} b a^{n_2} \dots b a^{n_k}$, computes the value $f_{\max}(w) = \max_{j=1}^{k} n_j$.

Min-automata and max-automata. A *min-automaton* is a one-counter B-automaton $\mathcal{B}$, with only two operations allowed: $\mathtt{i}$ (increment) and ε (do nothing). In particular, resets are not allowed during the run. However, every counter is reset at the end of the run. Therefore the last counter value is a reset value. In other words, a min-automaton is a nondeterministic finite automaton in which every edge carries one of the two operations $\mathtt{i}$ or ε which manipulate the only counter. The cost of a run is the last value attained by the counter, and $[\![\mathcal{B}]\!](w)$ is the minimum of the costs of all accepting runs. This corresponds exactly to the definition of a *distance automaton* given in the introduction, with $\mathtt{i}$ corresponding to 1 and ε corresponding to 0 in the distance automaton. The automaton from Example 2.1 is a min-automaton.

Dually, a *max-automaton* $\mathcal{A}$ is a one-counter S-automaton, with only the two operations $\mathtt{i}$ and ε allowed, and where the automaton may, depending of the last state assumed, reset or not the counter at the end of the run. Therefore, the cost of a run is again the last value attained by the counter if it is reset, and $+\infty$ if it is not reset. The value of a word

$\llbracket \mathcal{A} \rrbracket (w)$ is the maximum of the costs of all accepting runs. Example 2.3 gives an example of a max-automaton.

As mentioned in the related work in the introduction, min/max-automata are related to other notions from the literature. We establish this connection in the proposition below, and later on in this paper, we will only talk about min- and max-automata.

A *weighted automaton* over a semiring $\mathcal{S}$ specifies a $n \times n$ matrix $h(a)$ over $\mathcal{S}$ for each letter a in the input alphabet, and two vectors I, F of length n over $\mathcal{S}$. The *value* associated to a word $a_1 \ldots a_l$ over the input alphabet is the product of the matrices $I^T \cdot h(a_1) \cdots h(a_l) \cdot F$, which is a 1×1 matrix, identified with an element of $\mathcal{S}$. In the proposition below, a value $\bot$ returned by a weighted automaton is interpreted as 0.

▶ Proposition 2.4 ([2, 5]). Min-automata are equivalent to distance automata, to non-deterministic weighted automata over the semiring $(\mathbb{N}_\infty, \min, +)$, and to deterministic automata with many registers storing elements of $\mathbb{N}_\infty$, allowing the binary min operation and unary incrementation operation.

Dually, max-automata are equivalent to nondeterministic weighted automata over the semiring $(\mathbb{N}_\infty \cup \{\bot\}, \max, +)$, and to deterministic automata with many registers storing elements of $\mathbb{N}_\infty$, allowing the binary max operation and unary incrementation operation.

The goal of this paper is to find effective algebraic characterisations of functions computable by min- and max-automata amongst all functions computable by B- and S-automata. These characterisations are up to equivalence of cost functions.

2.2 Theory of regular cost functions.

Throughout this paper, fix a finite input alphabet Σ. Given two functions $f, g : \Sigma^* \to \mathbb{N}_\infty$, we write $f \approx g$ if for all $X \subseteq \Sigma^*$, if g is bounded over X (meaning $\sup g|_X < \infty$), then f is bounded over X, and vice-versa. A *cost function* over the alphabet Σ is an equivalence class of $\approx$. Let $[f]$ denote the equivalence class of $f : \Sigma^* \to \mathbb{N}_\infty$. We will often identify a cost function with any of its representatives and say that f is a cost function, implicitly talking about $[f]$.

Regular cost functions. A cost function is *regular* if it is the cost function of the function computed by some B-automaton. For example, the (equivalence classes of the) functions described in Examples 2.1 and 2.2 are regular cost functions. It turns out [6] that B- and S-automata define equal classes of cost functions (see Theorem 2.15 below). Not every cost function is regular – indeed, there are uncountably many cost functions.

The reason we prefer to study cost functions computed by B-automata rather than the functions themselves is due to the following results, and to the fact that information about boundedness properties suffice in the contexts we will be interested in.

▶ **Theorem 2.5** (Krob [13]). *Given two min-automata $\mathcal{A}$ and $\mathcal{B}$, it is undecidable whether the functions $\llbracket \mathcal{A} \rrbracket$ and $\llbracket \mathcal{B} \rrbracket$ are equal.*

▶ **Theorem 2.6** (Colcombet [6]). *Given two B- or S-automata $\mathcal{A}$ and $\mathcal{B}$, it is decidable whether the functions $\llbracket \mathcal{A} \rrbracket$ and $\llbracket \mathcal{B} \rrbracket$ define the same cost function.*

If we take $\mathcal{B}$ in the theorem above to be such that $\llbracket \mathcal{B} \rrbracket (w) = 0$ for all words w, we see that in particular, it is decidable whether the function $\llbracket \mathcal{A} \rrbracket$ is bounded over all words. The limitedness problem for B-automata easily reduces to this problem.

Cost regular expressions.   Cost regular expressions are weighted extensions of classical regular expressions. They come in two forms, B-expressions and their dual S-expressions. A *B-expression* is given by the grammar,

$$E ::= a \in \Sigma \mid \emptyset \mid E \cdot E \mid E + E \mid E^{\leq n} \mid E^*,$$

where n is a variable (there is only one variable available). Note that by substituting $k \in \mathbb{N}$ for n in a cost regular expression E, denoted by $E[k \to n]$, one obtains a regular expression of finite words. Given a B-expression E the cost function computed by E is defined as:

$$[\![E]\!]\,(u) = \inf\{k \mid u \in E[k \to n]\}.$$

Similarly one defines S-expressions, with the difference that we are allowed to use $> n$ instead of $\leq n$, and at the end we take sup instead of inf.

Both kinds of expressions define exactly all regular cost functions. Indeed, it is not difficult to convert between B-expressions and B-automata (and between S-expressions and S-automata), similarly to the conversions between regular expressions and finite automata.

▶ **Example 2.7.** The cost function $f_{\max}$ is defined by the B-expression $(a^{\leq n}b)^*$ and the S-expression $(a^*b)^*a^{>n}(ba^*)^*$. Dually, the cost function $f_{\min}$ is defined by the B-expression $(a^*b)^*a^{\leq n}(ba^*)^*$ and the S-expression $(a^{>n}b)^*a^{>n}$.

Cost monadic second order logic.   We recall the basics of monadic second order logic (abbreviated as MSO) over words. The formulas use first order variables $x, y, z \dots$ that range over positions of the word and second order variables $X, Y, Z \dots$ that range over sets of positions of the word. MSO formulas are build using the atomic predicates $x \leq y$, $x \in X$, $a(x)$ (denoting that the label at position x is a, where $a \in \Sigma$), the connectives $\neg\varphi$, $\varphi \vee \psi$, $\varphi \to \psi$ and quantifiers $\exists x.\varphi$, $\forall x.\varphi$ (ranging over single elements) and $\exists X.\varphi, \forall X.\varphi$ (ranging over sets of elements). *Cost monadic second order logic* (cost-MSO) extends the logic by allowing formulas of the form $|X| \leq n$, where $|X|$ denotes the size of the set X and n is a fixed variable ranging over the natural numbers, with the restriction that they occur only positively (under even number of negations). Given a cost-MSO formula $\varphi(n)$ the cost function defined by $\varphi(n)$ is

$$[\![\varphi]\!]\,(u) = \inf\{n \mid u, n \models \varphi\}.$$

▶ **Example 2.8.** The cost function $f_{\min}$ is expressed by $\exists X.block_a(X) \wedge (|X| \leq n)$, where $ablock_a(X)$ is the first-order formula expressing that X is a maximal block of consecutive a's. Dually, $f_{\max}$ is expressed by the formula $\forall X.block_a(X) \to (|X| \leq n)$.

Stabilisation monoids.   Recall that if $\mathbf{M}$ is a finite monoid, $h : \Sigma \to \mathbf{M}$ is a mapping, and F is a subset of $\mathbf{M}$, then the triple $(\mathbf{M}, h, F)$ defines a regular language $L = \hat{h}^{-1}(F)$, where $\hat{h} : \Sigma^* \to \mathbf{M}$ is the unique homomorphism extending h. Conversely, any regular language L is induced by some triple $(\mathbf{M}, h, F)$ of this form. In this section we recall how this correspondence lifts to regular cost functions, by replacing finite monoids to stabilisation monoids. Let $E(\mathbf{M}) = \{e \in M | ee = e\}$ denote the set of idempotents of the monoid $\mathbf{M}$.

A *stabilisation monoid* $\mathbf{M} = \langle M, \cdot, \leq, \sharp \rangle$ is a finite monoid equipped with a partial order

$\leq$ and an operation $\sharp : E(\mathbf{M}) \to E(\mathbf{M})$ (called stabilisation), satisfying the following axioms:

$$a \cdot x \leq b \cdot y \qquad \text{for} \quad a \leq b, x \leq y \qquad (1)$$

$$e^{\sharp} \leq f^{\sharp} \qquad \text{for} \quad e \leq f, e, f \in E(\mathbf{M}) \qquad (2)$$

$$(a \cdot b)^{\sharp} = a \cdot (b \cdot a)^{\sharp} \cdot b \qquad \text{for} \quad a, b \in \mathbf{M} \text{ such that } a \cdot b, b \cdot a \in E(\mathbf{M}) \qquad (3)$$

$$e^{\sharp} \leq e \qquad \text{for} \quad e \in E(\mathbf{M}) \qquad (4)$$

$$(e^{\sharp})^{\sharp} = e^{\sharp} \qquad \text{for} \quad e \in E(\mathbf{M}) \qquad (5)$$

▶ **Example 2.9.** Any finite monoid can be seen as a stabilisation monoid, in which $e^{\sharp} = e$ for every idempotent, and where the order is trivial, i.e., $x \leq y$ iff $x = y$.

▶ **Example 2.10.** Every min-automaton $\mathcal{A}$ defines a finite transition stabilisation monoid, defined as follows. Let $\mathcal{T}$ denote the *tropical semiring* or the $(min, +)$-semiring, with domain $\mathbb{N}_{\infty}$ and min playing the role of addition, and $+$ of multiplication. We equip $\mathbb{N}_{\infty}$ with the usual topology, in which ∞ is the limit of every strictly increasing sequence.

First we define an infinite monoid, parametrised by a finite set of states Q. Let $\mathbf{M}_Q$ denote the set of $Q \times Q$ matrices with entries from $\mathcal{T}$. Matrices can be multiplied using the semiring operations of $\mathcal{T}$, and the set of matrices $\mathbf{M}_Q$ inherits the product topology from $\mathcal{T}$, i.e., a sequence $(M_n)_{n=1}^{\infty}$ of matrices is convergent if $M_n[p, q]$ is convergent in $\mathbb{N}_{\infty}$ for each $p, q \in Q$. Matrix multiplication is continuous, and moreover, one can show that for every matrix $M \in \mathbf{M}_Q$, when $n \to \infty$, the sequence $M^{n!}$ is convergent to a matrix denoted $M^{\sharp}$; moreover, the mapping $M \mapsto M^{\sharp}$ is continuous.

An automaton $\mathcal{A}$ with state space Q defines a mapping $h : \Sigma^* \to M$ which assigns to a word $w \in \Sigma$ the matrix $h(w)$ such that for two states p, q of $\mathcal{A}$, the value $h(w)[p, q]$ is the minimal cost of a run of $\mathcal{A}$ over w which starts in state p, ending in state q. The mapping h is a monoid homomorphism.

Define an equivalence relation $\sim_1$ on $\mathbf{M}_Q$ so that two matrices are equivalent iff they yield the same result when each finite, positive entry is replaced by 1. Define $\mathbf{M}_Q^1$ to be the set of $\sim_1$-equivalence classes. We identify an element of $\mathbf{M}_Q^1$ with its unique representative which is a matrix with entries in $\{0, 1, \infty\}$. It turns out [17] that the equivalence $\sim_1$ preserves multiplication and the operation $\sharp$. It follows that $\mathbf{M}_Q^1$ inherits the structure of a monoid, and also an operation $\sharp$; we restrict this operation only to idempotents (for a non-idempotent M, we can still recover $M^{\sharp}$ as $E^{\sharp}$, where E is the idempotent power of M).

One way to compute a product of two matrices $M, N \in \mathbf{M}_Q^1$ is to take a min-automaton $\mathcal{A}$ with states Q and with $h(a) = M$ and $h(b) = N$; then $M \cdot N$ is obtained by substituting 1 for every finite positive number in $h(ab)$. Similarly, if $M \in \mathbf{M}_Q^1$ is idempotent, then $M^{\sharp}$ is obtained by taking an automaton $\mathcal{A}$ with $h(a) = M$ and writing $M^{\sharp}[p, q] = 0$ if for arbitrarily large n, $h(a^n)[p, q] = 0$, otherwise $M^{\sharp}[p, q] = 1$ if for arbitrarily large n $h(a^n)[p, q]$ remains bounded, and finally, $M^{\sharp}[p, q] = \infty$ if $h(a^n)[p, q]$ converges to ∞ when $n \to \infty$. The computations can be performed purely mechanically (see Appendix). For example, for the automaton from Example 2.1, we compute $h(a) \cdot h(b)$ and $h(a)^{\sharp}$ in $\mathbf{M}_Q^1$:

$$\begin{bmatrix} 0 & \infty & \infty \\ \infty & 1 & \infty \\ \infty & \infty & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & \infty & \infty \\ 0 & \infty & \infty \\ \infty & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & \infty & \infty \\ 1 & \infty & \infty \\ \infty & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & \infty & \infty \\ \infty & 1 & \infty \\ \infty & \infty & 0 \end{bmatrix}^{\sharp} = \begin{bmatrix} 0 & \infty & \infty \\ \infty & \infty & \infty \\ \infty & \infty & 0 \end{bmatrix}$$

For $M, N \in \mathbf{M}_Q^1$, write $M \preceq N$ if there is a sequence of representatives of N which converges to a representative of M. In other words, M can be obtained from N by replacing some 1's by ∞'s. (The order $\preceq$ is the specialisation order associated to the quotient topology on $\mathbf{M}_Q^1$ inherited from $\mathbf{M}_Q$.)

It can be shown [17] that $\langle \mathbf{M}_Q^1, \cdot, \preceq, \sharp \rangle$ is a stabilisation monoid. The axiom (1) is a remnant of continuity of multiplication in $\mathbf{M}_Q$, (2) – of continuity of the operation $M \mapsto M^\sharp$, (3) – of associativity, (4) – of the fact that $M^\sharp$ is the limit of $M^{n!}$, and (5) – of an analogous property which holds in $\mathbf{M}_Q$.

Let $h^1 : \Sigma \to \mathbf{M}_Q^1$ be defined so that for a letter $a \in \Sigma$, the matrix $h^1(a)$ is equal to $h(a)$ (this is a matrix with entries in $\{0, 1, \infty\}$). The mapping h^1 encodes the transition structure of the automaton $\mathcal{A}$. Let I be the set of matrices $M \in \mathbf{M}_Q^1$ such that $M[p, q] = \infty$ for all initial states p and accepting states q of $\mathcal{A}$. The set I encodes the acceptance condition of $\mathcal{A}$. The cost function $\llbracket \mathcal{A} \rrbracket$ defined by $\mathcal{A}$ can be recovered from the triple $(\mathbf{M}_Q^1, h^1, I)$, as we will now describe.

▶ **Definition 2.11** (Computation tree)**.** An n-*computation tree* t is a finite rooted ordered unranked tree in which each node x has an associated *output* in $\mathbf{M}$ and is of one of four types:

Leaf x has no children and has an associated *label* $a \in \Sigma$, and the output of x is $h(a)$;

Binary node x has exactly two children and and the output of x is the product of the output of the first child and the output of the second child;

Idempotent node x has k children with $k \leq n$ and for some idempotent $e \in \mathbf{M}$, the output of each child is equal to e and the output of x is equal to e.

Stabilisation node x has k children with $k > n$ and for some idempotent $e \in \mathbf{M}$, the output of each child is equal to e and the output of x is equal to $e^\sharp$.

The *input* of the tree t is the word formed by the labels of the leaves of the tree, read left to right. The *output* of the tree t is the output of the root, and the neutral element of $\mathbf{M}$ if t is the empty tree.

An *ideal* in a stabilisation monoid $\mathbf{M}$ is a subset I which is downward-closed, i.e., $x \leq y$ and $y \in I$ imply $x \in I$. Let $h : \Sigma \to M$ be a mapping from a finite alphabet to a stabilisation monoid $\mathbf{M}$, and let I be an ideal in $\mathbf{M}$. The triple $(\mathbf{M}, h, I)$ induces a cost function, denoted $\llbracket \mathbf{M}, h, I \rrbracket$, and defined as follows. For a fixed height $k \in \mathbb{N}$, let

$$\llbracket \mathbf{M}, h, I \rrbracket_k (w) = \inf\{n \mid \text{there is a } n\text{-computation on } w \text{ with output in } \mathbf{M} \setminus I, \text{ height} \leq k\}.$$

It turns out [6] that the cost function $\llbracket \mathbf{M}, h, I \rrbracket_k$ does not depend on the choice of $k \geq 3|M|$. We define $\llbracket \mathbf{M}, h, I \rrbracket$ as $\llbracket \mathbf{M}, h, I \rrbracket_k$ for $k = 3|M|$.

▶ **Example 2.12.** Let $\mathbf{M}_{\max} = \langle \{1, a, b, a^\sharp\}, \cdot, \sharp, \leq \rangle$ be the stabilisation monoid with identity 1, zero $a^\sharp$, such that $ab = ba = b = bb = b^\sharp$, $aa = a$ and $a^\sharp \leq a$. The monoid $\mathbf{M}_{\max}$ defines the function $f_{\max}$ with ideal $\{a^\sharp\}$ and mapping $h(a) = a$ and $h(b) = b$.

▶ **Example 2.13.** Let $\mathbf{M}_{\min} = \langle \{1, a, a^\sharp, b, ba^\sharp, a^\sharp b, 0\}, \cdot, \sharp, \leq \rangle$ be the stabilisation monoid with identity 1, zero 0, product $a^\sharp ba^\sharp = a^\sharp$, $ba^\sharp b = b = ab = ba$, $bb = 0$, stabilisation $(ba^\sharp)^\sharp = ba^\sharp$, $(a^\sharp b)^\sharp = a^\sharp b$, and order $a^\sharp \leq a$, $a^\sharp b \leq b$, $ba^\sharp \leq b \leq 0$. The monoid $\mathbf{M}_{\min}$ defines the function $f_{\min}$ with ideal $\{a^\sharp\}$ and mapping $h(a) = a$ and $h(b) = b$.

▶ **Proposition 2.14.** For a min-automaton $\mathcal{A}$ with states Q, define $(\mathbf{M}_Q^1, h^1, I)$ as in Example 2.10. Then I is an ideal and the cost functions $\llbracket \mathbf{M}_Q^1, h^1, I \rrbracket$ and $\llbracket \mathcal{A} \rrbracket$ are equal.

Example 2.10 and Proposition 2.14 are a special case of a more general construction for B- and S-automata [17].

Closure properties.   Regular cost functions have several closure properties, described below. The fundamental cost function is the function count : $\{a, b\} \to \mathbb{N}_\infty$, defined by $\text{count}(u) = |u|_a$, the number of occurrences of letter a. A regular language $L \subseteq \Sigma^*$ is viewed as a

(regular) cost function mapping a word w to 0 if $w \in L$ and to ∞ if $w \notin L$. Note that since regular languages are closed under complements, exchanging 0 with ∞ in this definition would give the same class of cost functions.

Let $\mathcal{C}$ be a class of cost functions, possibly over different input alphabets. We define several closure properties for the class $\mathcal{C}$:

composition with morphisms if the cost function $f : \Sigma^* \to \mathbb{N}_\infty$ is in $\mathcal{C}$ and $\alpha : \Gamma^* \to \Sigma^*$ is a morphism, then the cost function $\alpha \circ f : \Gamma^* \to \mathbb{N}_\infty$ is in $\mathcal{C}$;

min if for any two cost functions $f, g : \Sigma^* \to \mathbb{N}_\infty$ in $\mathcal{C}$, the function $w \mapsto \min(f(w), g(w))$ also belongs to $\mathcal{C}$;

max defined dually, with max instead of min;

min with regular languages if for any cost function $f : \Sigma^* \to \mathbb{N}_\infty$ and regular language g viewed as a cost function (see above) the function $w \mapsto \min(f(w), g(w))$ also belongs to $\mathcal{C}$;

sup-projections if the cost function $f : \Sigma^* \to \mathbb{N}_\infty$ is in $\mathcal{C}$ and $\alpha : \Sigma^* \to \Gamma^*$ is a morphism, then the cost function $v \mapsto \sup\{f(w) : w \in \alpha^{-1}(v)\}$ is in $\mathcal{C}$;

inf-projections defined dually, with inf instead of sup.

The main theorem of regular cost functions. The theorem below shows that all the introduced notions give rise to the same class of cost functions, namely regular cost functions.

▶ **Theorem 2.15** (Colcombet [7])**.** *The following formalisms are effectively equivalent as recognisers of regular cost functions: B-automata, S-automata, B-expressions, S-expressions, cost MSO formulas, and stabilisation monoids. Moreover, the class of regular cost functions is the smallest class of cost functions which is closed under min, max, inf-projections, sup-projections, contains the function count, and all regular languages.*

We may now specify the goal of this paper more precisely. Among all regular cost functions, we characterise those which are of the form $[\![\mathcal{A}]\!]$ for a min- or max-automaton $[\![\mathcal{A}]\!]$. Our characterisations (Theorem 3.2 and Theorem 4.2) are in terms of cost regular expressions, fragments of cost MSO, and stabilisation monoids. Moreover, the characterisations are effective. In algebraic language theory, the usual way of providing effective characterisations is by means of certain algebraic conditions satisfied by the syntactic monoid. For this reason, we need to recall the notion of a syntactic stabilisation monoid.

Syntactic stabilisation monoid. A homomorphism of stabilisation monoids is a mapping $h : \mathbf{M} \to \mathbf{N}$ of stabilisation monoids which is monotone (i.e., $u \le v \implies h(u) \le h(v)$), preserves multiplication (i.e., $h(u \cdot v) = h(u) \cdot h(v)$) and stabilisation (i.e., $h(e^\sharp) = h(e)^\sharp$ for every idempotent $e \in \mathbf{M}$).

▶ **Theorem 2.16.** *[11] Let $f : \Sigma^* \to \mathbb{N}_\infty$ be a regular cost function. There is a unique (up to isomorphism) triple $(\mathbf{M}_f, h_f, I_f)$ recognising f with the following property. For any triple $(\mathbf{M}, h, I)$ recognising f, there is a unique surjective homomorphism $\phi : \mathbf{M} \to \mathbf{M}_f$ such that $h_f = \phi \circ h$ and $I = \phi^{-1}(I_f)$. $\mathbf{M}_f$ is called the syntactic stabilisation monoid of f.*

Moreover, for any triple $(\mathbf{M}, h, I)$ recognising f, the triple $(\mathbf{M}_f, h_f, I_f)$ can be computed in polynomial time from $(\mathbf{M}, h, I)$.

Idempotent power. It is a standard fact that every element a in a finite monoid has a unique idempotent power, denoted a^ω. It can be shown that $a^\omega = a^{n!}$ where n is the size of the monoid. We use the notation $a^{\omega\sharp}$ to denote the element $(a^\omega)^\sharp$.

3   Max-automata

In this section we characterise cost functions computed by max-automata. First we introduce the algebraic condition that characterises this class.

▶ **Definition 3.1** (Max-property). Let $\mathbf{M} = \langle M, \cdot, \sharp, \leq \rangle$ be a stabilisation monoid and $I \subseteq M$ be an ideal. The pair M, I has *Max-property* if for every $u, v, x, y, z \in M$,

1. if $xu^{\omega\sharp}yv^{\omega\sharp}z \in I$, then either $xu^{\omega\sharp}yv^{\omega}z \in I$, or $xu^{\omega}yv^{\omega\sharp}z \in I$, and
2. if $x(uy^{\omega\sharp}v)^{\omega\sharp}z \in I$, then either $x(uy^{\omega\sharp}v)^{\omega}z \in I$, or $x(uy^{\omega}v)^{\omega\sharp}z \in I$.

Now we present the main theorem of Section 3.

▶ **Theorem 3.2.** *The following are effectively equivalent for a regular cost function f:*

1. *f is accepted by a max-automaton,*
2. *f is definable by a formula of the form $\psi \wedge \forall X \ (\varphi(X) \to |X| \leq n)$ where ψ, φ are MSO formulas, i.e. they do not contain cost predicates,*
3. *f is in the smallest class (call it MAX) of cost functions that contains the function count and regular languages, and closed under min with regular languages, max, sup-projections, and composition with morphisms,*
4. *f is equivalent to $[\![\mathbf{M}, h, I]\!]$ for some mapping h from Σ to a stabilisation monoid $\mathbf{M}$ with ideal I having Max-property,*
5. *The syntactic stabilisation monoid and ideal of f has Max-property,*
6. *f is definable by an S-regular expression of the form $h + \sum_i e_i$ where h is a regular expression and each e_i is of the form $ef^{>n}g$ where e, f and g are regular expressions.*

▶ **Example 3.3.** The stabilisation monoid $\mathbf{M}_{\max}$ with ideal $\{a^\sharp\}$ recognising $f_{\max}$ has Max-property. But $\mathbf{M}_{\min}$ with ideal $I = \{a^\sharp\}$ recognising $f_{\min}$ violates the Max-property since $a^\sharp ba^\sharp = a^\sharp$ is in I, but neither of $a^\sharp ba = a^\sharp b, aba^\sharp = ba^\sharp$ belongs to I.

Since $\mathbf{M}_{\min}$ is the syntactic monoid of the function $f_{\min}$, the example above implies that $f_{\min}$ is not accepted a max-automata; in particular, max-automata are not closed under inf-projection. Similarly one verifies that cost functions computed by max automata are not closed under min (for instance the functions $u \in \{a, b\}^* \to |u|_a$ and $u \in \{a, b\}^* \to |u|_b$, as well as their max, is in MAX, but not their min).

Proof sketch. We sketch the implications: $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 1$.

$1 \to 2$ is by observing that there is a cost MSO formula encoding the runs of a 1-counter S-automata, of the form described in item 2.

To prove $2 \to 3$, it is enough to observe that all the subformulas (including the formula itself) of $\psi \wedge \forall X \ (\varphi \to |X| \leq n)$ (where ψ, φ are cost free) define cost functions in the class MAX. For this, we remark that the cost function count is in MAX, so it is also the case of $[\![|X| \leq n]\!]$, using the usual semantic for formulae with free variables, i.e. enriching the alphabet with a $\{0, 1\}$-component. Moreover MAX is closed under the operation min with regular languages, composition with morphisms (together they imply $[\![\neg\varphi \vee |X| \leq n]\!]$ is in MAX) and sup-projections (hence $[\![\forall X \ (\varphi(X) \to |X| \leq n)]\!]$ is in MAX). Finally closure under max implies that the formula defines a function in MAX.

To show $3 \to 4$, we show that the monoid constructions corresponding to the operations of max, min with a regular language, sup-projection and composition with a morphism preserve the Max-property, and also that the syntactic stabilisation monoids computing the function count as well as regular languages have the Max-property.

$4 \to 5$ follows from Theorem 2.16. Implication $5 \to 6$ is the hardest part of the theorem. By definition, the value of a word w given by the cost function $[\![\mathbf{M}, h, I]\!]$ is the maximum $n \in \mathbb{N}_\infty$ such that there is an n-computation tree of height $3|M|$ with input w and output in the ideal I. A way to attain this value using a cost regular expression is to write an expression that encodes all such computation trees by induction on the tree height; binary nodes translate to concatenation, idempotent nodes stand for Kleene star, and stabilisation nodes translate to the operator $>n$. But as such, this idea results in an expression with multiple occurrences of $>n$ (as there could be many stabilisations in the tree). This difficulty is circumvented by showing that it is sufficient to consider trees with only one stabilisation node. This is achieved by repeated use of the Max-property of the monoid $\mathbf{M}$ and ideal I.

For $6 \to 1$, note that the standard construction from cost regular expressions to S-automata (analogous to the translation from regular expressions to finite state automata) applied to the cost regular expressions of the form described in item 6, gives a max-automaton.

We note that all the transformations are effectively computable. ◀

Given a stabilisation monoid $\mathbf{M}$ and ideal I it is computable in polynomial time whether $\mathbf{M}, I$ has Max-property. Hence by Theorem 2.15 and Theorem 2.16 we obtain,

▶ **Theorem 3.4.** *It is decidable if a regular cost function satisfies a condition of Theorem 3.2.*

4 Min-automata

In this section we characterise cost functions definable by min-automata.

▶ **Definition 4.1** (Min-property)**.** We define the relation $\mathcal{R} \subseteq \mathbf{M} \times \mathbf{M}$ as the smallest reflexive relation satisfying the following implications: (1) if $x \mathcal{R} y$ and $a \mathcal{R} b$, then $(x \cdot a) \mathcal{R} (y \cdot b)$, and (2) if $x \mathcal{R} y$ then $x^\omega \mathcal{R} y^{\omega\sharp}$ and $x^{\omega\sharp} \mathcal{R} y^{\omega\sharp}$. A monoid $\mathbf{M}$ satisfies the *Min-property* if for all elements x, y, if $x^{\omega\sharp} \mathcal{R} y$, then $x^{\omega\sharp} = x^{\omega\sharp} y x^{\omega\sharp}$.

▶ **Theorem 4.2.** *The following are effectively equivalent for a regular cost function f:*

1. *f is accepted by a min-automaton,*
2. *f is definable by a formula of the form $\exists X \, (\varphi(X) \wedge |X| \leq n)$ where φ does not contain any cost predicates,*
3. *f belongs to the smallest class of cost functions containing count and regular languages that is closed under min, max and inf-projections,*
4. *f is recognised by a stabilisation monoid $\mathbf{M}$ with Min-property,*
5. *The syntactic stabilisation monoid of f has Min-property,*
6. *f is accepted by a B-regular expression E that is generated by the grammar*

$$E := F \mid E + E \mid E \cdot E \mid E^{\leq n} \qquad\qquad F := a \mid F + F \mid F \cdot F \mid F^*$$

   i.e. any subexpression of E of the form F^* is a regular expression (without $X^{\leq n}$),
7. *f is accepted by a B-automaton without reset.*

▶ **Example 4.3.** The monoid $\mathbf{M}_{\max}$ of $f_{\max}$ violates the Min-property since $b = b^\sharp \mathcal{R} a^\sharp$ (to see this, observe $a \mathcal{R} a^\sharp, b \mathcal{R} b$ and hence $ab = b \mathcal{R} a^\sharp = a^\sharp b$), but $b = b^\sharp \neq b^\sharp a^\sharp b^\sharp = ba^\sharp b = a^\sharp$. On the contrary, it can be verified that the monoid $\mathbf{M}_{\min}$ has Min-property.

Whereas max automata are not closed under min, min automata are closed under max. The class MIN falls only short of sup-projections, comparing to all regular cost functions.

The Min-property can be expressed in terms of identities, as follows. Consider the set T of terms involving variables from an infinite set of variables, a binary multiplication operation

and unary operations ω and $\omega\sharp$. Let $\mathcal{R}$ be the smallest binary relation on T that is reflexive and satisfies the implications 1 and 2 from Definition 4.1. Then, Min-property is expressed as the family of identities $x^{\omega\sharp} = x^{\omega\sharp} y x^{\omega\sharp}$, indexed by pairs of terms x, y such that $x^{\omega\sharp} \mathcal{R} y$.

Proof sketch. We sketch the implications: $1 \to 2 \to 3 \to 1 \to 4 \to 5 \to 1 \to 6 \to 7 \to 1$.

$1 \to 2$, $2 \to 3$, $6 \to 7$ are analogous to the corresponding cases in Theorem 3.2, and $3 \to 1$ is demonstrated by verifying that all functions and operations in item 3 can be carried out in the framework of min-automata.

$1 \to 4$ proceeds by studying the transition monoid of a min-automaton, described in Example 2.10, and checking that the equation of the Min-property is verified for any $x \preceq y$ (where $\preceq$ is the ordering used in the transition monoid). In particular, the equation is true for the relation $\mathcal{R}$, which is a subset of any stabilisation order.

$4 \to 5$ uses the fact that Min-property is equational, so is preserved by quotients.

$5 \to 1$ is obtained by performing substitutions in computation trees: any idempotent node that is the descendant of a stabilisation node can be transformed into a stabilisation node itself. We get a normal form for computation trees over monoids of this fragment, that we call frontier trees. We then design min-automata that witness the existence of frontier trees for any input word.

We show $1 \to 6$ by adapting the classical automaton-to-expression algorithm performing inductive state removal. Here, new transitions are labeled by regular expressions together with an action from $\{\varepsilon, \mathtt{i}\}$. When the classical algorithm produces a Kleene star, we do so if the looping transition is labeled ε; otherwise if it is $\mathtt{i}$, we replace the Kleene star by $\leq n$, and the resulting edge is again labeled $\mathtt{i}$. This way, a Kleene star cannot be produced on top of a subexpression containing a $\leq n$.

We show $6 \to 7$ by induction on the structure of the expression. We build an ad-hoc generalisation of the expression→ automaton algorithm, and show that the output B-automaton contains no reset.

Finally, $7 \to 1$ is obtained by observing that in the absence of resets, increments performed on k distinct counters can be performed on a single counter, by increasing the result at most k times. ◄

By a saturation algorithm, we can verify whether a given stabilisation monoid has Min-property in polynomial time. Hence by Theorem 2.15 and Theorem 2.16 we obtain:

▶ **Theorem 4.4.** *It is decidable if a regular cost function satisfies a condition of Theorem 4.2.*

5 Conclusion

We studied two dual classes of cost functions, defined by min-automata (also called distance automata), and max-automata. Both these classes have been studied in detail in other works. We showed that these classes enjoy many equivalent characterisations — such as restrictions of automata, logics, and expressions, and algebraic conditions. In both cases, the algebraic characterisation leads to decidability of membership in the class, in the spirit of Schützenberger's seminal work on star-free languages [16]. Combining with the finite-index characterisation of regular cost functions from [17], we obtain a purely algebraic characterisation of cost functions defined by min- or max-automata.

───── **References** ───

1   Parosh Aziz Abdulla, Pavel Krcál, and Wang Yi. R-automata. In *CONCUR 2008*, volume 5201, pages 67–81, 2008.
2   Rajeev Alur, Loris D'Antoni, Jyotirmoy V. Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *LICS 2013*, pages 13–22, 2013.
3   Mikolaj Bojanczyk. Weak MSO with the unbounding quantifier. *Theory Comput. Syst.*, 48(3):554–576, 2011.
4   Mikolaj Bojańczyk and Thomas Colcombet. Bounds in ω-regularity. In *LICS 06*, pages 285–296, 2006.
5   Mikolaj Bojanczyk and Szymon Torunczyk. Deterministic automata and extensions of weak mso. In *FSTTCS*, pages 73–84, 2009.
6   Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *Automata, Languages and Programming, Internatilonal Collogquium, ICALP 2009, Proceedings, Part II*, pages 139–150, 2009.
7   Thomas Colcombet. *Fonctions régulières de coût*. Habilitation thesis, Université Paris Diderot–Paris, 2013.
8   Thomas Colcombet. Regular cost functions, part I: logic and algebra over words. *Logical Methods in Computer Science*, 9(3), 2013.
9   Thomas Colcombet and Laure Daviaud. Approximate comparison of distance automata. In *STACS 2013*, volume 20 of *LIPIcs*, pages 574–585, 2013.
10 Thomas Colcombet, Laure Daviaud, and Florian Zuleger. Size-change abstraction and max-plus automata. In *MFCS 2014*, volume 8634 of *Lecture Notes in Computer Science*, pages 208–219, 2014.
11 Thomas Colcombet, Denis Kuperberg, and Sylvain Lombardy. Regular temporal cost functions. *Automata, Languages and Programming*, pages 563–574, 2010.
12 Daniel Kirsten. Distance desert automata and the star height problem. *ITA*, 39(3):455–509, 2005.
13 Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Internat. J. Algebra Comput.*, 4(3):405–425, 1994.
14 Denis Kuperberg. Linear temporal logic for regular cost functions. *Logical Methods in Computer Science*, 10(1), 2014.
15 Martin Lang, Christof Löding, and Amaldev Manuel. Definability and transformations for cost logics and automatic structures. In *MFCS 2014*, volume 8634 of *Lecture Notes in Computer Science*, pages 390–401. Springer, 2014.
16 M.P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190 – 194, 1965.
17 Szymon Torunczyk. *Languages of profinite words and the limitedness problem*. PhD thesis, PhD thesis, University of Warsaw, 2011.

A   Characterisation of Max-automata

In this section we prove Theorem 3.2.

1 → 2: Max-automata to Cost-MSO Formula

Assume that the cost function f is accepted by a max-automaton $\mathcal{A}$ with the transition relation $\Delta = \{\delta_1, \ldots, \delta_k\}$, set of initial states I, set of final states that are resetting F_1, and set of final final states that are not resetting F_0. As usual the idea is to encode the runs of the automaton $\mathcal{A}$ as a formula.

We write $\varphi(X)$ to mean that the second order variable X is free in the formula φ. Let $\psi(X_{\delta_1}, \ldots, X_{\delta_k}, X)$ be an MSO formula over words over the alphabet Σ that defines the following: On a given word w,

- taking X_{δ_i} to be the set of positions where the transition δ_i is applied, the monadic predicates $X_{\delta_1}, \ldots, X_{\delta_n}$ corresponds to a successful run of the automaton $\mathcal{A}$ on the word w and,
- the predicate X contains precisely the set of positions where a transition that increments the counter.

Let $\chi_1(X_{\delta_1}, \ldots, X_{\delta_k})$ (respectively $\chi_0(X_{\delta_1}, \ldots, X_{\delta_k})$) be the formula that expresses that the last transition of the correponding run ends in a final state that is (respectively *is not*) resetting. We claim that the formula φ where

$$\varphi = \varphi_1 \wedge \varphi_2$$
$$\varphi_0 = \exists X \exists X_{\delta_1} \cdots, \exists X_{\delta_k} \, (\psi \wedge \chi_0)$$
$$\varphi_1 = \forall X \, (\exists X_{\delta_1} \cdots, \exists X_{\delta_k} \, (\psi \wedge \chi_1) \rightarrow |X| \leq n)$$

defines the cost function f. Let $\mathcal{A}_0$ (respectively $\mathcal{A}_1$) be the restriction of $\mathcal{A}$ to the set of final states F_0 (respectively F_1). We prove that $[\![\mathcal{A}_i]\!] = [\![\varphi_i]\!]$ for $i = 0, 1$. Since $[\![\mathcal{A}]\!] = \max([\![\mathcal{A}_0]\!], [\![\mathcal{A}_1]\!])$ this immediately yields the claim:

$$[\![\varphi]\!] = \max([\![\varphi_0]\!], [\![\varphi_1]\!]) = \max([\![\mathcal{A}_0]\!], [\![\mathcal{A}_1]\!]) = [\![\mathcal{A}_1]\!] \ .$$

Since φ_0 is an MSO formula for all words w, $[\![\varphi_0]\!](w) \in \{0, \infty\}$.
In particular $[\![\varphi_0]\!](w) = \infty$,
   iff the formula φ_0 is true over the word w,
   iff there is a successful run of $\mathcal{A}_0$ over w,
   iff $[\![\mathcal{A}_0]\!](w) = \infty$, since $\mathcal{A}_0$ does not reset.

Next we prove $[\![\varphi_1]\!] = [\![\mathcal{A}_1]\!]$. Let w be a word. We have two cases depending on whether the formula $\exists X_{\delta_1} \cdots, \exists X_{\delta_k} \, (\psi \wedge \chi_1)$ is true over the word w or not. Assume the first case. Then,

$[\![\varphi_1]\!](w) = n \in \mathbb{N}$ (it can never be ∞),
   iff the least value for which the formula φ_1 is true over the word w is n,
   iff n is precisely the highest number of increments among all the successful runs of $\mathcal{A}_1$ on w
   iff $[\![\mathcal{A}_1]\!](w) = n$.

For the second case, when $\exists X_{\delta_1} \cdots, \exists X_{\delta_k} \, (\psi \wedge \chi_1)$ is false over the word w, then $[\![\varphi_1]\!]$ is true for value 0 as well as $[\![\mathcal{A}_1]\!](w) = 0$ since $\mathcal{A}_1$ does not have a successful run over w. This completes the proof.

2 → 3: From cost-MSO formula to the class MAX

For alphabet Σ, and φ a formula of cost-MSO, we write $\varphi(\Sigma)$ to mean that φ contain free second order variables denoting a labelling of positions by letters in Σ. Similarly we write $\varphi(X)$ to mean that the second order variable X is free in φ.

Assume we are given a formula

$$\chi = \psi \wedge \forall X \ (\varphi(\Sigma, X) \rightarrow |X| \leq n) \equiv \psi \wedge \forall X \ (\neg\varphi(\Sigma, X) \vee |X| \leq n)$$

where $\neg\psi(\Sigma, X)$ is cost-free. We want to show that the cost function $[\![\chi]\!] : \Sigma^* \rightarrow \mathbb{N}_\infty$ is in the class MAX. First observe that since $\neg\psi(\Sigma, X)$ is cost-free, $[\![\neg\psi]\!] : (\Sigma \times \{0, 1\})^* \rightarrow \mathbb{N}_\infty$ is a regular language, where the second component of the alphabet $\Sigma \times \{0, 1\}$ indicates the valuation of the second order variable X. Next, let $g : (\Sigma \times \{0, 1\}) \rightarrow \{a, b\}$ be the function defined as $g((\ell, 1)) = a$ and $g((\ell, 0)) = b$ for all $\ell \in \Sigma$. Extend g uniquely to a morphism from $(\Sigma \times \{0, 1\})^*$ to $\{a, b\}^*$. The cost function $g \circ \mathrm{count} : (\Sigma \times \{0, 1\})^* \rightarrow \mathbb{N}_\infty$ is in the class MAX since the function count is in MAX and MAX is closed under composition with morphisms. Furthermore $g \circ \mathrm{count}$ is precisely the cost function defined by the formula $|X| \leq n$ on words labelled by the alphabet $\Sigma \times \{0, 1\}$. Since MAX is closed under minimum with regular languages, the cost function $[\![\neg\varphi \vee |X| \leq n]\!] = \min([\![\neg\varphi]\!], [\![|X| \leq n]\!]) = \min([\![\neg\varphi]\!], g \circ \mathrm{count})$ is in MAX. Finally let h be the morphism from $\Sigma \times \{0, 1\} \rightarrow \Sigma$ defined canonically by extending the function $h((\ell, i)) = \ell$ for all $\ell \in \Sigma$ and $i \in \{0, 1\}$. By definition, $[\![\forall X \ (\neg\varphi(\Sigma, X) \vee |X| \leq n)]\!] = h_{\sup}(g \circ \mathrm{count})$ and since MAX is closed under sup-projections. Appealing to closure under maximum, $[\![\chi]\!] = [\![\psi \wedge \forall X \ (\neg\varphi(\Sigma, X) \vee |X| \leq n)]\!]$ is in MAX. This completes the proof.

3 → 4: From the class MAX to stabilisation monoids with Max-property

It suffices to show that every cost function in the class MAX is recognised by stabilisation monoid, ideal pair that has Max-property.

First we show that,

▶ **Lemma 1.1.** *The stabilisation monoid* $\mathbf{C} = \langle \{1, a, 0\}, \cdot, \sharp, \leq \rangle, I = \{0\}$ *satisfies the max property.*

Proof. By simple verification. For every $u, v, x, y, z \in \{1, a, 0\}$,

1. if $xu^{\omega\sharp}yv^{\omega\sharp}z = 0$, then one of $x, u^{\omega\sharp}, y, v^{\omega\sharp}, z$ is 0, and hence either $xu^{\omega\sharp}yv^\omega z = 0$, or $xu^\omega yv^{\omega\sharp}z = 0$, and
2. if $x(uy^{\omega\sharp}v)^{\omega\sharp}z = 0$, then one of $x, u, y^{\omega\sharp}, v, (uy^\omega v)^{\omega\sharp}, z$ is 0, and hence either $x(uy^{\omega\sharp}v)^\omega z = 0$, or $x(uy^\omega v)^{\omega\sharp}z = 0$.

◀

The above Lemma shows that the cost function count is recognised by a stabilisation monoid, ideal pair that satisfies the Max-property.

Since every regular language is recognised by an ordered monoid (also a stabilisation monoid with trivial stabilisation, i.e. identity) we get that,

▶ **Lemma 1.2.** *Every regular language is recognised by a stabilisation monoid, ideal pair that has Max-property.*

Next we introduce the notion of $\sharp$-expressions that is used in the remaining proofs.

▶ **Definition 1.3** (♯-expressions). Let $\mathbf{M} = \langle M, \cdot, \sharp, \leq \rangle$ be a stabilisation monoid. A ♯-expression E over a set $X \subseteq M$ is an expression composed of letters from X, products, ω-powers, and exponents with $\omega\sharp$. It naturally evaluates to an element of M, denoted $\mathrm{val}(E)$, and called the value of E.

Next we introduce some conventions we follow with ♯-expressions. For convenience, sometimes we drop the product operation while writing the ♯-expressions. For ♯-expressions E_1 and E_2, we write $E_1 = E_2$ (respectively $E_1 \leq E_2$) to mean $\mathrm{val}(E_1) = \mathrm{val}(E_2)$ (respectively $\mathrm{val}(E_1) \leq \mathrm{val}(E_2)$). Similarly when E is a ♯-expression and $Y \subseteq M$ is a set, then we write $E \in Y$ to mean $\mathrm{val}(E) \in Y$. To express the syntactical equivalence we write $E_1 =_{syn} E_2$. Clearly, if $E_1 =_{syn} E_2$, then $E_1 = E_2$.

Sometimes we manipulate ♯-expressions syntactically. For convenience we allow the use of the *empty ♯-expression*. It is straightforward to eliminate the empty ♯-expression from a ♯-expression. In particular, we don't distinguish between ♯-expressions that are unique upto removal of ϵ, i.e. $E_1 =_{syn} E_2$ if removing ϵ from E_1 and E_2 results in the same expression.

The *stabilisation rank* of a ♯-expression is the number of $\omega\sharp$ in it. A ♯-expression is called *strict* if it has stabilisation rank at least 1.

We write $\langle X \rangle^{\omega\sharp}$ for the set of values of ♯ expressions over X. Equivalently, it is the least set which contains X and is closed under product and stabilisation of idempotents. One also denotes $\langle X \rangle^{\omega\sharp+}$ for the set of values of strict ♯-expressions over X.

Next we want to show that if f is a cost function that is recognised by a stabilisation monoid $\mathbf{M}$, ideal I pair that has max property, then the inf-projections of f under morphisms as well as minimum of f with a regular language, both are recognised by stabilisation monoid, ideal pairs that have Max-property. Our strategy is as follows. We use the standard constructions on $\mathbf{M}, I$ (namely powerset and product) that give stabilisation monoid, ideal pairs that recognise the said functions and show that the resulting monoid, ideal pairs satisfy the Max-property. The details are postponed until we establish some lemmas that allow us to extend the Max-property, i.e. conversion of $\omega\sharp$ to ω, to all ♯-expressions.

▶ **Definition 1.4.** If E is a ♯-expression then $\overline{E}$ denotes the ♯-expression obtained from E by replacing all $\omega\sharp$ by ω. For example $\overline{(ab^{\omega\sharp}c)^{\omega\sharp}} = (ab^{\omega}c)^{\omega}$.

▶ **Lemma 1.5.** *Let $\mathbf{M} = \langle M, \cdot, \sharp, \leq \rangle$ be a stabilisation monoid and I be an ideal such that $\mathbf{M}, I$ has Max-property. If E is a ♯-expression over M, of positive stabilisation rank, that is in I, then there is a ♯-expression E' with stabilisation rank 1, that is also in I, such that E' is obtained from E by replacing all but one $\omega\sharp$ by ω.*

Proof. Fix an $n \in \mathbb{N}$ such that $a^n = a^{\omega}$ for all elements a of the monoid. Below, we use the usual descendant order on the nodes of the tree. Two nodes are incomparable if neither of them is a descendant of the other.

We treat the expression E as a labelled tree, called ♯-tree, whose nodes correspond to the subexpressions of E, and where each node is of one of the following types: a binary multiplication node (from now on, simply binary node) with label $\cdot$, a unary stabilisation node with label $\omega\sharp$, a unary idempotent node with label ω, or a leaf labelled by an element of the monoid. It is easy to inductively define the translations between ♯-expressions and ♯-trees.

An *expanded tree* is similar to the ♯-tree, except that an idempotent node with label ω has exactly n children. Here also we obtain a ♯-expression $E(x)$ corresponding to a node x inductively: a when it is a leaf labelled a, $E(y_1) \cdot E(y_2)$ when x is a binary node with children y_1 and y_2, $E(y_1) \cdots E(y_n)$ when x is an idempotent node with children $y_1, \ldots, y_n$,

$E(y)^{\omega\sharp}$ when x is a stabilisation node with child y. We define the value of a subtree to be the value of the $\sharp$-expression corresponding to the root of the subtree.

Given a $\sharp$-tree it is straightforward to convert it to an expanded tree, for every idempotent node in the $\sharp$-tree we duplicate the subtree rooted at its child n times and do it inductively. We observe that since n is an idempotent power of all the elements in the monoid, the $\sharp$ expressions corresponding to the $\sharp$-tree and its expanded tree have the same value.

Fix a height k. To every expanded tree of height k we associate a k-ary vector $(i_1, \ldots, i_k) \in \mathbb{N}^k$, called *level vector*, where i_j is the number of stabilisation nodes present at level j (we follow the convention that level 1 is the root). We put a preorder on the set of all expanded trees of height k according to the lexicographic order on their level vectors. In the following we perform the operation *sharp removal* on an exapanded tree, that replaces a stabilisation node with an idempotent node and duplicate the subtree rooted at its sole child n-times. We note that the resulting expanded tree is strictly smaller according to the preorder defined just now.

Our strategy is as follows. Given a $\sharp$-expression we construct the corresponding $\sharp$-tree, and then convert it to an expanded tree. The $\sharp$-expression corresponding to the expanded tree does not contain any ω-powers, hence we can inductively replace the $\omega\sharp$ occurring in the tree with ω-power. But the resulting tree is no longer an expanded tree, thus we are forced to expand the newly created idempotent node, by duplicating its children which increases the number of $\omega\sharp$ used in the tree. However, the resulting tree is smaller according to the preorder, which ensures termination of the procedure. At the end we are left with a tree with exactly one $\sharp$-node and we reconstruct the $\sharp$-expression ensuring that all but one $\omega\sharp$ is replaced by ω.

Next we describe the construction. Let E be the given $\sharp$-expression that has value in the ideal I. We construct the $\sharp$ corresponding to E and convert it to an expanded tree. With each node x of the expanded tree we associate the corresponding subexpression of E inherited via the $\sharp$-tree, called the *old expression of x*. We preserve the associated old expressions while we duplicate nodes.

During the construction, we preserve the invariant that the intermediate trees have value in the ideal. To begin with this is true since E has value in I.

At any stage during the construction, when the expanded tree has at least two stabilisation nodes, one of the following two cases occurs.

- There are two incomparable stabilisation nodes u, v, neither of which is a descendant of a stabilisation node.
- There are two stabilisation nodes u, v, such that u is not a descendant of a stabilisation node, and v is the descendant of u, and there is no other stabilisation node between u and v.

In each case we proceed by making one of the stabilisation nodes an idempotent node and duplicating its children.

First case. Let z denote the least common ancestor of u and v. Observe that the path π_u from the root to u contains no stabilisation nodes, likewise the path π_v from the root to v.

a to be the product of the values of the subtrees whose roots are left siblings of some node in π_u,

x to be the value of the subtree rooted at u,

b_1 to be the product of the values of the subtrees whose roots are right siblings of some node in π_u which is a descendant of z,

b_2 to be the product of the values of the subtrees whose roots are left siblings of some node in π_v which is a descendant of z,

y to be the value of the subtree rooted at v,

c to be the product of the values of the subtrees whose roots are right siblings of some node in π_v.

Note that value of the tree is equal to $ax^{\omega\sharp}by^{\omega\sharp}c$ and by assumption, this value belongs to I. We apply the first case in the definition of the Max-property, and derive that one of the values $ax^{\omega\sharp}by^{\omega}c$ or $ax^{\omega}by^{\omega\sharp}c$ belongs to I. Suppose that $ax^{\omega\sharp}by^{\omega}c \in I$, the other case being symmetric. Convert the node v in the expanded tree to an idempotent node and duplicate the subtrees rooted at its child n-times.

Observe that the value of the tree is $ax^{\omega\sharp}by^{\omega}c$ and belongs to I, and that the resulting tree has dropped strictly in the preorder.

Next we treat the second case.

Second case. In this case, there are two stabilisation nodes u, v, and u is not a descendant of any stabilisation node, whereas v is a descendant of u and has no other stabilisation ancestor.

Denote the path from the root to u by π, and the path from u to v by σ. Observe that both paths contain no stabilisation nodes, apart from their terminal vertices (respectively, u and v).

Define

x to be the product of the values of the subtrees whose roots are left siblings of some node in π,

a to be the product of the values of the subtrees whose roots are left siblings of some node in σ,

y to be the value of the subtree rooted at v,

b to be the product of the values of the subtrees whose roots are right siblings of some node in σ,

z to be the product of the values of the subtrees whose roots are right siblings of some node in π.

Note that value of the tree is $x(ay^{\omega\sharp}b)^{\omega\sharp}z$, and by assumption, this value belongs to I.

We apply the second case in the definition of the Max-property, and derive that one of the values $x(ay^{\omega}b)^{\omega\sharp}z$ or $x(ay^{\omega\sharp}b)^{\omega}z$ belongs to I. Suppose that $x(ay^{\omega}b)^{\omega\sharp}z \in I$. Convert the node v to an idempotent node and duplicate the subtrees rooted at its children n-times yielding an expanded tree whose value is $x(ay^{\omega}b)^{\omega\sharp}z \in I$. Furthermore, the expanded tree has one less stabilisation node in the level of v. Hence the resulting tree has fallen strictly in the preorder.

On the other hand, if $x(ay^{\omega\sharp}b)^{\omega}z \in I$, we proceed similarly, by converting the node u into an idempotent node and duplicate the subtrees rooted at its children n-times. The resulting tree, has value $x(ay^{\omega\sharp}b)^{\omega}z \in I$, and has dropped strictly in the preorder.

This completes the second case.

Proceeding inductively one obtains an expanded tree t with exactly one stabilisations node and that has value in the ideal I.

We next use this expanded tree to construct the $\sharp$-expression E'. Let us call the unique stabilisation node *good node* and let F be its old expression. We observe the following inductively:

1. The $\sharp$-expressions corresponding to the children of an idempotent node that is not an ancestor of the good node are syntactically equivalent and is equal to $\overline{H}$ (in fact syntactically equivalent to the expression obtained from $\overline{H}$ by inductively duplicating every subexpression of the form G^ω by G^n), where H is the old expression for them.
2. For the good node, its $\sharp$ expression is equal to $\overline{F}^{\omega\sharp}$.
3. For every idempotent node that is an ancestor of the good node, if G is the expression corresponding to a child which is not an ancestor of the good node, and H is the expression corresponding to the child that is an ancestor of the good node, then $G \leq H$.

We associate a *new expression* every node such that the new expression has value smaller than the value of its corresponding $\sharp$-expression. For every node that is not an ancestor of the good node the new expression is simply $\overline{H}$ where H is its old expression. For the good node, the new expression is $\overline{F}^{\omega\sharp}$. Next we inductively compute the new expression, by taking products of new expressions on binary nodes, and taking F^ω on idempotent nodes where F is the new expression of the unique child that is the ancestor of the good node. The new expression of the root is an expression that is syntactically equivalent to $\overline{E}$, belongs to the ideal, and has stabilisation rank 1. This completes the proof.

◄

As a corollary we obtain the following lemma:

▶ **Lemma 1.6.** *Let* $\mathbf{M} = \langle M, \cdot, \sharp, \leq \rangle$ *be a stabilisation monoid and I be an ideal in M such that* $\mathbf{M}, I$ *has Max-property. Let* E_1, E_2 *be $\sharp$-expressions over the monoid* $\mathbf{M}$*. Then for every* $x, y \in M$,

1. *If* $xE_1 E_2 y \in I$*, then either* $xE_1 \overline{E_2} y \in I$ *or* $x\overline{E_1} E_2 y \in I$*.*
2. *If* $xE_1^{\omega\sharp} y \in I$*, then either* $x\overline{E_1}^{\omega\sharp} y \in I$ *or* $xE_1^\omega y \in I$*.*

Next we continue with the proof:

Let f be a cost function over Σ recognised by the monoid $\mathbf{M} = \langle M, \cdot, \sharp, \leq \rangle$, ideal I and morphism $h : \Sigma^* \to \mathbf{M}$. Let $g : \Sigma^* \to \Sigma_1^*$ be a morphism. Then the sup-projection of f under g is given by

$$g_{\sup}(f)(w) = \sup\{f(u) \mid g(u) = w\}.$$

We next define a stabilisation monoid $\mathbf{M}^\uparrow$ that recognises the cost function $g_{\sup}(f)$.

Let X be a subset of M, we define $X\!\uparrow$ to be the upward closure of X, that is,

$$X\!\uparrow = \{y \mid \exists x \in X.\ x \leq y\}.$$

A *co-ideal* of M is a set which is upward-closed. Let $M^\uparrow$ be the set of all co-ideals of M equipped with the order

$$X \leq' Y \text{ iff } X \supseteq Y.$$

We equip $M^\uparrow$ with the product and stabilisation,

$$X \odot Y = \{x \cdot y \mid x \in X, y \in Y\}\!\uparrow$$

$$X^{\oplus} = \langle X \rangle^{\omega\sharp+}\!\uparrow$$

The sup-projection of f under g is recognised by $\mathbf{M}^\uparrow = \langle M^\uparrow, \odot, \oplus, \leq' \rangle, h_1, K$ where h_1 is the morphism from Σ_1^* to $\mathbf{M}^\uparrow$ defined as $h_1(w) = h(g^{-1}(w))\!\uparrow$ and $K \subseteq \mathbf{M}^\uparrow$ is the set $K = \{X \in \mathbf{M}^\uparrow \mid X \cap I \neq \emptyset\}$.

For $X, Y \subseteq M$, let $X \cdot Y$ denote the set $\{x \cdot y \mid x \in X, y \in Y\}$.

▶ **Lemma 1.7.** *For $X, Y \subseteq M$, $(X \cdot Y)\uparrow = (X\uparrow \cdot Y\uparrow)\uparrow$ and $\langle X \rangle^{\omega\sharp+}\uparrow = \langle X\uparrow \rangle^{\omega\sharp+}\uparrow$.*

Proof. By definition $(X \cdot Y)\uparrow \subseteq (X\uparrow \cdot Y\uparrow)\uparrow$ and $\langle X \rangle^{\omega\sharp+}\uparrow \subseteq \langle X\uparrow \rangle^{\omega\sharp+}\uparrow$. We show the other direction.

Let $z_1 \in (X\uparrow \cdot Y\uparrow)\uparrow$. There exist $z \in (X\uparrow \cdot Y\uparrow)$, $x_1 \in X\uparrow$, $y_1 \in Y\uparrow$, $x \in X$ and $y \in Y$ such that $z_1 \geq z$, $z = x_1 \cdot y_1$, $x_1 \geq x$ and $y_1 \geq y$. Then $z_1 \geq z = x_1 \cdot y_1 \geq x \cdot y$. Hence $z_1 \in (X \cdot Y)\uparrow$.

Similarly let $x \in \langle X\uparrow \rangle^{\omega\sharp+}\uparrow$. Then there exist a $\sharp$-expression $E \in \langle X\uparrow \rangle^{\omega\sharp+}\uparrow$ such that $x \geq E$. We prove by structural induction that $(\star)$ for every $E \in \langle X\uparrow \rangle^{\omega\sharp+}$ there is a $\sharp$-expression $E_1 \in \langle X \rangle^{\omega\sharp+}$ such that $E \geq E_1$. For the base case when $a \in X\uparrow$ choose $a_1 \in X$ such that $a \geq a_1$. Let $E_1 \leq E$ and $F_1 \leq F$ be expressions given by the induction hypothesis for expressions E and F. Let $G =_{syn} E \cdot F$ then $G_1 =_{syn} E_1 \cdot F_1 \leq E \cdot F \leq G$. Similarly when $G = E^{\omega\sharp}$ observe that $G_1 = E_1^{\omega\sharp} \leq E^{\omega\sharp} \leq G$. Therefore by $(\star)$ there is a $\sharp$-expression $E_1 \in \langle X \rangle^{\omega\sharp+}$ such that $E_1 \leq E$ and hence $E \in \langle X \rangle^{\omega\sharp+}\uparrow$.

◀

▶ **Lemma 1.8.** *Assume E is a $\sharp$-expression over the stabilisation monoid $\mathbf{M}^{\uparrow}$. Let E_1 be the expression obtained from E by replacing each subexpression $F \odot G$ by $F \cdot G$ and $F^{\omega\text{⊕}}$ by $\langle F \rangle^{\omega\sharp+}$. Then $E = E_1\uparrow$.*

Proof. By induction on the structure of E using Lemma 1.7. When E is simply an element of $M^{\uparrow}$, then by definition it is a co-ideal and the claim follows. When E is of the form $F \odot G$, let F_1 and G_1 be the expressions given by the induction hypothesis. Then, by Lemma 1.7, $E = (F \cdot G)\uparrow = (F_1\uparrow \cdot G_1\uparrow)\uparrow = (F_1 \cdot G_1)\uparrow$. When E is of the form $F^{\omega\text{⊕}}$ and F_1 is the expression corresponding to F given by induction hypothesis, Again by the Lemma, $F^{\omega\text{⊕}} = \langle F \rangle^{\omega\sharp+}\uparrow = \langle F_1\uparrow \rangle^{\omega\sharp+}\uparrow = \langle F_1 \rangle^{\omega\sharp+}\uparrow$. This concludes the induction. ◀

▶ **Lemma 1.9.** *If M, I has Max-property then $\mathbf{M}^{\uparrow}, K$ has Max-property as well.*

Proof. Proof is by contradiction. Assume M, I has Max-property while $\mathbf{M}^{\uparrow}, K$ does not. We have two cases.

For the first case, let $X, U, Y, V, Z \in \mathbf{M}^{\uparrow}$ be such that $X \odot U^{\omega\text{⊕}} \odot Y \odot V^{\omega} \odot Z \notin K$ and $X \odot U^{\omega} \odot Y \odot V^{\omega\text{⊕}} \odot Z \notin K$ but $X \odot U^{\omega\text{⊕}} \odot Y \odot V^{\omega\text{⊕}} \odot Z \in K$ (1). Using the Lemma 1.8 we get that $(X\langle U^{\omega}\rangle^{\omega\sharp+}YV^{\omega}Z)\uparrow \cap I = \emptyset$ and $(XU^{\omega}Y\langle V^{\omega}\rangle^{\omega\sharp+}Z)\uparrow \cap I = \emptyset$ but $(X\langle U^{\omega}\rangle^{\omega\sharp+}Y\langle V^{\omega}\rangle^{\omega\sharp+}Z)\uparrow \cap I \neq \emptyset$. Let S be the set of sharp expressions $X\langle U^{\omega}\rangle^{\omega\sharp+}Y\langle V^{\omega}\rangle^{\omega\sharp+}Z$. Since I is an ideal it follows that $S \cap I \neq \emptyset$. Let $x \cdot E_1 \cdot y \cdot E_2 \cdot z \in S \cap I$ where $x \in X$, $y \in Y$, $z \in Z$, and E_1 and E_2 are strict sharp expressions over U^{ω} and V^{ω} respectively. Since M, I has Max-property, by Lemma 1.5, one of of the expressions $x \cdot \overline{E_1} \cdot y \cdot E_2 \cdot z$, $x \cdot E_1 \cdot y \cdot \overline{E_2} \cdot z$ is in the Ideal. Assume it is the first one (the other case is analogous). Then, since U^{ω} an idempotent, $\overline{E_1} \in U^{\omega}$. It follows that $x \cdot \overline{E_1} \cdot y \cdot E_2 \cdot z$ is in $XU^{\omega}Y\langle V^{\omega}\rangle^{\omega\sharp+}Z$ as well as in the ideal I. This contradicts the assumption that $\left(XU^{\omega}Y\langle V^{\omega}\rangle^{\omega\sharp+}Z\right)\uparrow \cap I = \emptyset$.

For the second case, let $X, U, Y, V, Z \in \mathbf{M}^{\uparrow}$ be such that $X \odot (U \odot Y^{\omega\text{⊕}} \odot V)^{\omega\text{⊕}} \odot Z \notin K$ and $X \odot (U \odot Y^{\omega} \odot V)^{\omega\text{⊕}} \odot Z \notin K$ but $X \odot (U \odot Y^{\omega\text{⊕}} \odot V)^{\omega} \odot Y \in K$. Applying Lemma 1.8, we obtain that

$$\left(X \cdot \langle (U \cdot \langle Y^{\omega}\rangle^{\omega\sharp+} \cdot V)^{\omega}\rangle^{\omega\sharp+} \cdot Z\right)\uparrow \cap I \neq \emptyset, \tag{6}$$

$$\left(X \cdot \langle (U \cdot Y^{\omega} \cdot V)^{\omega}\rangle^{\omega\sharp+} \cdot Z\right)\uparrow \cap I = \emptyset, \tag{7}$$

$$\left(X \cdot (U \cdot \langle Y^{\omega}\rangle^{\omega\sharp+} \cdot V)^{\omega} \cdot Z\right)\uparrow \cap I = \emptyset . \tag{8}$$

Let S be the set of sharp expressions $\left(X \cdot \langle (U \cdot \langle Y^{\omega}\rangle^{\omega\sharp+} \cdot V)^{\omega}\rangle^{\omega\sharp+} \cdot Z\right)$. Since I is an ideal, from 6, we deduce that $S \cap I \neq \emptyset$. Hence there exists a sharp expression $x \cdot F \cdot y \in S \cap I$

where F is a sharp expression over the set of sharp expressions $T = (U \cdot \langle Y^\omega \rangle^{\omega\sharp+} \cdot V)^\omega$. By Lemma 1.5 there is an F' with stabilisation rank at most 1, obtained by replacing all but one $\omega\sharp$ in F by ω, such that $x \cdot F' \cdot y \in S \cap I$. Next we examine the F'. If F' is composed of elements from the set $(U \cdot Y^\omega \cdot V)^\omega$ then clearly $x \cdot F' \cdot y \in \left(X \cdot \langle (U \cdot Y^\omega \cdot V)^\omega \rangle^{\omega\sharp+} \cdot Z \right)\!\uparrow \cap I$ which contradicts the Assumption 7. Otherwise F' contains an element from the set T. Since F' has stabilisation rank at most 1, all other elements are from the set $(U \cdot Y^\omega \cdot V)^\omega$. Notice that T is an idempotent, and $(U \cdot Y^\omega \cdot V)^\omega \subseteq T\!\uparrow$ since Y^ω is an idempotent. Therefore $F' \in T\!\uparrow$, which further implies that $x \cdot F' \cdot y \in X \cdot T \cdot Z\!\uparrow \cap I$. This is in contradiction with the Assumption 8. This completes the proof of the second case.

◀

From the above lemma we get that:

▶ **Lemma 1.10.** *If a cost function is recognised by a stabilisation monoid, ideal pair that has Max-property then its sup-projections are also recognised by a a stabilisation monoid, ideal pair that has Max-property.*

Now we turn our attention to the closure under min with regular languages. First we introduce the product of two stabilisation monoids.

Product of two stabilisation monoids $\mathbf{M}_1 = \langle M_1, \cdot_1, \sharp_1, \leq_1 \rangle$ and $\mathbf{M}_2 = \langle M_2, \cdot_2, \sharp_2, \leq_2 \rangle$ is the stabilisation monoid $\mathbf{M}_1 \times \mathbf{M}_2 = \langle M_1 \times M_2, \cdot, \sharp, \leq \rangle$ where the product and stabilisation are defined as, for all $x_1, x_2 \in M_1$, $e \in E(M_1)$, $y_1, y_2 \in M_2$, $f \in E(M_2)$,

$$(x_1, y_1) \cdot (x_2, y_2) = (x_1 \cdot_1 x_2, y_1 \cdot_2 y_2) \, ,$$
$$(e, f)^\sharp = (e^{\sharp_1}, f^{\sharp_2}) \, ,$$
$$(x_1, y_1) \leq (x_2, y_2) \text{ iff } (x_1 \leq_1 x_2, y_1 \leq_2 y_2) \, .$$

Naturally the operations of min and and max of two cost functions can be computed by the product of stabilisation monoids recognising them.

▶ **Lemma 1.11.** *Let $\mathbf{M}_1 = \langle M_1, \cdot_1, \sharp_1, \leq_1 \rangle, I_1 \subseteq M_1$ be a stabilisation monoid and Ideal that satisfies the Max-property. Let $\mathbf{M}_2 = \langle M_2, \cdot_2, \sharp_2, \leq_2 \rangle, I_2 \subseteq M_2$ be stabilisation monoid recognising a regular language (i.e., $\sharp_2(e) = e$ for all $e \in E(M_2)$). Then $\mathbf{M}_1 \times \mathbf{M}_2 = \langle M_1 \times M_2, \cdot, \sharp, \leq \rangle, I_1 \times I_2$ satisfies the Max-property.*

Proof. Proof is by contradiction. For convenience we omit explicitly mentioning the product operations.

Assume $M_1 \times M_2, I_1 \times I_2$ does not satisfy the Max-property. We have two cases.

Let $(x_1, x_2), (u_1, u_2), (y_1, y_2), (v_1, v_2), (z_1, z_2) \in M_1 \times M_2$ be such that

$$(x_1, x_2)(u_1, u_2)^{\omega\sharp}(y_1, y_2)(v_1, v_2)^{\omega\sharp}(z_1, z_2) \in I_1 \times I_2 \tag{9}$$
$$(x_1, x_2)(u_1, u_2)^{\omega}(y_1, y_2)(v_1, v_2)^{\omega\sharp}(z_1, z_2) \notin I_1 \times I_2 \tag{10}$$
$$(x_1, x_2)(u_1, u_2)^{\omega\sharp}(y_1, y_2)(v_1, v_2)^{\omega}(z_1, z_2) \notin I_1 \times I_2 \tag{11}$$

From Equation 9 we get that $x_1 u_1^{\omega\sharp_1} y_1 v_1^{\omega\sharp_1} z_1 \in I_1$ and $x_2 u_2^{\omega\sharp_2} y_2 v_2^{\omega\sharp_2} z_2 \in I_2$. Since $\sharp_2(e) = e$ for all $e \in E(M_2)$ we obtain $x_2 u_2^\omega y_2 v_2^{\omega\sharp_2} z_2 = x_2 u_2^{\omega\sharp_2} y_2 v_2^\omega z_2 \in I_2$ (⋆). Since $\mathbf{M}_1, I_1$ has Max-property, by Lemma 1.5 we get that either $x_1 u_1^\omega y_1 v_1^{\omega\sharp_1} z_1 \in I_1$ or $x_1 u_1^{\omega\sharp_1} y_1 v_1^\omega z_1 \in I_1$. Assume $x_1 u_1^\omega y_1 v_1^{\omega\sharp_1} z_1 \in I_1$ (the other case is analogous). Then (⋆) implies that $(x_1, x_2)(u_1, u_2)^\omega(y_1, y_2)(v_1, v_2)^{\omega\sharp}(z_1, z_2) \notin I_1 \times I_2$ which contradicts the Assumption 10. This completes the proof of the first case.

Let $(x_1, x_2), (u_1, u_2), (y_1, y_2), (v_1, v_2), (z_1, z_2) \in M_1 \times M_2$ be such that

$$(x_1, x_2)((u_1, u_2)(y_1, y_2)^{\omega \sharp}(v_1, v_2))^{\omega \sharp}(z_1, z_2) \in I_1 \times I_2 \tag{12}$$

$$(x_1, x_2)((u_1, u_2)(y_1, y_2)^{\omega}(v_1, v_2))^{\omega \sharp}(z_1, z_2) \notin I_1 \times I_2 \tag{13}$$

$$(x_1, x_2)((u_1, u_2)(y_1, y_2)^{\omega \sharp}(v_1, v_2))^{\omega}(z_1, z_2) \notin I_1 \times I_2 \tag{14}$$

From 9 we obtain that $x_1(u_1 y_1^{\omega \sharp_1} v_1)^{\omega \sharp_1} z_1 \in I_1$ and $x_2(u_2 y_2^{\omega \sharp_2} v_2)^{\omega \sharp_2} z_2 \in I_2$. Since $\mathbf{M}_2, I_2$ is regular, $x_2(u_2 y_2^{\omega} v_2)^{\omega \sharp_2} z_2 = x_2(u_2 y_2^{\omega \sharp_2} v_2)^{\omega} y_2 \in I_2$ (†). Since $\mathbf{M}_1, I_1$ has Max-property, by Lemma 1.5 we get that $x_1(u_1 y_1^{\omega} v_1)^{\omega \sharp_1} z_1 \in I_1$ or $x_1(u_1 y_1^{\omega \sharp_1} v_1)^{\omega} z_1 \in I_1$. Assume $x_1(u_1 y_1^{\omega} v_1)^{\omega \sharp_1} z_1 \in I_1$; the other case is analogous. Therefore from (†) we get that $(x_1, x_2)((u_1, u_2)(y_1, y_2)^{\omega}(v_1, v_2))^{\omega \sharp}(z_1, z_2) \in I_1 \times I_2$ which contradicts the Assumption 13. This concludes the proof of the second case.

◀

The previous lemma implies that:

▶ **Lemma 1.12.** *If a cost function is recognised by a stabilisation monoid, ideal pair that has Max-property then its minimum with a regular language is also recognised by a a stabilisation monoid, ideal pair that has Max-property.*

Lastly we note that, if $f : \Sigma^* \to \mathbb{N}_{\infty}$ is a cost function recognised by the stabilisation monoid, ideal pair $\mathbf{M}, I$ and morphism h, and $g : \Sigma_1^* \to \Sigma$ then the cost function $g \circ f : \Sigma_1^* \to \mathbb{N}_{\infty}$ is recognised by $\mathbf{M}, I$ and morphism $g \circ h$. Therefore we obtain:

▶ **Lemma 1.13.** *If a cost function is recognised by a stabilisation monoid, ideal pair that has Max-property then its composition under a morphism is also recognised by a a stabilisation monoid, ideal pair that has Max-property.*

Next we prove closure under max.

▶ **Lemma 1.14.** *Let* $\mathbf{M}_1 = \langle M_1, \cdot_1, \sharp_1, \leq_1 \rangle, I_1 \subseteq M_1$, *and* $\mathbf{M}_2 = \langle M_2, \cdot_2, \sharp_2, \leq_2 \rangle, I_2 \subseteq M_2$ *be stabilisation monoid and ideal pairs that satisfy the Max-property. Then* $\mathbf{M}_1 \times \mathbf{M}_2 = \langle M_1 \times M_2, \cdot, \sharp, \leq \rangle, (M_1 \times I_2) \cup (I_1 \times M_2)$ *satisfies the Max-property.*

Proof. We verify both items of the Max-property on the monoid $\mathbf{M}_1 \times \mathbf{M}_2$ and ideal $(M_1 \times I_2) \cup (I_1 \times M_2)$.

Let $(x_1, x_2), (u_1, u_2), (y_1, y_2), (v_1, v_2), (z_1, z_2) \in M_1 \times M_2$ be such that

$$(M_1 \times I_2) \cup (I_1 \times M_2) \ni (x_1, x_2)(u_1, u_2)^{\omega \sharp}(y_1, y_2)(v_1, v_2)^{\omega \sharp}(z_1, z_2)$$
$$= (x_1 u_1^{\omega \sharp_1} y_1 v_1^{\omega \sharp_1} z_1, x_2 u_2^{\omega \sharp_2} y_2 v_2^{\omega \sharp_2} z_2)$$

If $(x_1 u_1^{\omega \sharp_1} y_1 v_1^{\omega \sharp_1} z_1, x_2 u_2^{\omega \sharp_2} y_2 v_2^{\omega \sharp_2} z_2) \in M_1 \times I_2$ (the other case being similar), then by using Max-property of $\mathbf{M}_2, I_2$, either $(x_1 u_1^{\omega \sharp_1} y_1 v_1^{\omega \sharp_1} z_1, x_2 u_2^{\omega_2} y_2 v_2^{\omega \sharp_2} z_2) \in M_1 \times I_2$ (that implies

$$(x_1 u_1^{\omega} y_1 v_1^{\omega \sharp_1} z_1, x_2 u_2^{\omega} y_2 v_2^{\omega \sharp_2} z_2) = (x_1, x_2)(u_1, u_2)^{\omega}(y_1, y_2)(v_1, v_2)^{\omega \sharp}(z_1, z_2) \in M_1 \times I_2$$

and the property is verified), or $(x_1 u_1^{\omega \sharp_1} y_1 v_1^{\omega \sharp_1} z_1, x_2 u_2^{\omega \sharp_2} y_2 v_2^{\omega} z_2) \in M_1 \times I_2$ (that implies

$$(x_1 u_1^{\omega \sharp_1} y_1 v_1^{\omega} z_1, x_2 u_2^{\omega \sharp_2} y_2 v_2^{\omega} z_2) = (x_1, x_2)(u_1, u_2)^{\omega \sharp}(y_1, y_2)(v_1, v_2)^{\omega}(z_1, z_2) \in M_1 \times I_2$$

and the property is shown). This verifies the first item of the Max-property.

Next let $(x_1, x_2), (u_1, u_2), (y_1, y_2), (v_1, v_2), (z_1, z_2) \in M_1 \times M_2$ be such that

$$(M_1 \times I_2) \cup (I_1 \times M_2) \ni (x_1, x_2) \left((u_1, u_2)(y_1, y_2)^{\omega\sharp}(v_1, v_2) \right)^{\omega\sharp} (z_1, z_2)$$

$$= \left(x_1 \left(u_1 y_1^{\omega\sharp_1} v_1 \right)^{\omega\sharp_1} z_1, x_2 \left(u_2 y_2^{\omega\sharp_2} v_2 \right)^{\omega\sharp_2} z_2 \right)$$

If $\left(x_1 \left(u_1 y_1^{\omega\sharp_1} v_1 \right)^{\omega\sharp_1} z_1, x_2 \left(u_2 y_2^{\omega\sharp_2} v_2 \right)^{\omega\sharp_2} z_2 \right) \in M_1 \times I_2$ (the other case being similar),

then by using Max-property of $\mathbf{M}_2, I_2$, either $\left(x_1 \left(u_1 y_1^{\omega\sharp_1} v_1 \right)^{\omega\sharp_1} z_1, x_2 (u_2 y_2^{\omega_2} v_2)^{\omega\sharp_2} z_2 \right) \in$
$M_1 \times I_2$ (that implies

$$\left(x_1 (u_1 y_1^{\omega} v_1)^{\omega\sharp_1} z_1, x_2 (u_2 y_2^{\omega} v_2)^{\omega\sharp_2} z_2 \right) = (x_1, x_2) \left((u_1, u_2)(y_1, y_2)^{\omega}(v_1, v_2) \right)^{\omega\sharp} (z_1, z_2) \in M_1 \times I_2$$

and the property is shown), or $\left(x_1 \left(u_1 y_1^{\omega\sharp_1} v_1 \right)^{\omega\sharp_1} z_1, x_2 \left(u_2 y_2^{\omega\sharp_2} v_2 \right)^{\omega} z_2 \right) \in M_1 \times I_2$ (that
implies

$$\left(x_1 \left(u_1 y_1^{\omega\sharp_1} v_1 \right)^{\omega} z_1, x_2 \left(u_2 y_2^{\omega\sharp_2} v_2 \right)^{\omega} z_2 \right) = (x_1, x_2) \left((u_1, u_2)(y_1, y_2)^{\omega\sharp}(v_1, v_2) \right)^{\omega} (z_1, z_2) \in M_1 \times I_2$$

and the property is verified). This verifies the second item of the Max-property.

◀

The above lemma implies that:

▶ **Lemma 1.15.** *If two cost functions are recognised by stabilisation monoid, ideal pairs that have Max-property, then their maximum is also recognised by a a stabilisation monoid, ideal pair that has Max-property.*

This completes the proof of the implication.

$4 \rightarrow 5$: **From stabilisation monoids to syntactic stabilisation monoids**

Assume f is recognised by a stabilisation monoid $\mathbf{M} = \langle M, \cdot, \sharp, \leq \rangle$, ideal I, and a morphism
$h : \Sigma^* \to M$ such that $\mathbf{M}, I$ has Max-property. We verify that the syntactic stabilisation
monoid $\mathbf{M}_f = \langle M_f, \cdot_f, \sharp_f, \leq_f \rangle$, and ideal I_f has the Max-property. By Theorem 2.16 there
is a surjective morphism $g : \mathbf{M} \to \mathbf{M}_f$ such that $I = g^{-1}(I_f)$ and $f = [\![\mathbf{M}_f, I_f, h \circ g]\!]$.

Let $x', u', y', v', z' \in M_f$ be such that $x' u'^{\omega\sharp_f} y' v'^{\omega\sharp_f} z' \in I_f$. Fix $x, u, y, v, z \in M$ such
that $g(x) = x', g(u) = u', g(y) = y', g(v) = v', g(z) = z'$. Then,

$$\begin{aligned}
x u^{\omega\sharp} y v^{\omega\sharp} z &\in g^{-1} \left(g(x u^{\omega\sharp_f} y v^{\omega\sharp} z) \right) \\
&= g^{-1} \left(g(x) g(u)^{\omega\sharp_f} g(y) g(v)^{\omega\sharp_f} g(z) \right) && \text{Since } g \text{ is a morphism} \\
&= g^{-1} \left(x' u'^{\omega\sharp_f} y' v'^{\omega\sharp_f} z' \right) \\
&\subseteq I && \text{By assumption and Theorem 2.16}
\end{aligned}$$

Since $\mathbf{M}, I$ has Max-property, either $x u^{\omega} y v^{\omega\sharp} z \in I$ or $x u^{\omega\sharp} y v^{\omega} z \in I$. Assume $x u^{\omega} y v^{\omega\sharp} z \in I$,
the other case is similar. Then, since $g(I) = I_f$, $g \left(x u^{\omega} y v^{\omega\sharp} z \right) = g(x) g(u)^{\omega} g(y) g(v)^{\omega\sharp_f} g(z) =$
$x' u'^{\omega} y' v'^{\omega\sharp_f} z' \in I_f$. This verifies Item 1 of the Max-property.

Next let $x', u', y', v', z' \in M_f$ be such that $x'(u'y'^{\omega\sharp_f} v')^{\omega\sharp} z' \in I_f$. Fix $x, u, y, v, z \in M$ such that $g(x) = x', g(u) = u', g(y) = y', g(v) = v', g(z) = z'$. Then,

$$
\begin{aligned}
x \left(uy^{\omega\sharp}v\right)^{\omega\sharp} z &\in g^{-1}\left(g\left(x\left(uy^{\omega\sharp}v\right)^{\omega\sharp} z\right)\right) \\
&= g^{-1}\left(g(x)\left(g(u)g(y)^{\omega\sharp_f}g(v)\right)^{\omega\sharp_f} g(z)\right) \quad \text{Since } g \text{ is a morphism} \\
&= g^{-1}\left(x'\left(u'y'^{\omega\sharp_f} v'\right)^{\omega\sharp_f} z'\right) \\
&\subseteq I \qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{By assumption and Theorem 2.16}
\end{aligned}
$$

Because $\mathbf{M}, I$ has Max-property, either $x(uy^{\omega\sharp}v)^{\omega} z \in I$, or $x(uy^{\omega}v)^{\omega\sharp} z \in I$. Assume the first case, the other being similar. Then $g\left(x(uy^{\omega\sharp}v)^{\omega} z\right) = g(x)\left(g(u)g(y)^{\omega}g(v)\right)^{\omega\sharp_f} g(z) = x'\left(u'y'^{\omega_f} v'\right)^{\omega\sharp_f} z' \in I_f$. This verifies the Item 2 of the Max property.

Thus the proof is completed.

5 → 6: From stabilisation monoids satisfying Max-property to cost regular expressions

Assume that the cost function f is recognised by the stabilisation monoid $\mathbf{M}$, ideal I and morphism $h : \Sigma^* \to \mathbf{M}$ such that $\mathbf{M}, I$ has Max-property.

▶ **Definition 1.16.** Let $\mathbf{M} = \langle M, \cdot, \sharp, \leq \rangle$ be a stabilisation monoid and $h : \Sigma^* \to M$ be a morphism. For an element $a \in M$, we write E_a for a regular expression denoting the language $h^{-1}(a)$ recognised by the finite monoid $\langle M, \cdot, \leq \rangle$ and the morphism h.

Let E be the S-regular expression

$$
E = \sum_{a \in I} E_a + \sum_{\substack{a \cdot b^\sharp \cdot c \in I, \\ a,b,c \in M}} \left(E_a \cdot (E_b)^{>n} \cdot E_c\right) \ . \tag{15}
$$

Let us note that E is in the desired form. It remains to show that the expression E and the triple $\mathbf{M}, h, I$ define the same cost function. First we prove the simple direction.

Computation trees are not sufficient for our purpose, we need the following extension of computation trees.

▶ **Definition 1.17** (Over-computation tree [8]). An n-*over-computation tree* t is a finite rooted ordered unranked tree in which each node x has an associated *output* in M and is of one of four types:

Leaf x has no children and has an associated *label* $a \in \Sigma$, and the output of x is *greater or equal* (according to the order $\leq$ of the monoid) $h(a)$;

Binary node x has exactly two children and and the output of x *greater or equal* to the product of the output of the first child and the output of the second child;

Idempotent node x has k children with $k \leq n$ and for some idempotent $e \in M$, the output of each child is e and the output of x is *greater or equal* to e.

Stabilisation node x has k children with $k \leq n$ and for some idempotent $e \in M$, the output of each child is equal to e and the output of x is *greater equal* to $e^\sharp$.

The *input* of the tree t is the word formed by the labels of the leaves of the tree, read left to right. The *output* of the tree t is the output of the root, and the neutral element of $\mathbf{M}$ if t is the empty tree.

For a fixed height $k \in \mathbb{N}$, define

$$[\![\mathbf{M}, h, I]\!]_k^{++}(w) = \sup\{n \mid \text{there exists an } n\text{-over-computation with input } w,$$
$$\text{output not in the Ideal } I, \text{ and height} \leq k\} .$$

It turns out the cost function $[\![\mathbf{M}, h, I]\!]_k^{++}$ is exactly the function $[\![\mathbf{M}, h, I]\!]$. It also does not depend on the choice of $k \geq 3|M|$.

▶ **Lemma 1.18.** $[\![E]\!] \preccurlyeq [\![\mathbf{M}, h, I]\!]$.

Proof. We claim the following: *For a word $w \in \Sigma^*$, if $w \in E[m \to n]$, then there is a m-over-computation tree of height $3|M|+3$ over the word w with output in the ideal I.* Next we prove the claim. Assume the word $w \in E[m \to n]$. Then by definition, there exist elements $a, b, c \in M$ and words $w_1, w_2, w_3 \in \Sigma^*$ such that $a \cdot b^\sharp \cdot c \in I$, $w = w_1 \cdot w_2 \cdot w_3$, and $w_1 \in E_a, w_2 \in (E_b)^m, w_3 \in E_c$. Furthermore, the word w_2 can be split into $w_2 = u_1 \cdots u_m$, $u_i \in \Sigma^*$ such that $u_i \in E_b$. Let $t_1, s_1, \ldots, s_n, t_2$ be m-computation trees of height $3|M|$ over the words $w_1, u_1, \ldots, u_n, w_3$ respectively. By definition, $v(t_1) \leq a, v(s_1) \leq b, \ldots, v(s_n) \leq b, v(t_2) \leq c$. Let $t_1', s_1', \ldots, s_n', t_2'$ be the m-over-computation trees obtained from $t_1, s_1, \ldots, s_n, t_2$ by relabelling the roots by the elements $a, b, \ldots, b, c$ respectively. Define t' to be the tree
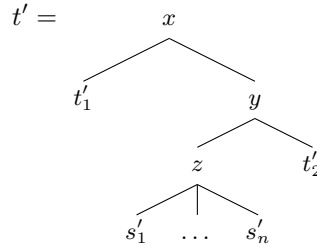
The vertex x, y, z are labelled respectively by $a \cdot b^\sharp \cdot c$, $b^\sharp \cdot c$, and $b^\sharp$. In particular the vertex z is a stabilisation node. The tree t' is a valid n-over-computer tree over the word w of height $3|M| + 3$.

Next we show that $[\![E]\!] \preccurlyeq [\![\mathbf{M}, h, I]\!]_{3|M|+3}^{++}$. Let $w \in \Sigma^*$ be a word and let $[\![E]\!](w) = m \in \mathbb{N}_\infty$. We have two cases.

Case 1: When $m \in \mathbb{N}$. Then by definition of $[\![E]\!](w)$, the word $w \in E[m \to n]$ and hence by the claim above there is a m-over-computation tree of height $3|M| + 3$ over the word w with the output in the ideal I. Therefore, by definition, $[\![\mathbf{M}, h, I]\!]_{3|M|+3}^{++}(w)$ is at leat m.

Case 2: When $m = \infty$. By definition of $[\![E]\!](w)$, there exist arbitrarily large $m \in \mathbb{N}$ such that the word $w \in E[m \to n]$. Hence by the above claim, for arbitrarily large $m \in \mathbb{N}$, there exist m-over-computation trees of height $3|M| + 3$ over the word w with output in the ideal I. Therefore, by definition, $[\![\mathbf{M}, h, I]\!]_{3|M|+3}^{++}(w)$ is ∞.

It follows that $[\![E]\!] \preccurlyeq [\![\mathbf{M}, h, I]\!]_{3|M|+3}^{++} \approx [\![\mathbf{M}, h, I]\!]$.

◀

Next we want to show that, $[\![\mathbf{M}, h, I]\!] \preccurlyeq [\![E]\!]$.

Let $\overline{\mathbf{M}}$ be the stabilisation monoid $\overline{\mathbf{M}} = \langle M, \cdot, \sharp', \leq \rangle$ where the stabilisation $\sharp'$ is the identity, i.e. $e^{\sharp'} = e$ for all idempotents e. Define the product stabilisation monoid $\mathbf{M}' = \mathbf{M} \times \overline{\mathbf{M}}$. Let $\mu : \mathbf{M}' \to \mathbf{M}$ be the morphism $\mu((a, b)) = a$.

▶ **Lemma 1.19** ([8], Lemma 4.3). *For μ a morphism of stabilisation monoids from $\mathbf{M}_1$ to $\mathbf{M}_2$, h a morphism from Σ^* to $\mathbf{M}_1$ and ideal I_2 of $\mathbf{M}_2$, we have:* $[\![\mathbf{M}_1, h, \mu^{-1}(I_2)]\!] = [\![\mathbf{M}_2, \mu \circ h, I_2]\!]$.

Let h' be the morphism $h' : \Sigma^* \to \mathbf{M}'$ defined as $h'(w) = (h(w), h(w))$. Lastly, let $I' = I \times M$, then by the previous lemma,

▶ **Lemma 1.20.** $[\![\mathbf{M}', h', I' = \mu^{-1}(I)]\!] = [\![\mathbf{M}, h = \mu \circ h', I]\!]$.

Assume for a word $w \in \Sigma^*$, $[\![\mathbf{M}', h', I']\!](w) = n \in \mathbb{N}$. Then there exist a n-computation tree t over the word w of height $3|M|^2$ with output in the ideal I'. For a vertex x in the tree, let us denote by $v(x)$ the output of the vertex x. For each vertex x in t we associate a $\sharp$-expression $E(x)$ inductively;

- if x is a leaf and $v(x) = (a, a) \in M \times M$: then $E(x) = a$,
- if x is a binary node with children y and z: then $E(x) = E(y) \cdot E(z)$,
- if x is an idempotent node with children $y_1, \ldots, y_k$: then $E(x) = E(y_1) \cdots E(y_k)$,
- if x is an stabilisation node with children $y_1, \ldots, y_k$: then $E(x) = (E(y_1) \cdots E(y_k))^{\omega\sharp}$.

We write $s(x) = u$ to mean that the input of the vertex x is the infix u of w.

▶ **Lemma 1.21.** *Let x be a vertex in t labelled by the pair $(a, b) \in M^2$ and let E be the $\sharp$-expression associated with x, then*

1. $val(\overline{E(x)}) = b = h(s(x))$.

Proof. (1). By induction on the height of the vertex x. When x is a leaf, then by definition $E(x) = a$. When x is a binary/idempotent node with children $y_1, \ldots, y_k$ that are labelled with elements $(a_1, b_1), \cdots, (a_k, b_k)$ respectively, then

$$
\begin{aligned}
val(E(x)) &= val(E(y_1) \cdots E(y_k)) && \text{by definition of } E(x) \\
&= a_1 \cdots a_k && \text{by I. H.} \\
&= a && \text{by definition of the computation tree.}
\end{aligned}
$$

Similarly, when x is a stabilisation node with children $y_1, \ldots, y_k$ that are labelled with elements $(a_1, b_1), \cdots, (a_k, b_k)$ respectively, then

$$
\begin{aligned}
val(E(x)) &= val\left((E(y_1) \cdots E(y_k))^{\omega\sharp}\right) && \text{by definition of } E(x) \\
&= (a_1 \cdots a_k)^{\omega\sharp} && \text{by I. H. and since } a_1 = \cdots = a_k \text{ is an idempotent} \\
&= a && \text{By definition of the computation tree.}
\end{aligned}
$$

(2). Again, by induction on the height of the vertex x. First we show that if x is labelled by (a, b) then $val(\overline{E(x)}) = b$. When x is a leaf, then by definition $\overline{E(x)} = b$. When x is a binary/idempotent node with children $y_1, \ldots, y_k$ that are labelled with elements $(a_1, b_1), \cdots, (a_k, b_k)$ respectively, then

$$
\begin{aligned}
val(\overline{E(x)}) &= val(\overline{E(y_1) \cdots E(y_k)}) && \text{by definition of } E(x) \\
&= val(\overline{E(y_1)} \cdots \overline{E(y_k)}) && \\
&= b_1 \cdots b_k && \text{by I. H.} \\
&= b && \text{by definition of the computation tree.}
\end{aligned}
$$

When x is a stabilisation node with children $y_1, \ldots, y_k$ that are labelled with elements $(a_1, b_1), \cdots, (a_k, b_k)$ respectively, then

$$
\begin{aligned}
\mathrm{val}(\overline{E(x)}) &= \mathrm{val}(\overline{(E(y_1) \cdots E(y_k))^{\omega\sharp}}) && \text{by definition of } E(x) \\
&= \mathrm{val}\left(\left(\overline{E(y_1)} \cdots \overline{E(y_k)}\right)^{\omega}\right) && \\
&= (b_1 \cdots b_k)^{\omega} && \text{by I. H.} \\
&= b && \text{since } b_1 = \cdots = b_k \text{ is an idempotent.}
\end{aligned}
$$

Next we show that if x is labelled by (a, b) then $b = h(s(x))$. When x is a leaf it is obvious. When x is a binary/idempotent/stabilisation node with children $y_1, \ldots, y_k$ that are labelled with elements $(a_1, b_1), \cdots, (a_k, b_k)$ respectively, then

$$
\begin{aligned}
h(s(x)) &= h(s(y_1) \cdots s(y_k)) && \text{by definition of } s(x) \\
&= h(s(y_1)) \cdots h(s(y_k)) && \text{since } h \text{ is a morphism} \\
&= b_1 \cdots b_k && \text{by I. H.} \\
&= b && \text{by definition of the computation tree.}
\end{aligned}
$$

This concludes the proof.

◀

▶ **Lemma 1.22.** *For a word $w \in \Sigma^*$, if there is a n-computation tree, $n \in \mathbb{N}$, over w whose output is in the ideal I', then*

1. *either $h(w) \in I$, or*
2. *there exists $u, w_1, \ldots, w_k, v \in \Sigma^*$, $k > n$, such that $w = u w_1 \cdots w_k v$, $h(w_1) = \cdots = h(w_k) = e \in M$ is an idempotent, and $h(u) \cdot e^{\omega\sharp} \cdot h(v) \in I$.*

Proof. Assume there is a n-computation tree t over w with height k whose value is in the ideal I'. By definition of I' and Lemma 1.21, it follows that $\mathrm{val}(E(x)) \in I$ for the root x of the tree t. For a vertex x in the tree, define $l(x)$ and $r(x)$ to be respectively the words $u, v \in \Sigma^*$ such that $w = u \cdot s(x) \cdot v$. Let P be the set of all vertices in t that satisfies the property that $h(l(x)) \cdot \mathrm{val}(E(x)) \cdot h(r(x)) \in I$. Next we prove the following claims:

Claim 1: *Let x be a binary or idempotent node. If x is in the P, then there is a child of x that is also in P.*

Claim 2: *Let x be a stabilisation node. If x is in the P, then either there is a child of x that is also in P, or $h(l(x)) \cdot (h(s(y_1)) \cdots h(s(y_k)))^{\omega\sharp} \cdot h(r(x)) \in I$ and $h(s(y_1)) = \cdots = h(s(y_k)) = b$ is an idempotent, where $y_1, \ldots, y_k$ are the children of x.*

Before proving the claims we first show that the claims imply the lemma. Firstly the root of the tree is in P. Inductively applying the Claims 1 and 2, we get that either there is a leaf x that is in P (in which case $h(w) = h(l(x)) \cdot E(x) \cdot h(r(x)) \in I$ and the Item 1 of the lemma is satisfied), or for some stabilisation node x with children $y_1, \ldots, y_k$, $h(l(x)) \cdot (h(s(y_1)) \cdots h(s(y_k)))^{\omega\sharp} \cdot h(r(x)) \in I$ and $h(s(y_1)) = \cdots = h(s(y_k)) = b$ is an idempotent. In the latter case, taking $u = l(x), w_1 = s(y_1), \ldots, w_k = s(y_k), v = r(x)$, the Item 2 of the lemma is satisfied. This proves the lemma.

Next we prove the claims.

Proof of Claim 1: Assume that x is a binary or an idempotent node that is in P and let $y_1, \ldots, y_k, 2 \leq k \in \mathbb{N}$ be its children. Then $E(x) = E(y_1) \cdots E(y_k)$. Applying the Lemma

1.6 item (1), $k-1$ times on the $\sharp$-expression $h(l(x)) \cdot E(y_1) \cdots E(y_k) \cdot h(r(x)) \in I$ yields that there is a $1 \le j \le k$ such that $h(l(x)) \cdot \overline{E(y_1)} \cdots \overline{E(y_{j-1})} \cdot E(y_j) \cdot \overline{E(y_{j+1})} \cdots \overline{E(y_k)} \cdot h(r(x)) \in I$. By Lemma 1.21 $\mathrm{val}(\overline{E(y_i)}) = h(s(y_i))$ for each i and hence

$$
\begin{aligned}
I \ni \mathrm{val}\left(h(l(x)) \cdot \overline{E(y_1)} \cdots \overline{E(y_{j-1})} \cdot E(y_j) \cdot \overline{E(y_{j+1})} \cdots \overline{E(y_k)} \cdot h(r(x)) \right) \\
= h(l(x)) \cdot h(s(y_1)) \cdots h(s(y_{j-1})) \cdot \mathrm{val}\left(E(y_j) \right) \cdot h(s(y_{j+1})) \cdots h(s(y_k)) \cdot h(r(x)) \\
= h(l(x) \cdot s(y_1) \cdots s(y_{j-1})) \cdot \mathrm{val}\left(E(y_j) \right) \cdot h(s(y_{j+1}) \cdots s(y_k)) \cdot h(r(x)) \\
= h(l(y_j)) \cdot \mathrm{val}\left(E(y_j) \right) \cdot h(r(y_j)) \ .
\end{aligned}
$$

Proof of Claim 2: Assume that x is a stabilisation node with children $y_1, \ldots, y_k$, $n \le k \in \mathbb{N}$ and label $(a,b) \in M^2$. Then by definition $E(x) = (E(y_1) \cdots E(y_k))^{\omega\sharp}$. Applying the Lemma 1.6 item (2) on the $\sharp$-expression $h(l(x)) \cdot (E(y_1) \cdots E(y_k))^{\omega\sharp} \cdot h(r(x)) \in I$ implies that either

$$
h(l(x)) \cdot (E(y_1) \cdots E(y_k))^{\omega} \cdot h(r(x)) \in I
$$

or

$$
h(l(x)) \cdot \left(\overline{E(y_1)} \cdots \overline{E(y_k)} \right)^{\omega\sharp} \cdot h(r(x)) \in I \ .
$$

Assume that $h(l(x)) \cdot (E(y_1) \cdots E(y_k))^{\omega} \cdot h(r(x)) \in I$. Since $\mathrm{val}(E(y_1)) = \cdots = \mathrm{val}(E(y_k))$ is an idempotent, $\mathrm{val}\left((E(y_1) \cdots E(y_k))^{\omega} \right) = \mathrm{val}\left(E(y_1) \cdots E(y_k) \right)$, and therefore $h(l(x)) \cdot E(y_1) \cdots E(y_k) \cdot h(r(x)) \in I$. Next we repeat the argument for the idempotent node and obtain that there is a vertex y_j such that

$$
h(l(y_j)) \cdot \mathrm{val}\left(E(y_j) \right) \cdot h(r(y_j)) \in I \ .
$$

In the second case we have $h(u) \cdot \left(\overline{E(y_1)} \cdots \overline{E(y_k)} \right)^{\omega\sharp} \cdot h(v) \in I$. By Lemma 1.21, $\mathrm{val}(\overline{E(y_i)}) = h(s(y_i)) = b$, for each i, is an idempotent, and hence $h(l(x)) \cdot (h(s(y_1)) \cdots h(s(y_k)))^{\omega\sharp} \cdot h(r(x)) \in I$.

◀

▶ **Lemma 1.23.** $\llbracket M, h, I \rrbracket \preccurlyeq \llbracket E \rrbracket$

Proof. We claim that there exists a correction function α, such that for all words $w \in \Sigma^*$, if $\llbracket M, h, I \rrbracket (w) > m \in \mathbb{N} \implies \llbracket E \rrbracket (w) \ge \alpha(m)$. To verify the claim, assume that $\llbracket M, h, I \rrbracket (w) > m$. Let α be a correction function, guaranteed by the Lemma 1.20, such that $\alpha(\llbracket \mathbf{M}, h, I \rrbracket (w)) = \llbracket \mathbf{M}', h', I' \rrbracket (w)$. Therefore, if $\llbracket M, h, I \rrbracket (w) > m$ then there is a $\alpha(m)$-computation tree over $\tilde{h}'(w)$ with output in the ideal I'. By the Lemma 1.22 either $h(w) \in I$ (which means $w \in \sum_{a \in I} E_a$, and hence $E(w) = \infty$ and the lemma is satisfied) or there exists $u, w_1, \ldots, w_k, v \in \Sigma^*$, $k > \alpha(m)$, such that $w = u w_1 \cdots w_k v$, $h(w_1) = \cdots = h(w_k) = e \in M$ is an idempotent, and $h(u) \cdot e^{\omega\sharp} \cdot h(v) \in I$. In the latter case $w \in \left(E_{h(u)} \cdot (E_e)^{>n} \cdot E_{h(v)} \right) [n \leftarrow \alpha(m)]$, and hence $\llbracket E \rrbracket (w) > \alpha(m)$. Hence the claim is proved. It follows that for all words $w \in \Sigma^*$, $\llbracket M, h, I \rrbracket (w) \le \llbracket E \rrbracket (w)$ and the lemma is proved.

◀

From Lemma 1.18 and Lemma 1.23 we conclude that $\llbracket M, h, I \rrbracket = \llbracket E \rrbracket$ and this proves the direction.

6→1 : From Cost regular expressions to max-automata

Assume we are given a cost regular expression $r = h + \sum_{i=1}^{k} e_i$ where h is a regular expression and each e_i is of the form $ef^{>n}g$ where e, f and g are regular expressions. We construct a max-automaton $\mathcal{A}$ such that $[\![\mathcal{A}]\!] = [\![r]\!]$. Let $\mathcal{A}_h$ be a finite state automaton that accept the cost function corresponding to the regular expression h. We observe that it is sufficient to show how to construct a max-automaton $\mathcal{A}_i$ that accepts the cost function $[\![e_i]\!]$. Then the disjoint union $\mathcal{A}$ of the automata $\mathcal{A}_h, \mathcal{A}_1, \ldots, \mathcal{A}_k$ accepts the cost function

$$[\![\mathcal{A}]\!] = \max\left([\![\mathcal{A}_h]\!], [\![A_1]\!], \ldots, [\![A_k]\!]\right) = \max\left([\![h]\!], [\![e_1]\!], \ldots, [\![e_k]\!]\right) = \left[\!\!\left[h + \sum_{i=1}^{k} e_i \right]\!\!\right] = [\![r]\!] \ .$$

To construct the automaton $\mathcal{A}_i$, we let $\mathcal{A}_e, \mathcal{A}_f, \mathcal{A}_g$ be finite state automata corresponding to the regular expression e, f and g. Furthermore assume that each of these automata has a unique initial state and a unique final state such that no transition enters the initial state and no transition leaves the final state. This is easy to achieve making use of nondeterminism. For convenience we allow transitions on the *empty word*. Note that using standard techniques it is possible to remove the transitions the on empty word from a max-automaton; so it does not affect the expressiveness of the class. Let $\mathcal{A}'_f$ be the max-automaton that is obtained from $\mathcal{A}_f$ by adding a transition on the empty word from the final state to the initial state that increments the unique counter. Finally we define $\mathcal{A}_i$ by chaining the automata $\mathcal{A}_e, \mathcal{A}'_f$ and $\mathcal{A}_g$, i.e. we add transitions on empty word from the final state of $\mathcal{A}_e$ to the initial state of $\mathcal{A}'_f$, as well as from the final state of $\mathcal{A}'_f$ to the initial state of $\mathcal{A}_g$, that does not touch the counter. Moreover we reset the counter of $\mathcal{A}_i$ precisely at the final states (namely the final state of $\mathcal{A}_g$). We claim that $[\![\mathcal{A}_i]\!] = [\![e_i]\!]$. Let w be a word in Σ^*.

Then, $[\![e_i]\!](w) > m \in \mathbb{N}$,

    iff $w \in L(ef^r g)$, for some $\mathbb{N} \ni r > m$,

    iff $w = xy_1 \cdots y_r z$ where $r > n$, $x \in L(e)$, $y_1, \ldots, y_r \in L(f)$ and $z \in L(g)$

    iff $\mathcal{A}_e$ has a run on x from its initial state to its final state, and $\mathcal{A}'_f$ has a run on $y_1 \cdots y_m$ from its initial state to its final state that increments the counter $r - 1$ times, and $\mathcal{A}_g$ has a run on z from its initial state to its final state,

    iff $\mathcal{A}_i$ has a run on w that increments the counter $r - 1$ times,

    iff $[\![\mathcal{A}_i]\!](w) > m - 1$.

It follows that $[\![\mathcal{A}_i]\!] = [\![e_i]\!]$. This completes the proof.

B   Characterisations of min-automata

This section is devoted to the proof of this Theorem 4.2 .

1 → 2

If $\mathcal{A}$ is a min-automaton we show that we can write a formula in the fragment expressing its semantic.

    Indeed, let $\mathcal{A} = (Q, A, \Delta, I, F)$, with $\Delta \subseteq Q \times \{0, 1\}$. Let $Q = \{q_1, \ldots, q_k\}$, and $u \in A^*$ be a word. We remind that the structure induced by this word is the set of positions, with predicates indicating the label in A of each position. For short we will note $a(x)$ the label of position x. If $X, X_1, \ldots, X_k$ are sets, we write $\psi(X, X_1, \ldots, X_k)$ as a conjunction of the following MSO-expressible statements:

- the X_i's form a partition of the set of positions. We will note $p(x)$ the unique p_i such that $x \in X_i$.
- for all consecutive positions $x \in X_i$, $y \in X_j$, there is an action τ such that $(q(x), a(x), \tau, q(y)) \in \Delta$. Moreover if τ must be 1 then $x \in X$.
- If x_0 is the first position, then $q(x_0) \in I$.
- If x_f is the last position, there is a transition of the form $(q(x_f), a(x_f), \tau, p)$ with $p \in F$.

Finally, let $\varphi(X) = \exists X_1, \exists X_2, \ldots, \exists X_k, \psi(X, X_1, \ldots, X_k)$. This formula expresses the fact that there is an accepting run of $\mathcal{A}$ where X represents the set of increments. Finally, $\phi = \exists X(\varphi(X) \wedge |X| \leq N)$ expresses that there is a run of $\mathcal{A}$ with at most n increments. Since the semantics of both cost MSO and min-automata are based on an infimum on possible choices, we get the equality $[\![\phi]\!] = [\![\mathcal{A}]\!]$. Notice that exact values are preserved by this translation.

1 ↔ 7

The $1 \leftrightarrow 7$ direction is trivial: any min-automaton is a B-automaton without reset.

Conversely, assume the cost function f is recognised by a B-automaton without reset $\mathcal{A} = (Q, A, \Gamma = [k], \Delta, I, F)$. We define the min-automaton $\mathcal{A}'$ that has the same set of states and transitions as that of $\mathcal{A}$ such that $\mathcal{A}'$ increments its (only) counter whenever $\mathcal{A}$ increments some counter. Formally $\mathcal{A}' = (Q, A, [1], \Delta', I, F)$ where

$$\Delta' = \left\{ (p, a, \gamma', q) \;\middle|\; (p, a, \gamma, q) \in \Delta, \text{and } \gamma' = \begin{cases} \epsilon & \text{if } \gamma = \{\epsilon\}^k \\ \mathtt{ic} & \text{otherwise} \end{cases} \right\}$$

By definition of the automaton $\mathcal{A}'$ it is clear that for all words $w \in A^*$, $[\![\mathcal{A}]\!](w) \leq [\![\mathcal{A}']\!](w)$. In the other direction if $[\![\mathcal{A}]\!](w)$ is $n \in \mathbb{N}$ then $[\![\mathcal{A}']\!](w)$ is at most $k \cdot n$ (the worst case is when all the counters of $\mathcal{A}$ are incremented at disjoint sets of positions) and hence $[\![\mathcal{A}']\!](w) \leq k \cdot [\![\mathcal{A}]\!](w)$ for all words $w \in A^*$. Therefore both automata $\mathcal{A}$ and $\mathcal{A}'$ define the same cost function.

2 → 3

Let $\mathbf{D}$ be the smallest class of cost functions containing the cost functions count and regular languages that is closed under min, max and inf-projections.

Let $\psi = \exists X(\varphi(X) \wedge |X| \leq N)$, where φ is a MSO formula. Let f be the function recognised by φ, on alphabet A. Let L be the language on $A \times \{0, 1\}$, recognised by φ, where the second component indicates membership in X. Let f_1 be the cost function on $A \times \{0, 1\}$ defined by $|X| \leq n$. Let $g : A \times \{0, 1\} \to \{a, b\}$ defined by $g(x, 1) = a$ and $g(x, 0) = b$ for all $x \in A$. We get that for all $w \in (A \times \{0, 1\})^*$, $f_1(w) = \inf\{\mathrm{count}(u) | g(w) = u\}$ and therefore $f_1 \in \mathbf{D}$. This means that $f_2 = \max(\chi_L, f_1)$ is also in $\mathbf{D}$. Finally, let $\pi : A \times \{0, 1\} \to A$ be the projection on the first component, we get that for all $v \in A^*$, $f(v) = \inf\{f_2(w) | \pi(w) = v\}$. This concludes the proof that $f \in \mathbf{D}$.

3 → 1

Since min-automata can recognise all regular languages and the function count, it suffices to show that there are closed under min, max, and inf-projection.

Let $\mathcal{A}_1, \mathcal{A}_2$ be min-automata, then their union recognises $\min([\![\mathcal{A}_1]\!], [\![\mathcal{A}_2]\!])$, as non-determinism resolves as a minimum.

To compute the max, we build the product $\mathcal{A}_1 \times \mathcal{A}_2$ of $\mathcal{A}_1$ and $\mathcal{A}_2$, where

- a state is initial (resp. final) if both components are initial (resp. final),
- a transition in the product performs an increment if one of the automaton performs an increment.

Therefore, a run of $\mathcal{A}_1 \times \mathcal{A}_2$ of value n can be matched to runs of $\mathcal{A}_1$ and $\mathcal{A}_2$ with values at most n. Conversely, any two runs of $\mathcal{A}_1$ and $\mathcal{A}_2$ of value at most n can be matched to a run of $\mathcal{A}_1 \times \mathcal{A}_2$ of value at most $2n$.

Therefore, $[\![\mathcal{A}_1 \times \mathcal{A}_2]\!] = \max([\![\mathcal{A}_1]\!], [\![\mathcal{A}_2]\!])$, since as soon as one automaton computes a big value, then the product does too.

Finally, let $\mathcal{A}$ be a min-automaton on alphabet A, and $h : A \to B$. We build a min-automaton on B for the inf-projection of $[\![\mathcal{A}]\!]$ with respect to h by simply guessing an antecedent via h of each letter, and running $\mathcal{A}$ on the guessed letters in A. Since non-determinism resolves as a minimum, this automaton computes indeed an inf-projection with respect to h.

$1 \to 4$

Let $\mathcal{A} = (Q, A, \Delta, I, F)$ be a min-automaton, with $\Delta \subseteq Q \times A \times \{0, 1\} \times Q$. Let $\mathbf{M}$ be its transition monoid, i.e. $\mathbf{M}$ is a set of matrices of size $Q \times Q$ with coefficients in $\{0, 1, \infty\}$. We show that $\mathbf{M}$ can be equipped with a structure of stabilisation monoid recognising f, and verifying the Min-property.

Product in $\mathbf{M}$ is the classical matrix product over the semiring $\{0, 1, \infty\}$ equipped with operations $(\min, \max)$, where min and max are with respect to the order $\infty \le 1 \le 0$. We will use the product notation instead of max in $\{0, 1, \infty\}$.

The product in $\mathbf{M}$ can therefore be written $E \cdot F = \inf\{E(p, r)E(r, r)E(r, q) \mid r \in Q\}$.

We also define a $\sharp$ operation on $\{0, 1, \infty\}$ by $0^\sharp = 0$ and $1^\sharp = \infty^\sharp = \infty$. Stabilisation in $\mathbf{M}$ is defined on idempotents y $E^\sharp(p, q) = \inf\{E(p, r)E(r, r)^\sharp E(r, q) \mid r \in Q\}$.

If $a \in A$, we associate to it the matrix M_a defined by $M_a(p, q) = x$ if $(p, a, x, q) \in \Delta$ and ∞ otherwise.

Finally, the support of the stabilisation monoid $\mathbf{M}$ is the smallest set of matrices containing $\{M_a \mid a \in A\}$ and close under product and stabilisation.

We define a special order $\lhd$ in $\mathbf{M}$ by using the component-wise order induced by the order $\lhd = \{(0, 0), (1, 1), (\infty, \infty), (\infty, 1)\}$ on coefficients.

▶ **Lemma 2.1.** $\mathbf{M}$ *equipped with the order $\lhd$ is a stabilisation monoid recognising f.*

Proof. The fact that $\mathbf{M}$ is a stabilisation monoid for f is shown in [6].

We have to verify that $\lhd$ verifies the order axioms of stabilisation monoids. It is clearly an order, since it is defined as the component-wise extension of a linear order.

We have to show closure by product. For this, it suffices to show that $\lhd$ on actions $\{0, 1, \infty\}$ is closed under min and max, i.e. for all actions a, b, c, d if $a \lhd b$ and $c \lhd d$, then $\min(a, c) \lhd \min(b, d)$ and $\max(a, c) \lhd \max(b, d)$.

The only interesting cases (up to symmetries) is when $a = b$, $c = \infty$ and $d = 1$. If a is minimal or maximal, then the result is trivial, as it is either neutral or absorbing for min / max. We are left with the case $a = b = 1$, and the result is also true: $1 \lhd 1$ and $\infty \lhd 1$.

Since the product in $\mathbf{M}$ is defined with respect to the $(\min, \max)$ semiring, this implies that whenever $E_1 \lhd E_2$ and $F_1 \lhd F_2$ then $E_1 \cdot F_1 \lhd E_2 \cdot F_2$.

Let $E \in \mathbf{M}$ be an idempotent, we verify $E^\sharp \lhd E$. Let $p, q \in Q$, since E is idempotent we have $E(p, q) = \inf\{E(p, r)E(r, r)E(r, q) \mid r \in Q\}$. But $E^\sharp(p, q) = \inf\{E(p, r)E(r, r)^\sharp E(r, q) \mid r \in Q\}$.

We show that there is $r \in Q$ such that $E(p,q) = E(p,r)E(r,r)E(r,q)$ and $E^\sharp(p,q) = E(p,r)E(r,r)^\sharp E(r,q)$. If $E(p,q) = \infty$ then any r will do. Otherwise, if $E(p,q) = 0$, then there is r with $E(p,r) = E(r,r) = E(r,q) = 0$, and the same r witnesses $E^\sharp(p,q) = 0$. Finally, if $E(p,q) = 1$, witnessed by some r, we are left with two cases: either $E^\sharp(p,q) = \infty$, and we can use the same r as witness, or $E^\sharp(p,q) = 1$, meaning there is r' such that $E^\sharp(p,q) = E(p,r')E(r',r')^\sharp E(r',q)$. This implies $E(r',r') = 0$, and the same r' can be used to show that $E(p,q) = 1$. Since $\lhd$ on is closed by product (max) on actions, and this is true for all (p,q), we get $E^\sharp \lhd E$.

We finally show closure under stabilisation: if $E \lhd F$ then $E^\sharp \lhd F^\sharp$. Let $p,q \in Q$, we have $E(p,q) = E(p,r)E(r,r)E(r,q)$ for some $r \in Q$. We know that $F(p,q) \leq F(p,r)F(r,r)F(r,q)$. Assume $F(p,q) < F(p,r)F(r,r)F(r,q)$, it means that there is a pair $\lambda \in \{(p,r),(r,r),(r,q)\}$ with $E(\lambda) \neq F(\lambda)$. But since $E(\lambda) \lhd F(\lambda)$, we get $E(\lambda) = \infty$ and $F(\lambda) = \texttt{ic}$, and therefore $E^\sharp(p,q) = \infty$ and $F^\sharp(p,q) \in \{1,\infty\}$, guaranteeing $E^\sharp(p,q) \lhd F^\sharp(p,q)$. Otherwise, we have for all three pairs λ, $E(\lambda) = F(\lambda)$, and since $E^\sharp(p,q) = E(p,r)E(r,r)^\sharp E(r,q)$ and $F^\sharp(p,q) = F(p,r')F(r',r')^\sharp F(r',q)$, we get $E^\sharp(p,q) \lhd F^\sharp(p,q)$. This concludes the proof that $E^\sharp \lhd F^\sharp$.

All the axioms are verified, so $\lhd$ is indeed a stabilisation monoid order for $\mathbf{M}$.

◄

▶ **Lemma 2.2.** *For all $a,b \in \mathbf{M}$, if $a \mathcal{R} b$ then $b \lhd a$.*

Proof. Straightforward, by definition of the relation $\mathcal{R}$ and Lemma 2.1.   ◄

▶ **Lemma 2.3.** *If $E \lhd F^{\omega\sharp}$, then $F^{\omega\sharp} = F^{\omega\sharp} E F^\sharp$.*

Proof. First, we can assume that F is an idempotent, by replacing it with F^ω, since we have $E \lhd F^\omega$.

Let $p,q \in Q$, and $K = F^\sharp E F^\sharp$. We remind that

$$K(p,q) = \inf\{F^\sharp(p,r)E(r,s)F^\sharp(s,q) | r,s \in Q\}.$$

if $F^\sharp(p,q) = \infty$, then we have $K(p,q) = \infty$, since for any $K(p,q) \lhd F^\sharp(p,q)$. If $F^\sharp(p,q) = 0$, then there are r,s with $F^\sharp(p,r)F^\sharp(r,s)F^\sharp(s,q) = 0$ ($F^\sharp$ is idempotent). Since $E(r,s) \lhd F^\sharp(r,s)$, we get $E(r,s) = 0$ (by definition of $\lhd$), and finally $K(p,q) = 0$.

Finally, if $F^\sharp(p,q) = 1$, then there is r with $F(p,r)F(r,r)^\sharp F(r,q) = 1$. This implies $F(r,r) = 0$ and $F(p,r)F(r,q) = 1$. Thus $E(r,r) = 0$, and $F^\sharp(p,r) \leq F(p,r)F(r,r)^\sharp \leq 1$. Similarly, $F^\sharp(r,q) \leq F(r,r)^\sharp F(r,q) \leq 1$. We obtain $K(p,q) \leq F^\sharp(p,r)E(r,r)F^\sharp(r,q) \leq 1$, i.e. $K(p,q) \in \{0,1\}$. Since we know that $K(p,q) \lhd 1$, we finally obtain $K(p,q) = 1$.

Since this is true for all p,q, we can conclude $F^\sharp = F^\sharp E F^\sharp$.   ◄

Lemmas 2.2 and 2.3 put together imply that $\mathbf{M}$ has the Min-property.

$4 \rightarrow 5$

Let $\mathbf{M}$ be a stabilisation monoid with the Min-property.

We want to show that this property is preserved by quotient, i.e. if there is a surjective stabilisation monoid morphism $\pi : \mathbf{M} \rightarrow \mathbf{M}'$, then $\mathbf{M}'$ satisfies the same condition.

Let $a',b' \in \mathbf{M}'$ such that $a'^{\omega\sharp} \mathcal{R} b'$. By induction on the definition of $\mathcal{R}$, we show that there are $a,b \in \mathbf{M}$ with $a^{\omega\sharp} \mathcal{R} b$, $\pi(a) = a'$ and $\pi(b) = b'$. Indeed, if $b' = a'^{\omega\sharp}$, then there is $a \in \mathbf{M}$ such that $\pi(a) = a'$ and by taking $b = a^{\omega\sharp}$ we get $\pi(b) = b'$, and $a^{\omega\sharp} \mathcal{R} b$. The induction cases follow straightforwardly from the fact that π is a morphism.

Therefore, we have $a^{\omega\sharp} = a^{\omega\sharp} b a^{\omega\sharp}$, and since π is a morphism, $a'^{\omega\sharp} = a'^{\omega\sharp} b' a'^{\omega\sharp}$.

Thus, $\mathbf{M}'$ satisfies the Min-property.

This is enough to show that if a cost function f is recognised by a monoid $\mathbf{M}$ with the Min-property, then the syntactic monoid of f has the Min-property, since it is a quotient of $\mathbf{M}$.

$5 \rightarrow 1$

We assume that the minimal stabilisation monoid $\mathbf{M}$ of f satisfies the Min-property.

The notion of under-computation used in this proof is defined in [8]. A tree is an under-computation where any node can be labelled by any element smaller than what is indicated by the rule for computation trees.

▶ **Definition 2.4.** A stabilising tree is a under-computation tree where all idempotent nodes are in fact stabilising: if a node N has more than 3 children, all labeled by an idempotent e, then the label of N is smaller than $e^\sharp$.

▶ **Lemma 2.5.** *Let $u = u_1 \ldots u_n \in \mathbf{M}^*$ such that there is a computation tree over a word of value a and height h. Then for any $u' = u'_1 \ldots u'_n$ such that for all i we have $u_i \, \mathcal{R} \, u'_i$, there is a stabilising tree over u' of value b and height at most $3|M|h$, such that $a \, \mathcal{R} \, b$.*

Proof. We prove this result by induction on the number of leaves. The rough principle is to replace idempotent nodes by stabilisation nodes everywhere in the tree. The base case of trees of height 0 consisting of a single node is trivial. The product case is obtained by the fact that $\mathcal{R}$ is closed under product. The only interesting case is when the root of the tree is an idempotent/stabilisation node, of root $E \in \{e, e^\sharp\}$. Subtrees below the root are trees $t_1 \ldots t_k$ sharing the same idempotent value e. By induction hypothesis, for each $i \in [1, k]$ there is a n-stabilising tree t_i of value e_i and height at most $3|M|(h-1)$, such that $e \, \mathcal{R} \, e_i$. By another use of our induction hypothesis, there is a computation tree t_e on $e_1 \ldots e_k$ with value e' such that $E \, \mathcal{R} \, e'$, of height at most $3|M|$. . Plugging $t_1 \ldots t_k$ at the leaves of t_e yields the wanted stabilising tree, of height at most $3|M|(h-1) + 3|M| = 3|M|h$. ◀

We will call *frontier tree* a tree such that each stabilisation node is the root of a stabilising tree.

▶ **Lemma 2.6.** *Given a n-computation tree of value a and height h over M, there is an n-under-computation tree of value a and height at most $3|M|h$ that is a frontier tree.*

Proof. We proceed by induction on the height on the computation tree t. The only nontrivial case is when the root of t is a stabilisation node of value $e^\sharp$, with children $t_0 \ldots t_{k+1}$ each of value e.

We apply the induction hypothesis on t_0 and t_{k+1}, and get trees t' and t'' of same value and height at most $3|M|(h-1)$, that are frontier trees. Since t' and t'' are under-computations, we can label their roots $e^\sharp$ instead of e.

We now consider the tree of value $e^\sharp$ with children $t_1, \ldots, t_l$, and apply Lemma 2.5 on it, getting a stabilising tree T of value b and height at most $3|M|(h-1)$, such that $e^\sharp \, \mathcal{R} \, b$.

We build a tree of height 3 with leaves $e^\sharp$, b, $e^\sharp$. We can now use the fact that $\mathbf{M}$ has Min-property, and conclude that the product $e^\sharp \cdot b \cdot e^\sharp$ evaluates to $e^\sharp$.

The resulting under-computation is a frontier tree, as it is obtained by combining two frontier trees and a stabilising tree. Its height is at most $3|M|(h-1) + 3 \le 3|M|h$. ◀

We will now describe how to build a min-automaton recognising f.

We will build an automaton $\mathcal{B}$ without counters implementing Lemma 2.5. Each state of the automaton $\mathcal{B}$ will be mapped to an element $b \in \mathbf{M}$ that we call its value. The value of a run of $\mathcal{B}$ is the value of its last state. On input word u, a run ρ of value b of the automaton $\mathcal{B}$ will witness the existence of a stabilising tree of value $\leq_{\min} b$.

This automaton is obtained by a modification of the algorithm from [6] that builds a B-automaton from a finite stabilisation monoid. In this algorithm, a state of the automaton is of the form $(m_1, \ldots, m_k)$ where the m_i's are elements of M from different J-classes. The value of such state is the product $m_1 \cdots m_k$. When m_k is idempotent, the automaton is allowed to stabilise it to obtain $m_k^\sharp$ before continuing the computation (this operation is labeled by a reset of a corresponding counter). Here, we will always do this operation, i.e. any idempotent encountered will be stabilised immediately. The rest of the construction is the same. Therefore, no counter is needed, and it is straightforward to show that runs of this automaton correspond to stabilising trees.

We can now build the main min-automaton $\mathcal{A}$ recognising f. The idea behind $\mathcal{A}$ is to implement Lemma 2.6, i.e. guess a frontier tree witnessing that the input word is accepted within a certain threshold.

The structure of $\mathcal{A}$ is defined by composition with the automaton $\mathcal{B}$: $\mathcal{A}$ will guess a factorisation of the input word u in $u_1 \ldots u_n$. The value of such a run is n: the counter is incremented each time we enter a new factor. On each factor b_i, the automaton $\mathcal{B}$ is used, and a value b_i is output by the guessed run. Finally, a factorisation tree is built on $b_1 \ldots b_n$, by using another variant of the construction from [6], where this time stabilisation is not allowed (therefore no counter is needed).

▶ **Lemma 2.7.** $\mathcal{A}$ *is a min-automaton recognising* f.

Proof. We show that we can identify runs of $\mathcal{A}$ with frontier trees described in Lemma 2.6.

First, if $f(u) \leq n$, then by Lemma 2.6, there is a frontier tree of threshold $\alpha_1(n)$ and value $b \in I$, where α_1 depends only on $|M|$.

Then in the ω-part of the tree (upper part where nothing is stabilised), all idempotent nodes have at most $\alpha_1(n)$ children. Since the height of the tree is bounded by a fixed constant, we get that the total number of nodes lying at the frontier is $n = \leq \alpha(n)$ where α depends only on $|M|$. If $t_1, \ldots, t_{\alpha(n)}$ are the tree rooted at the frontier, with leaves labelled by $u_1, \ldots, u_{\alpha(n)}$, respectively, then we can build a run of $\mathcal{A}$ following this factorisation, with value $\alpha(n)$. This shows that $[\![\mathcal{A}]\!] (u) \leq \alpha(n)$.

Conversely, any run of $\mathcal{A}$ with n increments witnesses the existence of a frontier tree of value at most n. This achieves the proof that $[\![\mathcal{A}]\!] \approx f$.   ◀

$1 \rightarrow 6$

We adapt the classical algorithm translating automata to regular expressions.

In this algorithm, intermediate automata have their transitions labeled by expressions. Two new states are added: one initial i, and one final f. Transitions are updated such that i becomes the only initial state and f the only final. Then the other states are inductively removed, while the transition structure is updated with expressions as labels, preserving the languages if the automaton if we interpreted transitions as accepting words from a language instead of letters. In particular, if a state q is removed, for each path of the form $p \xrightarrow{e} q \xrightarrow{e'} q \xrightarrow{e''} r$, a new transition $p \xrightarrow{ee'*e''} r$ is added. Additionally, for each path $p \xrightarrow{e} q \xrightarrow{e'} r$, a new transition $p \xrightarrow{ee'} r$ is added.

The algorithms proceeds until the automaton is of the form $i \xrightarrow{e} f$, and returns the expression e.

Here, each transition is additionally labeled by 0 or 1, and we only need to slightly adapt the above algorithm. We keep track of labels $0, 1$, with 1 absorbing for the concatenation. In case of two transitions $p \xrightarrow{e:0} q$, $p \xrightarrow{e':1} q$, we keep them separated instead of merging them in a transition labeled $e + e'$ as it was done in the different algorithm. We only merge transitions labeled with the same action. When a state with self-loop is removed, we will replace the Kleene star by $\leq n$ if the self-loop is labeled by action 1. This bounds by n the number of times the self-loop can be performed. Since the result path is still labeled 1, it is clear that no Kleene star can be generated on top of this $\leq n$ operator, and the grammar described in item 6 will be respected.

We show the equivalence of the resulting B-regular expression e with the original min-automaton $\mathcal{A}$. Let K be the number of times two 1 are concatenated in the above algorithm. Any run of $\mathcal{A}$ of value n witnesses a factorising choice for e of value at most n, since distincts increments in e can be matched to distinct increments in the run of $\mathcal{A}$. Conversely, any factorisation choice of value n for e witnesses a run of $\mathcal{A}$ of value at most $2^K n$, since an increment in e can represent at most 2^K increments in $\mathcal{A}$. This achieves the proof of $[\![\mathcal{A}]\!] \approx [\![e]\!]$.

6 → 7

Assume the cost function f is expressible by a B-regular expression E. By induction on the structure of E we prove that there is a B-automaton with k counters with no reset that recognises $[\![E]\!]$, where k is the number of subexpressions of the form $F^{\leq n}$ in E.

For the base case, when E is of the form $a \in A$ or F it defines a regular language and there is a finite state automaton recognising the cost function corresponding to E.

Next consider the inductive step. Let E_1 and E_2 be two cost regular expressions. By induction hypothesis we obtain two B-automata without reset $\mathcal{A}_1$ and $\mathcal{A}_2$,

$\mathcal{A}_1 = (Q_1, A, \Gamma_1 = [k_1], \Delta_1, I_1, F_1)$
$\mathcal{A}_2 = (Q_2, A, \Gamma_2 = \{k_1 + 1 \ldots, k_1 + k_2\}, \Delta_2, I_2, F_2)$

such that
- $[\![A_1]\!] = [\![E_1]\!]$,
- $[\![A_2]\!] = [\![E_2]\!]$,
- $Q_1 \cap Q_2 = \emptyset$,
- there are no incoming transitions to states I_1 and I_2, and no outgoing transitions from F_1 and F_2,
- k_1 and k_2 are respectively the number of subexpressions of the form $F^{\leq n}$ in E_1 and E_2 respectively.

If E is of the form $E_1 + E_2$ then disjoint union of the automata $\mathcal{A}_1$ and $\mathcal{A}_2$,

$\mathcal{A}_1 \cup \mathcal{A}_2 = (Q_1 \cup Q_2, A, \Gamma_1 \cup \Gamma_2 = [k_2], \Delta_1 \cup \Delta_2, I_1 \cup I_2, F_1 \cup F_2)$

recognise the cost function $[\![E_1 + E_2]\!]$ since $[\![\mathcal{A}_1 \cup \mathcal{A}_2]\!] = \min([\![\mathcal{A}_1]\!], [\![\mathcal{A}_1]\!]) = \min([\![E_1]\!], [\![E_1]\!]) = [\![E_1 + E_2]\!]$.

If E is of the form $E_1 \cdot E_2$ then we obtain the automata $\mathcal{A}_1 \cdot \mathcal{A}_2$ by merging the final states of $\mathcal{A}_1$ with the initial states of $\mathcal{A}_2$, i.e.,

$\mathcal{A}_1 \cdot \mathcal{A}_2 = (Q_1 \cup Q_2, A, \Gamma_1 \cup \Gamma_2 = [k_2], \Delta', I_1, F_2)$,
$\quad \Delta' = \Delta_1 \cup \Delta_2 \cup \{(p_1, a, \gamma, r) \mid (p, a, \gamma, q) \in \Delta_1, q \in F_1, r \in I_2\}$.

It is easy to verify that for a word $w \in A^*$,

$$\llbracket \mathcal{A}_1 \cdot \mathcal{A}_2 \rrbracket (w) = \min_{w_1, w_2 \in A^*} \{ \llbracket \mathcal{A}_1 \rrbracket (w_1) + \llbracket \mathcal{A}_2 \rrbracket (w_2) \mid w = w_1 w_2 \}$$

$$= \min_{w_1, w_2 \in A^*} \{ \llbracket E_1 \rrbracket (w_1) + \llbracket E_2 \rrbracket (w_2) \mid w = w_1 w_2 \}$$

$$= \llbracket E_1 \cdot E_2 \rrbracket (w) \ .$$

Next assume E is of the form $E_1^{\leq n}$. We add a new state $q' \notin Q_1$ and a new counter $k + 1$ to the automaton $\mathcal{A}_1$ such that

- q is the only initial and final state, and
- on every letter $a \in A$, $\mathcal{A}$ has a transition $(q, a, \gamma,)$

$$\mathcal{A}^{\leq n} = \Big(Q_1 \cup \{q\}, A, \Gamma_1 = [k_1 + 1], \Delta_1 \cup \{(q, a, i_{k_1+1}, q_0) \mid a \in A, q_0 \in I\}$$

$$\cup \{(q, a, i_{k_1+1}, q_0) \mid a \in A, q_0 \in I\}, \{q\}, F_1 \Big)$$

$$\llbracket \mathcal{A}^{\leq n} \rrbracket (w) = \min_{n \in \mathbb{N}} \{ n, \{ \llbracket \mathcal{A}_1 \rrbracket (w_1) + \cdots + \llbracket \mathcal{A}_1 \rrbracket (w_n) \mid w = w_1 \cdots w_n, w_1, \ldots, w_n \in A^+ \} \}$$

$$= \min_{n \in \mathbb{N}} \{ n, \{ \llbracket E_1 \rrbracket (w_1) + \cdots + \llbracket E_1 \rrbracket (w_n) \mid w = w_1 \cdots w_n, w_1, \ldots, w_n \in A^+ \} \}$$

$$= \llbracket E_1^{\leq n} \rrbracket (w) \ .$$