

Regular cost functions over words^{*}

Thomas Colcombet

LIAFA/CNRS/Université Paris 7, Denis Diderot, France

Long version of ICALP 09 paper, in preparation.

Abstract. We introduce and develop the notion of regular cost functions: a quantitative extension to the standard theory of regular languages.

This work is a continuation of the works on distance automata and similar models. These models of automata have been successfully used for solving the star-height problem, the finite power property, the finite substitution problem, the relative inclusion star-height problem or the boundedness problem for monadic-second order logic over words.

Our notion of regularity can be – as in the classical theory of regular languages – equivalently defined in terms of automata, of algebraic recognisability, and by a variant of the monadic second-order logic. Those equivalences are strict extensions of the corresponding classical results. The decidability of the limitedness problem of distance automata, as well as all its known variants can be deduced from our results.

1 Introduction

This paper introduces and studies a quantitative extension to the standard theory of regular languages of words. It is the only quantitative extension known to the author in which the milestone equivalence for regular languages:

accepted by automata = recognisable by monoids
= definable in monadic second-order logic = definable by regular expressions

can be faithfully extended. Hence, we introduce in this work automata (called B - and S -automata), algebraic structures (called stabilisation monoids), logics (called $\text{MSO}^{\leq N}$) and suitable regular expressions (called B - and S -regular expressions).

Related works. Though most of the objects mentioned above are new, some are very close to objects known from the literature. To this respect, the present work is the continuation of long branch of research.

A prominent question in this theory is the star-height problem. This story begins in 1963 when Eggan formulates the star-height decision problem [10]:

Input: A regular language of words L and a non-negative integer k .

^{*} Supported by the ANR project JADE: ‘Jeux et Automates, Décidabilité et Extensions’

Output: Yes, if there exists a regular expression¹ using at most k nesting of Kleene stars which defines L . No, otherwise.

Eggan proved the strictness of the hierarchy induced by k , but the decidability problem itself was quickly considered as central in language theory, and as the most difficult problem in the area.

Though some partial results were obtained by Mc Naughton, Dejean and Schützenberger [31, 9], it took twenty-five years before Hashiguchi came up with a proof of decidability spreading over four papers [14, 13, 15, 16]. This proof is notoriously difficult, and no clean exposition of it has ever been presented.

Hashiguchi used in his proof the model of *distance automata*. A distance automaton is a finite state non-deterministic automaton running over words which can count the number of occurrences of some ‘special’ states. Such an automaton associates to each word a natural number, which is the least number of occurrences of special states among all the accepting runs (or nothing if there is no accepting run over this input). The proof of Hashiguchi relies on a very difficult reduction to the following *limitedness* problem:

Input: A distance automaton.

Output: Yes, if the automaton is *limited*, i.e., if the function it computes is bounded over its domain. No, otherwise.

Hashiguchi established the decidability of this problem [13]. The notion of distance automata and its link with the tropical semiring (distance automata can be seen as automata over the tropical semiring, i.e. the semiring $(\omega + 1, \min, +)$) has been the source of many investigations [14, 17, 18, 27, 30, 35, 36, 38–40].

Despite those researches, the star-height problem itself remained not so well understood for seventeen more years. In 2005, Kirsten gave a much simpler and self-contained proof [23]. The principle is to use a reduction to the limitedness problem for a form of automata more general than distance automata, called *nested distance desert automata*. To understand this extension, let us first look again at distance automata: we can see a distance automaton as an automaton that has a counter which is incremented each time a ‘special’ state is encountered. The value attached to a word by such an automaton is the minimum over all accepting runs of the maximal value assumed by the counter. Presented like this, a nested distance desert automaton is nothing but a distance automaton in which multiple counters and reset of the counters are allowed (with a certain constraint of nesting of counters). Kirsten performed a reduction of the star-height problem to the limiteness of nested distance desert automata which is much easier than the reduction of Hashiguchi. He also proves that the limitedness problem of nested distance desert automata is decidable. For this, he generalises the proof methods developed previously by Hashiguchi, Simon and Leung for distance automata. This work closes the story of the star-height problem itself.

¹ Regular expressions are built on top of letters using the language operation of concatenation, union, and Kleene star. This problem is sometime referred to as the restricted star-height problem, while the version allowing also complementation is the generalised star-height problem, and has a very different status.

The star-height problem is the king among the problems solved using this method. But there are many other (difficult) questions that can be reduced to the limitedness of distance automata and variants. Some of the solutions to those problems paved the way to the solution of the star-height problem.

The *finite power property* takes as input a regular language L and asks whether there exists some positive integer n such that $(L + \varepsilon)^n = L^*$. It was raised by Brzozowski in 1966, and it took twelve years before being independently solved by Simon and Hashiguchi [35, 12]. This problem was the original motivation to the introduction of distance automata by Hashiguchi.

The *finite substitution problem* takes as input two regular languages L, K , and asks whether it is possible to find a finite substitution σ (i.e., a morphism mapping each letter of the alphabet of L to a finite language over the alphabet of K) such that $\sigma(L) = K$. This problem was shown decidable independently by Bala and Kirsten by a reduction to the limitedness of desert automata (a form of automata weaker than nested distance desert automata, but incomparable to distance automata), and a proof of decidability of this later problem [2, 21]. Kirsten has also proved the decidability of a new problem, the *relative inclusion star-height problem*, which simultaneously generalises the star-height problem and the finite substitution problem [20].

The *boundedness problem* is a problem of model theory. It consists in deciding if there exists a bound on the number of iterations that are necessary for the fixpoint of a logical formula to be reached. The existence of a bound means that the fixpoint can be eliminated by unfolding its definition sufficiently many times. The boundedness problem is usually parameterised by the logic chosen and by the class of models over which the formula is studied. The boundedness problem for monadic second-order formulae over the class of finite words was solved by a reduction to the limitedness problem of distance automata by Blumensath, Otto and Weyer [4].

One can also cite applications of distance automata in speech recognition [32, 33], databases [11], and image compression [19]. In the context of verification, Abdulla, Krcàl and Yi have introduced R -automata, which correspond to nested distance desert automata in which the nesting of counters is not required anymore [1]. They prove the decidability of the limitedness problem for this model of automata.

Finally, Löding and the author have also pursued this branch of researches in the direction of extended models. In [7], the *star-height problem over trees* has been solved, by a reduction to the limitedness problem of nested distance desert automata over trees. The later problem was shown decidable in the more general case of alternating automata. In [8] a similar attempt has been tried for deciding the Mostowski hierarchy of non-deterministic automata over infinite trees (the hierarchy induced by the alternation of fixpoints). The authors show that it is possible to reduce this problem to the limitedness problem for a form of automata that unifies nested distance desert automata and parity automata. The later problem is an important open question.

Bojańczyk and the author have introduced the notion of B -automata in [5], a model which resembles much (and is prior to) R -automata. The context was to show the decidability of some fragments of the logic MSOLB over infinite words, in which MSOLB is the extension of the monadic second order logic extended with the quantifier $UX.\phi$ meaning “there exists a set X as big as I want such that ϕ holds”. From the decidability results in this work, it is possible to derive every other limitedness results. However, the constructions are extremely complicated and of non-elementary complexity. Nevertheless, the new notion of S -automata was introduced, a model dual to B -automata. Recall that the semantics of distance automata and their variants can be expressed as a minimum over all runs of the maximum of the value taken by counters. The semantics of S -automata is dual: it is defined as the maximum over all runs of the minimum of the value taken by the counters at the moment of their reset. Unfortunately, it is quiet hard to compare in detail this work with all others. Indeed, since it was oriented toward the study of a logic over infinite words, the central automata are in fact ωB and ωS -automata: automata accepting languages of infinite words that have an infinitary accepting condition constraining the asymptotic behaviour of the counters along the run. This makes those automata very different: first, they accept languages instead of functions, and second, they use accepting conditions involving the asymptotic behaviours of counters, a characteristic which has no equivalent in distance automata, and which is in some sense ‘orthogonal’. For this reason, B -automata and S -automata in [5] are just intermediate objects that do not have all the properties we would like. In particular B - and S -automata in [5] are not equivalent. However, the principle of using two dual forms of automata is an important concept in the present work.

The proof methods for showing the decidability of the limitedness problem of distance automata and their variants, are also of much interest by themselves. While the original proof of Hashiguchi is quiet complex, a major advance has been achieved by Simon, using the notion of stabilisation [36]. The principle is to abstract the behaviour of the distance automaton in a monoid, and further describe the semantics of the counter using an operator of stabilisation, i.e., an operator which describes, given a element of the monoid, what would be the effect of iterating it a ‘lot of times’. This key idea was further used and refined by Simon himself, Leung, Kirsten, Abdulla, Krcál and Yi. This idea was not present in [5], and this is one explanation for the bad complexity of the constructions.

Contributions. In this paper, we revisit all the models of automata presented above, as well as the algebraic methods. The principle of our framework is to present those automata in a way comparable to the standard theory of regular languages and such that the standard results for regular languages as well as all results of decidability of limitedness presented above appear as special cases .

The effect of this presentation is that automata are not the central object anymore, but rather share equally the scene with stabilisation monoids (the algebraic presentation) and $\text{MSO}^{\leq N}$ (the logic presentation). We can even say

that the central object here is the stabilisation monoid, since it is the only model in which all proofs can be carried along in a self-contained way.

Let us describe our contributions in more details.

Cost functions. We extend the standard notion of language by the new notion of cost function. For this, we consider mappings from a set E to $\omega+1$ (in practice E is the set of finite words over some finite alphabet) and the equivalence relation \approx defined by $f \approx g$ if:

for all $X \subseteq E$, f restricted to X is bounded iff g restricted to X is bounded.

Hence two functions are equivalent if it is not possible to distinguish them using arguments of existence of bounds. A *cost function* is an equivalence class for \approx . The notion of cost functions is what we use as a quantitative extension to languages. Indeed, every language L can be identified with (the equivalence class of) the function mapping words in L to the value 0, and words outside L to ω . All our theory is presented in terms of cost functions. This means that all equivalences are considered modulo the relation \approx .

Cost automata. Our first way to define regular cost functions is to use cost automata, which comes in two flavours, B - and S -automata. Our B -automata correspond in their simple form to R -automata [1] and in their simple and hierarchical form to nested distance desert automata in [24, 25]. Those are also very close to B -automata in [5]. Following the ideas in [5], we introduce the dual variant of S -automata. Our S -automata are stronger than the S -automata in [5]. A consequence of the results in the paper is that B -automata and S -automata are equi-expressive in all their variants, an equivalence that we call the *duality theorem*.

History-determinism. Though it is not possible to determinise B - and S -automata, we prove that a new notion, called *history-determinism*, can be enforced. History-deterministic automata are non-deterministic automata that have some special semantic property that makes them resemblant to deterministic automata. We prove that history-deterministic B - and S -automata have the same expressive power as general B - and S -automata. This equivalence is required in the forthcoming extension, together with Christof Löding, of this theory to trees.

Stabilisation monoids. We introduce the new notion of stabilisation monoids. A stabilisation monoid is a finite ordered monoid together with a stabilisation operation. This stabilisation operation expresses what it means to iterate ‘a lot of times’ some element. The operator of stabilisation was introduced by Simon [35] and used also by Leung, Kirsten, Abdulla, Krcál and Yi. as a tool for analysing the behaviour of distance automata and their variants. The novelty here lies in the fact that in our case, stabilisation is now part of the definition of a stabilisation monoid. We prove that it is possible to associate unique semantics to all stabilisation monoids. Those semantics take the form of what we call compatible mappings. This key result shows that the notion of stabilisation

monoid has a ‘meaning’ independent from the existence of cost automata (in the same way monoids can be used for recognising language, independantly from the fact that it comes from a finite state automata). Introducing the notion of compatible mappings requires quiet some abstraction work, and the development of a suitable mathematical framework of topological nature.

Recognisable cost functions. We use stabilisation monoids for defining the new notion of recognisable cost functions. We show the closure of recognisable cost functions under min, max, and new operations called inf-projection and sup-projection (which are counterparts to projection in the theory of regular languages). We also prove that the relation \approx (in fact the corresponding preorder \preceq) is decidable over recognisable cost functions. This decidability result subsumes all limitedness results known from the litterature. Finally, we prove that this notion of recognisability for cost functions is equivalent to being accepted by the cost automata introduced above.

Extension of monadic second-order logic. We define the new logic $\text{MSO}^{\leq N}$ which is a strict extension to monadic second-order logic with quantitative capabilities. It is for instance possible to define the diameter of a graph in $\text{MSO}^{\leq N}$. We show that the cost functions over words definable in $\text{MSO}^{\leq N}$ coincide with the regular cost functions presented above. This equivalence is essentially the consequence of the closure properties of regular cost functions (as in the case of regular languages), and no new ideas are required here. The interest lies in the logic itself. Of course, the decision procedure for recognisable cost function entails decidability results for $\text{MSO}^{\leq N}$.

Continuations. This work has a natural continuation to finite trees, currently in preparation with Christof Löding. The extension to trees relies on automata methods, history-determinism, and games. Many results in this work to come are already interesting over finite words. One can find among them, translations of automata into their simple and hierarchical forms that have good complexity and preserve the history-determinism, efficient decision procedures for automata (all the decision procedures in this work are done over stabilisation monoids), and the equivalence with alternating cost automata.

Structure of the paper. The remaining of the paper is organised as follows.

In Section 2 we present our models of automata in their various forms, and establish results of closure and of non-closure. We also introduce the notion of cost function and of history-determinism. We finally state the main duality result, the proof of which is a consequence of the subsequent sections.

In Section 3 we present the underlying algebraic structure: stabilisation monoids. We then develop the mathematical framework of cost sequences that we use for introducing compatible mappings. We finally establish the key result of existence and uniqueness of compatible mappings.

In Section 4, we use stabilisation monoids for defining recognisable cost functions. We show various closure results for recognisable cost functions, the de-

cidability of the relation \preceq over them, and the equivalence with cost functions accepted by cost automata.

Section 5 is also rather short. We introduce in it the new logics $\text{MSO}^{\leq N}$, give some examples, and prove their equivalence with the above formalisms.

Some notations

As usual, we denote by ω the set of non-negative integers and $\omega + 1$ the set $\omega \cup \{\omega\}$. Those are ordered by $0 < 1 < \dots < \omega$. The identity mapping over ω is *id*. We use the notation \inf and \sup over subsets of $\omega + 1$ with their natural semantics. Since we assume that \inf and \sup are always computed in $\omega + 1$, we have $\inf \emptyset = \omega$ and $\sup \emptyset = 0$. We adapt the notion of characteristic function to this setting: given a set E and some $F \subseteq E$, we denote by χ_F the mapping from E to $\omega + 1$ which to $x \in F$ associates 0, and to $x \notin F$ associates ω . We call χ_F the characteristic function of F .

Given a set E , E^ω is the set of sequences of ω -length of elements in E . Such sequences will be denoted by bold letters. We fix a *finite alphabet* \mathbb{A} consisting of *letters*. The set of words over \mathbb{A} is \mathbb{A}^* . The empty word is ε . The concatenation of a word u and word v is uv . The length of word u is $|u|$. The number of occurrences of a letter a in u is $|u|_a$.

2 Cost automata

The topic of this section is to present the models of B and S -automata, and devise their properties. The key properties (in particular, the equivalence of the two forms of automata, Theorem 1) will be consequence of the two subsequent sections.

2.1 Cost automata

We present here the notions of B and S -automata. Our model subsume the models of distance automata of Haschiguchi, of desert automata, distance desert automata and nested distance desert automata of Kirsten, of B and S -automata of Bojańczyk and the author, and of R -automata of Abdulla, Krcál and Yi. Those models and the relations between them are presented at the end of this section.

A *cost automaton* (that can be either a B -automaton or an S -automaton) is a tuple $\langle Q, \mathbb{A}, In, Fin, \Gamma, \Delta \rangle$ in which Q is a finite set of *states*, \mathbb{A} is the alphabet, In and Fin are respectively the set of *initial* and *final* states, Γ is a finite set of *counters*, and $\Delta \subseteq Q \times \mathbb{A} \times (\{i, r, c\}^*)^F \times Q$ is a finite set of transitions. A cost automaton is said *deterministic* if for all $p \in Q$ and $a \in \mathbb{A}$, there is at most one transition of the form (p, a, c, q) .

Letters in $\{i, r, c\}$ are called *atomic actions*. The idea behind atomic actions is that a counter (the value of which ranges over ω) can either be incremented by

one (i), be reset to 0 (r), or be checked (c). Given a sequence of atomic actions u , it is read from left to right. At the beginning, the counter has value 0, and at each letter evolves according to the current atomic action (the value does not change when the counter is checked). The set $C(u) \subseteq \omega$ collects the values assumed by the counter each time an atomic action ‘check’ is performed.

A word in $(\{i, r, c\}^*)^{\Gamma}$ is called an *action*, and tells for each counter in Γ what sequence of atomic actions should be applied to it. Given a sequence u of such actions, $C(u)$ is the union of the sets $C(u_\iota)$ for all $\iota \in \Gamma$, in which u_ι is the projection of u over its ι ’s component. In other words, the counters in Γ evolve concurrently, and the set $C(u)$ collects every value assumed by a counter when checked.

A *partial run* σ of an automaton over a word $a_1 \dots a_n$ is defined as a sequence $q_0, a_1, c_1, q_1, \dots, q_{n-1}, a_n, c_n, q_n$ such that $(q_{i-1}, a_i, c_i, q_i) \in \Delta$ for all $i = 1 \dots n$. The partial run is called simply a *run* or an *accepting run* if furthermore q_0 is initial and q_n is final. An *initial run* is a partial run of first state initial. We set $C(\sigma) \subseteq \omega$ to be $C(c_1 c_2 \dots c_n)$. Hence, $C(\sigma)$ collects all values assumed by counters when checked during the run. One makes no distinction concerning when the value appeared, how many times it did appear, nor the counter it originates from.

The difference between B -automata and S -automata comes from their dual semantics, $\llbracket \cdot \rrbracket_B$ and $\llbracket \cdot \rrbracket_S$ respectively:

$$\begin{aligned} \text{for all } u \in \mathbb{A}^*, \quad \llbracket \mathcal{A} \rrbracket_B(u) &= \inf \{ \sup C(\sigma) : \sigma \text{ accepting run of } \mathcal{A} \text{ over } u \} , \\ \text{and,} \quad \llbracket \mathcal{A} \rrbracket_S(u) &= \sup \{ \inf C(\sigma) : \sigma \text{ accepting run of } \mathcal{A} \text{ over } u \} . \end{aligned}$$

(Recall that $\inf \emptyset = \omega$ and $\sup \emptyset = 0$.) A *B-automaton* is a cost automaton on which we always apply the semantics $\llbracket \cdot \rrbracket_B$ while an *S-automaton* is a cost automaton on which we always apply the semantics $\llbracket \cdot \rrbracket_S$. Hence, the function *accepted* by a cost automaton \mathcal{A} over alphabet \mathbb{A} is the mapping $\llbracket \mathcal{A} \rrbracket_B$ if \mathcal{A} is a B -automaton, and $\llbracket \mathcal{A} \rrbracket_S$ if it is an S -automaton.

An alternative way of describing the semantics of cost automata is by using the notion of n -run. An n -run of a B -automaton \mathcal{A} for some $n \in \omega$ is a (accepting) run σ such that $\sup C(\sigma) \leq n$. Equivalently, whenever a counter is checked during the run σ , it is required to have value at most n .² An n -run of an S -automaton \mathcal{A} for some $n \in \omega$ is a run σ such that $\inf C(\sigma) \geq n$. Equivalently, whenever a counter is checked during the run σ , it is required to have value at least n . We then have for all word u and all $n \in \omega$:

$$\begin{aligned} \llbracket \mathcal{A} \rrbracket_B(u) &= \inf \{ n : \text{there exists an } n\text{-run of } \mathcal{A} \text{ over } u \} , \\ \text{and} \quad \llbracket \mathcal{A} \rrbracket_S(u) &= \sup \{ n : \text{there exists an } n\text{-run of } \mathcal{A} \text{ over } u \} . \end{aligned}$$

(There is a slight ambiguity here since the definition of an n -run depends also on whether we consider the B -semantics or the S -semantics. In practice, this will always be clear from the context.)

² The original definition of the semantics used the atomic action ‘check’ for ‘outputting’ the current value. When using n -runs one sees the atomic action ‘check’ as ‘checking’ that the value is at most n . This explains the name ‘check’ rather than ‘output’.

This way of describing the functions accepted by cost automata has several advantages. First of all, it is easier to use: proving that a run σ is an n -run requires half the work necessary for proving that $\sup C(\sigma) = n$ (or $\inf C(\sigma) = n$). Second, it is possible that $\inf C(\sigma)$ be ω (if no counter is checked), hence, the original definition of $\llbracket \cdot \rrbracket_S$ involves the computation of the infimum of subsets of $\omega + 1$, while the use of n -runs involves only infimum of subsets of ω . This avoids to separate cases in proofs. Finally, at least in the case of B -automata, one can see the ‘check’ as an action which requires a certain quantity of ressource for being performed, namely, the value of the counter at this point. Hence, one can see an n -run as a run which can be performed under ressource limit n . In this perspective, the value $\llbracket \mathcal{A} \rrbracket_B(u)$ is nothing but the quantity of ressource necessary for processing the input u .

Remark 1. Those definitions are tightly related to the semantics of non-deterministic finite automata accepting languages. Indeed, imagine that \mathcal{A} is a non-deterministic finite automaton in the standard sense, accepting the language L , then it can be seen as a cost automaton without counter.

Imagine that we see \mathcal{A} as a B -automaton, and consider a word u outside L . By definition, there exist no run over u . As a consequence $\llbracket \mathcal{A} \rrbracket_B(u) = \inf \emptyset = \omega$. Consider now a word u in L , then there exists at least one run σ over u , and since no counter can be checked, all run σ satisfy $C(\sigma) = \emptyset$. We obtain $\llbracket \mathcal{A} \rrbracket_B(u) = \inf\{\sup \emptyset\} = 0$. For short, we have:

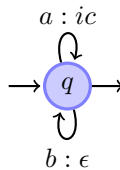
$$\llbracket \mathcal{A} \rrbracket_B = \chi_L .$$

(Recall that χ_L is the *characteristic function* which to each word u associates 0 if $u \in L$, and ω otherwise.)

Similarly, if we consider \mathcal{A} as an S -automaton, then \inf and \sup have their roles exchanged, and we get

$$\llbracket \mathcal{A} \rrbracket_S = \chi_{(\mathbb{A}^* \setminus L)} .$$

Example 1. Consider the following B -automaton:



It has only one counter, and each transition of the form (p, a, c, q) is depicted as an arrow from p to q labeled by ‘ $a : w$ ’ for a a letter, and w an action. Initial states are identified by ingoing edges that have no origin state, while final states are similarly identified by outgoing edges.

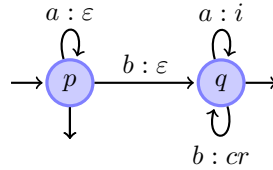
When reading a word over $\{a, b\}$, this automaton start with the value 0 of the counter, and each time a letter a is encountered, the counter is incremented and immediatly checked. When a letter b is met, no action is performed. The

result is that, if we call σ_u the only run (the automaton is deterministic) over a word $u \in \{a, b\}$, we have:

$$C(\sigma) = \{1, 2, \dots, |u|_a\} .$$

As a consequence $\llbracket \mathcal{A} \rrbracket_B(u) = \sup C(\sigma_u) = |u|_a$. This automaton counts the number of occurrences of the letter a .

Example 2. Consider now the following deterministic one-counter S -automaton:

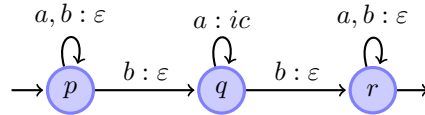


While reading a word u over $\{a, b\}$, this automaton first skips the first occurrences of the letter a , until it reaches the first occurrence of b . Then it increments the counter as long as a -letters are met, until it reaches a letter b , and at this moment the counter is checked and immediately reset. In other words, if we call σ the sole run over u , $C(\sigma)$ collects the length of segments of a 's surrounded by b 's, i.e.:

$$C(\sigma) = \{n : u = vba^n bw, \text{ for some } vw \in \{a, b\}^*\} .$$

Hence $\llbracket \mathcal{A} \rrbracket_S(u)$ is the minimal length of a segment of consecutive letter a 's surrounded by b 's. Notice also the special case when the word u has less than two occurrences of the letter b . In this situation, we have a run in which the counter is never checked, entailing $\llbracket \mathcal{A} \rrbracket_S(u) = \omega$.

Example 3. Our last example is a one-counter B -automaton which is not deterministic, and makes use of this non-determinism:



A run σ of this automaton over a word $u \in \{a, b\}^*$ can be described as follows. It first skips a certain number of letters while staying in state p . At some point it jumps to state q while reading a letter b . It stays in state q as long as the input letter is a , incrementing and checking the counter at each step. It exits state q by jumping to state r while reading a letter b , and then skips the end of the word. Notice there is no (accepting) run over words that have less than two occurrences of b 's. This means in this case that $\llbracket \mathcal{A} \rrbracket_B(u) = \omega$. Otherwise, if u contains more than two occurrences of b 's, each run σ is characterised by the two occurrences $i < j$ of b 's that makes it enter and leave respectively the state q . Then we have:

$$C(\sigma) = \{1, 2, \dots, j - i - 1\} .$$

Hence, \mathcal{A} computes the minimal length of a segment of letters which is surrounded by occurrences of b .

We can conclude that this automaton computes the same function as the one of Example 2.

We will encounter along the paper special models of automata. Those models happen to be equi-expressive with the general model, and are usually easier to handle. We present now.

A B -automaton of counters Γ is *simple* if the actions it uses belong to $\{\varepsilon, ic, r\}^\Gamma$. The above examples 1 and 3 are of this form. An S -automaton of counters Γ is *simple* if the actions that can be performed on counters belong to $\{\varepsilon, i, r, cr\}^\Gamma$. Example 2 is of this form.

All the examples of cost automata in this paper are in fact simple cost automata. One reason for that is that the general case can be – in a sense to be defined – easily reduced to it. We show this in Section 2.4.

A cost automaton is *hierarchical* of counters Γ if $\Gamma = \{1, \dots, |\Gamma|\}$, and for all action $a \in (\{i, c, r\}^*)^\Gamma$, if $a(k) \neq \varepsilon$ for some $k = 1, \dots, |\Gamma|$, then $a(k+1), \dots, a(|\Gamma|) \in \{r, cr\}$. Once more, those automata, even in their simple form, happen to be also equivalent to the general model. The idea behind this definition is that there is a form of nesting of counters: as soon as something is done on a counter, all counters which are higher have to be reset. We will make few use of this notion in this paper.

Link with other formalisms As mentioned in the introduction, the formalisms introduced here is linked with previous works. Let us briefly present those models in our formalism.

The *distance automata* of Hashiguchi are simple automata with one counters and no reset [13]. I.e., actions belong simply to $\{\varepsilon, ic\}$. The actions label states rather than transitions in our model.

The *desert automata* of Kirsten and Bala correspond to simple B-automata with one counter and actions belonging to $\{r, ic\}$ [2, 3, 21, 24]. The idea behind the name ‘desert’ is that the automaton has a tank of water of a certain capacity, which is consumed as long as one stays in the desert (corresponding to the action ic). However, as soon as an oasis (corresponding to the action r) is encountered, the tank of water can be replenished. The value attached to a word by such an automaton is the capacity of the tank required for processing the word.

The *distance desert automata* where introduced by Kirsten for solving the star-height one problem[22]. Distance desert automata are hierarchical simple B-automata with two counters, one acting as a distance counter, while the other is used as a desert counter. Formally, the counters are $\{1, 2\}$, and the actions are $\{(\varepsilon, ic), (\varepsilon, r), (ic, r)\}$ (the actions over two counters are presented as ordered pairs). Those automata are tailored for solving the star-height one problem.

The *nested distance desert automata* of Kirsten are a generalisation of the distance desert automata with an arbitrary number of counters. Those corresponds to hierarchical simple B-automata [23].

In [1], hierarchical simple B-automata were introduced under the name *R-automata*.

In [5, 6], a form of ‘B-automata’ was introduced. Those automata correspond to B-automata of the present work, but such that the counters could use actions in $\{\varepsilon, i, cr\}^F$, and that furthermore, every counter needs to be reset at least one for making a run accepting. The dual form of automata, S-automata was also introduced, with the same set of possible of actions, and the same constraint on the occurrence of resets. Those automata are to some extent weird and uneasy to handle. This can be explained by the fact that those objects were introduced for studying ωB and ωS -automata, the later being automata running over infinite words. A lot of bad phenomena linked to this definition of *B* and *S*-automata was eliminated because they were used in the context of an infinite computation (for instance, the constraint that counters be reset in a run was used when repeated infinitely often for validating a form of Büchi condition). The very important contribution of this work is the introduction of two dual forms of automata. This idea is of course absolutely essential in the present work.

2.2 Elementary constructions

In this section we present some elementary results about cost automata. Those constructions are for most of them straightforward adaptations of standard constructions for automata for languages.

Since the semantics of cost automata are defined in terms of infimum and supremum, we get in a straightforward manner some closure properties involving min and max computations.

Proposition 1. *The function accepted by [simple] [hierarchical] B-automata are closed under min. The function accepted by [simple] B-automata are closed under max. The function accepted by [simple] [hierarchical] S-automata are closed under max. The function accepted by [simple] S-automata are closed under min.*

Proof. The reader will recognise the constructions for the intersection and union of languages described by non-deterministic automata. Let us consider two cost automata $\mathcal{A} = \langle Q, \mathbb{A}, In, Fin, \Gamma, \Delta \rangle$ and $\mathcal{A}' = \langle Q', \mathbb{A}, In', Fin', \Gamma', \Delta' \rangle$.

Closure under min of [simple] [hierarchical] B-automata. The construction consists in simply taking the disjoint union of the two automata. We assume without loss of generality that Q and Q' are disjoint. We construct the automaton $\mathcal{A}_+ = \langle Q \cup Q', \mathbb{A}, In \cup In', Fin \cup Fin', \Gamma \cup \Gamma', \Delta_+ \rangle$ in which:

$$\Delta_+ = \{(p, a, c \uparrow \Gamma \cup \Gamma', q) : (p, a, c, q) \in \Delta \cup \Delta'\} ,$$

where $c \uparrow \Gamma \cup \Gamma'$ coincide with c on its domain, and maps every other counter in $\Gamma \cup \Gamma'$ to ε . It is clear that each run σ of \mathcal{A} (or \mathcal{A}') can directly be translated into a run σ_+ of \mathcal{A}_+ over the same word such that $C(\sigma) = C(\sigma_+)$. Conversely, every run σ_+ of \mathcal{A}_+ can be translated in a run σ either of \mathcal{A} or of \mathcal{A}' over the same word. Hence, we directly get for all word $u \in \mathbb{A}^*$:

$$\llbracket \mathcal{A}_+ \rrbracket_B(u) = \min(\llbracket \mathcal{A} \rrbracket_B(u), \llbracket \mathcal{A}' \rrbracket_B(u)) .$$

Furthermore, this construction preserves the simple and hierarchical nature of the automaton.

Closure under max of [simple] [hierarchical] S-automata. One uses the exact same construction as for the min of B -automata. This time we obtain for all word $u \in \mathbb{A}^*$:

$$\llbracket \mathcal{A}_+ \rrbracket_S(u) = \max(\llbracket \mathcal{A} \rrbracket_S(u), \llbracket \mathcal{A}' \rrbracket_S(u)) .$$

Closure under max of [simple] B-automata. This time we perform the product of the automata. We assume now without loss of generality that the sets of counters Γ and Γ' are disjoint. We construct the automaton $\mathcal{A}_\times = \langle Q \times Q', \mathbb{A}, In \times In', Fin \times Fin', \Gamma \cup \Gamma', \Delta_\times \rangle$ in which:

$$\Delta_\times = \{((p, p'), a, cc', (q, q')) : (p, a, c, q) \in \Delta, (p', a, c', q') \in \Delta'\} ,$$

where cc' maps each counter ι of Γ to $c(\iota)$ and each counter ι of Γ' to $c'(\iota)$. It is clear that given a run σ of \mathcal{A} and a run σ' of \mathcal{A}' over the same word u , we can construct by product a run σ_\times of \mathcal{A}_\times over u such that $C(\sigma_\times) = C(\sigma) \cup C(\sigma')$. Conversely, each run σ_\times of \mathcal{A}_\times over a word u can be decomposed into a word of \mathcal{A} over u and a word of \mathcal{A}' over u such that $C(\sigma_\times) = C(\sigma) \cup C(\sigma')$. Hence, we directly get for all word $u \in \mathbb{A}^*$:

$$\llbracket \mathcal{A}_\times \rrbracket_B(u) = \max(\llbracket \mathcal{A} \rrbracket_B(u), \llbracket \mathcal{A}' \rrbracket_B(u)) .$$

Furthermore, this construction preserves the simplicity of the automaton.

Closure under min of [simple] S-automata. One uses the exact same construction as for the max of B -automata. This time we obtain for all word $u \in \mathbb{A}^*$:

$$\llbracket \mathcal{A}_\times \rrbracket_S(u) = \min(\llbracket \mathcal{A} \rrbracket_S(u), \llbracket \mathcal{A}' \rrbracket_S(u)) .$$

□

In the previous proposition, we have just adapted the constructions for union and for intersection for non-deterministic finite automata accepting languages. There is another operation which is very natural for non-deterministic automata: the projection. Let us first describe what are the corresponding operations for cost automata.

Given a mapping f from \mathbb{A}^* to $\omega + 1$ and a mapping h from \mathbb{A} to \mathbb{B} that we extend into a morphism from \mathbb{A}^* to \mathbb{B}^* (\mathbb{B} being another alphabet) the *inf-projection* of f with respect to h is the mapping $f_{\text{inf},h}$ from \mathbb{B}^* to $\omega + 1$ defined for all $v \in \mathbb{B}^*$ by:

$$f_{\text{inf},h}(v) = \inf\{f(u) : h(u) = v\} .$$

Similarly, the *sup-projection* of f with respect to h is the mapping $f_{\text{sup},h}$ from \mathbb{B}^* to $\omega + 1$ defined for all $v \in \mathbb{B}^*$ by:

$$f_{\text{sup},h}(v) = \sup\{f(u) : h(u) = v\} .$$

By simply adapting the standard construction for projection of non-deterministic automata, we get:

Proposition 2. *The mappings accepted by [simple] [hierarchical] B -automata are closed under inf-projection. The mappings accepted by [simple] [hierarchical] S -automata are closed under sup-projection.*

Proof. Once more the construction precisely mimics the standard construction, for projection this time. Consider an automaton $\mathcal{A} = \langle Q, \mathbb{A}, In, Fin, \Gamma, \Delta \rangle$ and a mapping h from \mathbb{A} to another alphabet \mathbb{B} . We extend it into a morphism from \mathbb{A}^* to \mathbb{B}^* as above.

Closure under inf-projection of B -automata. Construct the automaton $\mathcal{A}_h = \langle Q, \mathbb{B}, In, Fin, \Gamma, \Delta_h \rangle$ in which Δ_h is defined by:

$$\Delta_h = \{(p, h(a), c, q) : (p, a, c, q) \in \Delta\} .$$

It is clear that each run σ of \mathcal{A} over a word $u \in \mathbb{A}^*$ can be turned into a run σ_h of \mathcal{A}_h over the word $h(u)$ such that $C(\sigma) = C(\sigma_h)$. Conversely, for each run σ_h of \mathcal{A}_h over a word $v \in \mathbb{B}^*$, there exists a run σ of \mathcal{A} over a word $u \in \mathbb{A}^*$ such that $h(u) = v$ and $C(\sigma_h) = C(\sigma)$. Hence we have for all $u \in \mathbb{B}^*$:

$$\llbracket \mathcal{A}_h \rrbracket_B(u) = \llbracket \mathcal{A} \rrbracket_{B_{\text{inf},h}}(u) .$$

Closure under sup-projection of S -automata. We use the same construction as above, and obtain this time for all $u \in \mathbb{B}^*$:

$$\llbracket \mathcal{A}_h \rrbracket_S(u) = \llbracket \mathcal{A} \rrbracket_{S_{\text{sup},h}}(u) .$$

□

Some properties of conversion between standard automata and cost automata are also useful. We have already seen in Remark 1 that for every regular language L , it was possible to construct a B -automaton and an S -automaton that both accept the function χ_L . We see here some form of converse.

Proposition 3. *For all B -automaton (resp. S -automaton) accepting a function f , the set:*

$$\text{support}(f) = \{u : f(u) < \omega\}$$

is regular.

Proof. If f is described by a B -automaton, then a word belongs to the support of f iff there exists a run of the automaton over it. Hence, it is sufficient to take the same automaton and get rid of all the information concerning the counters in order to obtain an automaton for its support.

If f is described by an S -automaton, then a word does not belong to the support of f iff there is a run of the automaton over this word which never performs a check action. Hence, if we take the same automaton and, (a) get rid of all the transitions that perform a check of some counter, and then (b) remove all information concerning the counters, we obtain an automaton for the complement of the support. A complementation of the resulting automaton completes the construction. □

Proposition 4. *For all B -automaton (resp. S -automaton) accepting a function f , and all $n \in \omega$, the language:*

$$\{u : f(u) \leq n\},$$

is regular.

Proof. We do not perform the construction. The only idea behind this is that it is possible to store a sufficient amount of information in an automaton for counting ‘up to value n ’. Since it is not allowed in cost automata to decrease the value of counters, once a counter has taken a value beyond n , the only way we can get back to a value at most n is by using a reset. Hence, it is sufficient to use an abstract value n^+ that will represent any possible value beyond n . \square

Proposition 5. *Every function accepted by a deterministic S -automaton is also accepted by a B -automaton. Every function accepted by a deterministic B -automaton is also accepted by an S -automaton.*

Proof. Let us informally describe the construction. Consider an S -automaton \mathcal{A} , the new B -automaton \mathcal{A}' we construct simulates precisely \mathcal{A} , dropping all check actions encountered, but a single one which the automaton guesses using its non-determinism (in practice this require to double the size of the automaton for remembering the bit of information: ‘the guessed check action has already been encountered’).

Consider now a word u . If its value is ω , this means that no check occur during the run, and hence the automaton \mathcal{A}' has no run over this input. Otherwise, if the value is n , n is the least value assumed by a counter when checked. This is of course what the B -automaton \mathcal{A}' computes.

The construction from B -automata to S -automata is identical. \square

2.3 Cost functions

In the previous section, we saw that the closure and equivalence properties were quite limited. We recover good properties by considering the functions modulo a certain equivalence (\approx). The subject of this section is the description of this equivalence.

A *correction function* is a non-decreasing mapping from ω to ω . From now, the symbols $\alpha, \alpha' \dots$ implicitly designate correction functions. Given x, y in $\omega+1$, $x \preceq_\alpha y$ holds if $x \leq \bar{\alpha}(y)$ in which $\bar{\alpha}$ is the extension of α with $\bar{\alpha}(\omega) = \omega$. For every set E , \preceq_α is extended to $(\omega+1)^E$ in a natural way by $f \preceq_\alpha g$ if $f(x) \preceq_\alpha g(x)$ for all $x \in E$, or equivalently $f \leq \bar{\alpha} \circ g$. Intuitively, f is dominated by g after it has been ‘stretched’ by α . One also writes $f \approx_\alpha g$ if $f \preceq_\alpha g$ and $g \preceq_\alpha f$. Finally, one writes $f \preceq g$ (resp. $f \approx g$) if $f \preceq_\alpha g$ (resp. $f \approx_\alpha g$) for some α . A *cost function* (over a set E) is an equivalence class of \approx (i.e., a set of mappings from E to $\omega+1$).

Some elementary properties of \preceq_α are:

Fact 1 *If $\alpha \leq \alpha'$ and $f \preceq_\alpha g$, then $f \preceq_{\alpha'} g$. If $f \preceq_\alpha g \preceq_\alpha h$, then $f \preceq_{\alpha \circ \alpha} h$.*

Example 4. Over $\omega \times \omega$, maximum and sum are equivalent for the doubling correction function (for short, $(\max) \approx_{\times 2} (+)$). Indeed, for all $x, y \in \omega$,

$$\max(x, y) \leq x + y \leq 2 \times \max(x, y) .$$

Our second example concerns mappings from sequence of words to ω . Given words $u_1, \dots, u_k \in \{a, b\}^*$, we have

$$|u_1 \dots u_k|_a \approx_\alpha \max(|K|, \max_{i=1 \dots k} |u_i|_a) \quad \text{where } K = \{i \in \{1, \dots, k\} : |u_i|_a \geq 1\}$$

in which α is the squaring function and $\alpha(m) = m^2$. Indeed, for one direction we just have to remark:

$$\max(|K|, \max_{i=1 \dots k} |u_i|_a) \leq |u_1 \dots u_k|_a,$$

and for the other direction we use:

$$|u_1 \dots u_k|_a \leq \sum_{i \in K} |u_i|_a \leq (\max(|K|, \max_{i=1 \dots k} |u_i|_a))^2 .$$

The relation \preceq has other characterisations:

Proposition 6. *For all f, g from E to $\omega + 1$, the following items are equivalent:*

- $f \preceq g$,
- $\forall n \in \omega. \exists m \in \omega. \forall x \in E. g(x) \leq n \rightarrow f(x) \leq m$, and;
- for all $X \subseteq E$, $g|_X$ is bounded implies $f|_X$ is bounded.

Proof. From 1 to 2. Let us assume $f \preceq g$, i.e., $f \preceq_\alpha g$ for some α . Let n be a non-negative integer, and $m = \alpha(n)$. We have for all x , that $g(x) \leq n$ implies $f(x) \leq (\alpha \circ g)(x) \leq \alpha(n) = m$, thus establishing the second statement.

From 2 to 3. Let $X \subseteq E$ be such that $g|_X$ be bounded. Let n be a bound of g over X . Item 2 states the existence of m such that $\forall x \in E. g(x) \leq n \rightarrow f(x) \leq m$. In particular, for all $x \in E$, we have $g(x) \leq n$ by choice of n , and hence $f(x) \leq m$. Hence $f|_X$ is bounded by m .

From 3 to 1. Let $n \in \omega$, consider the set $X_n = \{x : g(x) \leq n\}$. The mapping g is bounded over X_n (by n), and hence by Item 3, f is also bounded. We set $\alpha(n) = \sup f(X_n)$. Let now $x \in X$. If $g(x) < \omega$, we have that $x \in X_{g(x)}$ by definition of the X 's. Hence $f(x) \leq \sup f(X_{g(x)}) = \alpha(g(x))$. Otherwise $g(x) = \omega$, and we have $f(x) \leq \bar{\alpha}(g(x)) = \omega$. Hence $f \preceq_\alpha g$. \square

The last characterisation shows that the relation \approx is an equivalence relation that preserves the existence of bounds. Indeed, all this theory can be seen as an automata theoretic method for proving the existence/non-existence of bounds.

Cost functions over some set E ordered by \preceq form a lattice. Let us show how this lattice refines the lattice of subsets of E ordered by inclusion. The following elementary fact shows that we can identify a subset of E with the cost function of its characteristic function (remind that given a subset $X \subseteq E$, one denotes by χ_X its characteristic mapping defined by $\chi_X(x) = 0$ if $x \in X$, and ω otherwise):

Fact 2 For all $X, Y \subseteq E$, $\chi_X \preceq \chi_Y$ iff $Y \subseteq X$.

To this respect, the lattice of cost functions is a refinement of the lattice of subsets of E equipped with the superset ordering. Let us show that this refinement is strict. Indeed, there is only one language L such that χ_L does not have ω in its range, namely $L = E$, however, we will show in Proposition 7 that, as soon as E is infinite, there are uncountably many cost functions which have this property of not using the value ω .

Proposition 7. *If E is infinite, then there exist at least continuum many different cost functions from E to ω .*

Proof. Without loss of generality, let us assume that $E = \omega \setminus \{0\}$. Let p_0, p_1, \dots be the sequence of prime numbers. Every $n \in E$ is decomposed in a unique way as $p_1^{n_1} p_2^{n_2} \dots$ in which all n_i 's are null but finitely many (with an obvious meaning of the infinite product). For all $I \subseteq \omega$, one defines the function f_I from $\omega \setminus \{0\}$ to ω for all $n \in \omega \setminus \{0\}$ by:

$$f_I(n) = \max\{n_i : i \in I, n = p_1^{n_1} p_2^{n_2} \dots\}.$$

Consider now two different sets $I, J \subseteq \omega$. This means—up to a possible exchange of the roles of I and J , that there exists $i \in I \setminus J$. Consider now the set $X = \{p_i^k : k \in \omega\}$. Then, by construction, $f_I(p_i^k) = k$ and hence f_I is not bounded over X . However, $f_J(p_i^k) = 0$ and hence f_J is bounded over X . It follows by Proposition 6 that f_I and f_J are not equivalent for \approx . We can finally conclude that—since there exist continuum many subsets of ω —there is at least continuum many cost functions over E . \square

Of course, we will be applying this notion of cost functions to the B and S -automata defined above. Since our automata are defined in terms of infimum and supremum, it is convenient to use the following fact for establishing \preceq_α -relationship:

Fact 3 For all $E, F \subseteq \omega$, and all correction function α , then;

$$\begin{aligned} \inf E \preceq_\alpha \inf F & \quad \text{iff} \quad \forall n \in F. \exists m \in E. m \leq \alpha(n), \\ \sup E \preceq_\alpha \sup F & \quad \text{iff} \quad \forall m \in E. \exists n \in F. m \leq \alpha(n), \\ \text{and} \quad \sup E \preceq_\alpha \inf F & \quad \text{iff} \quad \forall m \in E. \forall n \in F. m \leq \alpha(n). \end{aligned}$$

2.4 Reduction to simple automata

In this section, we prove the following proposition:

Proposition 8. *The functions accepted by B -automata are equivalent (modulo \approx) to functions accepted by simple B -automata. The functions accepted by S -automata are equivalent (modulo \approx) to functions accepted by simple S -automata.*

The essential idea is that it is useless to use non-simple automata. In practice, all automata constructions in the paper (the important duality theorem included, see below) do preserve the simpleness of automata. Hence, there is no reason to consider non-simple automata. Since this construction is of few importance, we just give a rough description of it.

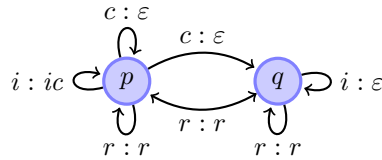
For clarity, we will work in this proof with automata that have only a single counter. Let us prove that this is not a restriction. Indeed, consider for instance a B-automaton with k -counter accepting a function f . It can be seen as computing the inf-projection of a deterministic B-automaton with k -counters (that would read the alphabet of transitions, exactly as a standard non-deterministic automaton can be seen as accepting the projection of the language accepted by a deterministic automaton). The latter can be seen as accepting the maximum of the functions computed by k deterministic B -automata, each of them using only one counter. Assume now that each such deterministic B -automaton is equivalent (modulo \approx) to a non-deterministic simple B -automaton with one counter. Then using Proposition 1, one can compute the maximum of their accepted functions by a simple B -automaton, and then by inf-projection, using Proposition 2, it is possible to obtain a simple B -automaton accepting a function equivalent to f . The same argument works for S -automata (just replace maximum by minimum, and infimum by supremum).

We will use simple* automata as an intermediate object. Remind that a simple B-automaton is an automaton which uses only actions of simple form, namely (in the context of one counter), $S_B = \{\varepsilon, ic, r\}$ for B-automata and $S_S = \{\varepsilon, i, cr, r\}$ for S-automata. A *simple** B-automaton is an automaton which uses actions in S_B^* . A *simple** S-automaton is an automaton which uses actions in S_S^* . The proof goes by first showing that every cost automaton is equivalent to a simple* cost automaton (Lemma 4), and then that every simple* cost automaton can be transformed into a simple cost automaton (Lemma 5).

Given a sequence $u \in \{i, c, r\}^*$, we will denote by $Cost_B(u)$ the value $\sup C(u)$, and by $Cost_S(u)$ the value $\sup C(u)$.

Lemma 4. *Every B-automaton (resp. S-automaton) with one counter can be transformed into an equivalent simple* B-automaton (resp. simple* S-automaton) with one counter and ε -transitions which uses only actions in $\{\varepsilon, i, c, r\}$.*

Proof. Case of B-automata: Consider the following simple one counter B-automaton \mathcal{C}_B (p is initial, and both states are final):

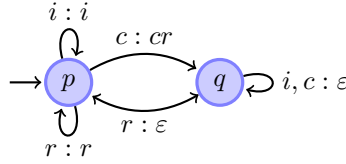


One can see it as a transducer reading a pair of words, one over $\{i, c, r\}^*$ (the input), the other over S_B^* (the output). One can convince oneself that for every pair (u, v) accepted by this transducer, $Cost_B(u) \leq Cost_B(v)$. Conversely

for every input u , there exists v such that $Cost_B(u) = Cost_B(v)$ and (u, v) is accepted. Overall $[[\mathcal{C}_B]]_B = Cost_B$.

Consider now a B-automaton with one counter \mathcal{A} over the alphabet \mathbb{A} . We can see it as a (non-deterministic finite state) transducer from \mathbb{A}^* to $\{i, c, r\}^*$. Using the standard construction, we can cascade \mathcal{A} and \mathcal{C}_B and obtain a non-deterministic finite state transducer from \mathbb{A}^* to S_B^* . If we inspect more closely this construction, one can remark that this transducer can be made syntactically a simple* one counter B-automaton. Using the equality $[[\mathcal{C}_B]]_B = Cost_B$, one easily shows that it accepts the function $[[\mathcal{A}]]_B$.

Case of S-automata: We use the exact same technique. This time, we consider the following one counter simple* S-automaton \mathcal{C}_S :



Once more, this automaton is such that $[[\mathcal{C}_S]]_S = Cost_S$. We can conclude using the same cascade product technique as for B-automata. \square

We need now to transform simple* automata into simple ones. This is achieved by the following lemma.

Lemma 5. *Every one counter simple* B-automaton (resp. S-automaton) can be turned into an equivalent simple one counter B-automaton (resp. S-automaton).*

Proof. Case of B-automata: Fix a simple* one counter B-automaton \mathcal{A} . The idea of the construction is to be able to contract transitions: we would like to replace each transition of the form (p, a, u, q) of \mathcal{A} (where $u \in S_B^*$), with a ‘simple’ transition of the form (p, a, u', q) where $u' \in S_B$. This reduction is obtained by describing a product ‘ \cdot ’ over S_B . Given $u, v \in S$, their product $u \cdot v$ is defined using the following table:

u/v	ε	ic	r
ε	ε	ic	r
ic	ic	ic	r
r	r	r	r

This product satisfies that there exists α such that for all $u, v \in S_B$ and all $w_0, \dots, w_k \in \{i, c, r^*\}$:

$$Cost_B(w_0 u v w_1 \dots w_{k-1} u v w_k) \approx_\alpha Cost_B(w_0 (u \cdot v) w_1 \dots w_{k-1} (u \cdot v) w_k) .$$

The proof is by case distinction depending on u, v . The consequence is that if we take a transition of the form (p, a, uvw, q) (for some $w \in S_B^*$) in \mathcal{A} and replace it with the transition $(p, a, (u \cdot v)w, q)$, we obtain a new automaton \mathcal{A}' such that the runs of \mathcal{A} can be identified with the runs of \mathcal{A}' . Furthermore, this translation of runs transform each run into a run of \approx_α -equivalent cost. Hence \mathcal{A}'

accepts a function \approx -equivalent to the function accepted by \mathcal{A} . By iterating this simplification process, we can turn our original automaton into an equivalent simple one.

Case of S-automata: Fix a simple* one counter S-automaton \mathcal{A} . We use the same technique. This time the product $u \cdot v$ for $u, v \in S_S$ is defined by:

u/v	ε	i	cr	r
ε	ε	i	cr	r
i	i	i	cr	r
cr	cr	cr	\perp	r
r	r	cr	\perp	cr

This product satisfies that there exists α such that for all $w_0, \dots, w_k \in \{i, c, r\}^*$,

$$Cost_S(w_0 u v w_1 \dots w_{k-1} u v w_k) \approx_\alpha Cost_S(w_0 (u \cdot v) w_1 \dots w_{k-1} (u \cdot v) w_k),$$

in which $Cost_S$ is supposed to be 0 for all word containing the symbol \perp (this is what happens when doing a reset immediately followed by a check). The transformation of the S-automaton \mathcal{A} into an equivalent simple S-automaton \mathcal{A}' is done exactly as in the case of B-automata, with the slight difference that newly introduced transitions that would use the symbol \perp are simply erased. \square

This concludes the proof of Proposition 8.

2.5 History-determinism

In general B and S -automata cannot be determinised (even modulo \approx). We consider here automata which possess a weaker property: history-determinism (note that this notion is meaningful even for other kinds of non-deterministic automata). Informally, a non-deterministic automaton is history-deterministic if it possible to choose deterministically the run while accepting an equivalent function. The subtlety comes from the fact that cost automata do not have a sufficient memory for ‘implementing’ this deterministic choice. History-determinism can be seen as a semantic notion of determinism as opposed to the standard notion that we can refer to as state-determinism³. This notion is required for the extension of the theory to trees.

Formally, let us fix ourselves a cost automaton (either B or S) with unique initial state $\mathcal{A} = \langle Q, \mathbb{A}, \{q_0\}, Fin, \Gamma, \Delta \rangle$. A *translation strategy*⁴ for \mathcal{A} is a mapping δ from $\mathbb{A}^* \times \mathbb{A}$ to Δ which tells deterministically how to construct a partial run of \mathcal{A} over a word. One defines the *initial run of \mathcal{A} over the word u driven by δ* inductively as follows: if $u = \varepsilon$, the partial run is q_0 . If u is of the form va , the run is the partial run of \mathcal{A} over v driven by δ prolonged with the transition $\delta(v, a)$ (if this procedure does not provide a valid partial run over u , then

³ In state-determinism, given the current state and a letter, there is only one possible transition, while in history-determinism, given the prefix of word seen so far (the history), and a letter, it is possible to uniquely choose one transition.

⁴ The name comes from the game theoretic part which is not developed here.

there is no partial run driven by δ over this entry). By construction, this partial run starts in an initial state. It may, or may not, end with a final state. If it is the case, it is called the run driven by δ over the entry.

A cost automaton is history deterministic if for all $n \in \omega$ there exists a translation strategy δ_n such that for all word u , the run driven by δ_n over u , if it exists, is an n -run, and;

- If automaton is a B -automaton then: for all n -run of \mathcal{A} over some word u , there is a run of \mathcal{A} driven by $\delta_{\alpha(n)}$ over u ,
- If the automaton is an S -automaton: for all $\alpha(n)$ -run of \mathcal{A} over some word u , there is a run of \mathcal{A} driven by δ_n over u ,

In other words, for all run, there is a driven run which is as good (up to correction α).

Another way to say the same thing is as follows. Let δ_n for all $n \in \omega$ be as above. If the automaton is a B -automaton, define for all word u ,

$$\llbracket \mathcal{A} \rrbracket_B^\delta(u) = \inf\{n : \text{there exists a } (n)\text{-run of } \mathcal{A} \text{ driven by } \delta_n \text{ over } u\} .$$

Hence, this is the same infimum definition as in the definition of $\llbracket \cdot \rrbracket_B$, but this time the infimum does not range over all runs, but solely over runs driven by δ . By definition, we have $\llbracket \mathcal{A} \rrbracket_B \leq \llbracket \mathcal{A} \rrbracket_B^\delta$. The automaton is history deterministic iff a form of converse holds, namely $\llbracket \mathcal{A} \rrbracket_B^\delta \preceq_\alpha \llbracket \mathcal{A} \rrbracket_B$.

Similarly, if the automaton is an S -automaton, one defines for all word u ,

$$\llbracket \mathcal{A} \rrbracket_S^\delta(u) = \sup\{n : \text{there exists a } (n)\text{-run of } \mathcal{A} \text{ driven by } \delta_n \text{ over } u\} .$$

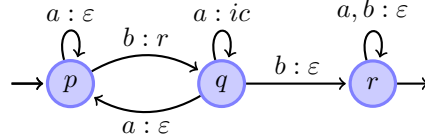
Once more, we have by definition $\llbracket \mathcal{A} \rrbracket_S^\delta \leq \llbracket \mathcal{A} \rrbracket_S$. The automaton is history deterministic iff a form of converse holds, namely $\llbracket \mathcal{A} \rrbracket_S \preceq_\alpha \llbracket \mathcal{A} \rrbracket_S^\delta$.

Example 5. The automaton of Example 3 is not history-deterministic. The reason for that is that, when in state p and while reading the letter b , the automaton has to make a choice: either stay in p , or go to q . For the automaton, it is good to go to q when the segment of a 's which follows is 'short'. Hence, it is needed to know what appears later in the word for making correctly this decision. This is typical of a behaviour which is not history-deterministic.

Formally, let us assume for the sake of contradiction that the automaton of Example 3 is history-deterministic. Let α be the corresponding correction function, and δ_n be the suitable translation strategy for all $n \in \omega$. Let us consider the behaviour of the automaton over the word bb . There is a run over this word of value 0, namely the one which goes successively through states p, q and r . Hence (by history-determinism), there is a run driven by $\delta_{\alpha(0)}$ over b . This run has to be the same one (which is the only one), and hence witnesses that $\delta_{\alpha(0)}(\varepsilon, b) = (p, b, \varepsilon, q)$. If we consider now the word $u = ba^{\alpha(0)+1}bb$, this word admits also a 0-run, namely the one staying $\alpha(0) + 2$ times in state p , then going to q and r . Hence (by history-determinism) there must be a run driven by $\delta_{\alpha(0)}$ over u . But we have seen that $\delta_{\alpha(0)}(\varepsilon, b) = (p, b, \varepsilon, q)$, i.e., the run must

start with the transition (p, b, ε, q) . This is a contradiction since no $\alpha(0)$ -run of the automaton do begin with this transition over the word u .

However, the same function can be accepted by the following history-deterministic automaton:



(remark that the sole source of non-determinism in this automaton occurs in state q when reading letter a). The automaton works as follows. State p waits for the next occurrence of the letter b , and then the run proceeds to state q . In state q , one counts the number of consecutive occurrences of letter a , but the automaton may also, at any moment, decide to withdraw the counting of the current segment of a 's, and go back to state p . If the automaton succeeds in reading a complete segment of a 's, then it can go while reading b to a the accepting state r . When the state r reached, whatever happens, the automaton stays in r , performing no actions on counters.

Let us show why this automaton is history-deterministic. We set α to be the identity. For all $n \in \omega$, we describe the translation strategy δ_n . The principle of the strategy is to stay as long as possible in state q , and to go back to state p only in the case staying one step more in state p would make the run violate the condition of being an n -run. We will not express in detail the value of δ_n . It is set to always go through the only possible transition when there is no choice, and when in state q and reading the letter a , the strategy is to take the transition (q, a, ic, q) if the current value of the counter is less than n , and to take the transition (q, a, ε, p) otherwise.

By construction, every partial run driven by δ_n is an n -partial run. Furthermore, if there is an n -run of the automaton, then the run driven by δ_n exists: The automaton is history-deterministic.

In practice, one is not just interested in proving that an automaton is history-deterministic, but also on the fact that it accepts a certain function (what we did not really do in the previous example). Hence, we do not directly use the above definition, but rather the following fact.

Fact 6 *Let \mathcal{A} be a B -automaton (resp., S -automaton) over the alphabet \mathbb{A} , and f be a mapping from \mathbb{A}^* to $\omega + 1$, then:*

- \mathcal{A} is history-deterministic and accepts a function \approx -equivalent to f ; iff
- $\llbracket \mathcal{A} \rrbracket_B \preceq f \preceq \llbracket \mathcal{A} \rrbracket_B^\delta$ (resp., $\llbracket \mathcal{A} \rrbracket_S \preceq f \preceq \llbracket \mathcal{A} \rrbracket_S$) in which δ_n is a translation strategy for all $n \in \omega$.

In this work, history-determinism plays a minor role. We will just see that every cost function accepted by a cost automaton can be accepted by an history-deterministic one (see Theorem 1 in the next section). This does not by itself justify the introduction of this notion. The reason for having history-deterministic

automata is that those automata behave particularly well in the context of games and trees. The extension of the theory of regular cost functions to trees is the subject of a forthcoming paper with Christof Löding. In this paper, history-deterministic automata and games play a central role, and some key steps rely on the use of Theorem 1.

2.6 Duality and regularity

The result of duality states that the two dual forms of automata, namely B - and S -automata, are equivalent. It is central in the theory.

Theorem 1 (duality). *A cost function over words is accepted by an [history-deterministic] [simple] [hierarchical] B -automaton iff it is accepted by an [history-deterministic] [simple] [hierarchical] S -automaton. Those equivalences are effective and of elementary complexity. Such cost functions are called regular.*

The proof of this theorem is difficult. It requires to translate the cost function accepted by the first automaton in the algebraic world, i.e., into a recognisable cost function (Section 4), and then to translate the recognisable cost function into the suitable form of automata (also Section 4). The results allowing to perform this are developed in the subsequent sections. The first step in the translation is possible thanks to Lemma 57. The second steps uses Lemma 58 if one aims at producing a B -automaton, or Lemma 61 if one is interested into an S -automaton.

Remark 2. According to Remark 1, translating a no-counter B -automaton into a no-counter S -automaton (and vice-versa) is easy to achieve by using any complementation construction for standard non-deterministic automata. Hence Theorem 1 can be seen as a replacement for both the results of complementation and determinisation in the classical theory of regular languages.

In [5], nothing similar to Theorem 1 can be found. However, with some amount of efforts, it is possible to derive the equivalence between [hierarchical] B -automata and [hierarchical] S -automata, (in slightly different simple form) from the result of complementation. However, using the proofs in [5] entails a long theoretical detour, and the constructions in [5] give a non-elementary blowup in the number of states. The methods are also radically different. Furthermore, the notion of history-determinism has no equivalent in [5], and cannot even be achieved with the form of automata used.

Hence, at this point of the exposition, we know that regular cost functions can be described by different forms of automata, that such functions are closed under minimum, maximum, inf-projection and sup-projection. Furthermore, we will also see in Section 4.4 that there exists an equivalent algebraic presentation to regular cost functions, and that the relation \preceq is decidable over them (Theorem 3). In the next section we introduce the necessary algebraic notions for obtaining these results.

3 The algebraic model: stabilisation monoids

The purpose of this section is to describe a purely algebraic model which is equivalent to the cost automata of the previous section. This framework is more technical. It is equivalent to cost automata only as far as one considers the functions modulo the equivalence \approx .

The key idea – which is directly inspired from the work of Simon and others – is to develop an algebraic notion (the stabilisation monoid) in which a special operator (called the stabilisation, \sharp) allows to express what happens when one iterates ‘a lot of times’ some element. In particular, it says whether one should count or not the number of iterations of this element. The terminology ‘a lot of times’ is very vague, and for this reason such a formalism cannot describe precisely functions. However, it is perfectly suitable for describing cost functions.

The remaining of the section is organised as follows. We first introduce the notion of stabilisation monoids in Section 3.1. We then present in Section 3.2 some useful notions in the study of stabilisation monoids. In Section 3.3 we introduce the notion of cost sequences, i.e., we describe the algebraic counterpart to the equivalence \approx used for cost functions, and develop the corresponding mathematical framework. Those notions are combined in Section 3.4 in which we introduce compatible mappings, the object capturing the semantics of stabilisation monoids. Section 3.5 is dedicated to the proof of the key Theorem 2 which states that to each stabilisation monoid corresponds a compatible mapping, and furthermore that it is unique.

3.1 Stabilisation monoids

A *semigroup* $\mathbf{S} = \langle S, \cdot \rangle$ is a set S equipped with an associative operation ‘ \cdot ’. A *monoid* is a semigroup such that the product has a *neutral element* 1, i.e., such that $1 \cdot x = x \cdot 1 = x$ for all $x \in S$. Given a semigroup $\mathbf{S} = \langle S, \cdot \rangle$, one extends the product to products of arbitrary length by defining π from S^+ to S by $\pi(a) = a$ and $\pi(ua) = \pi(u) \cdot a$. If the semigroup is a monoid of neutral element 1, one further set $\pi(\varepsilon) = 1$.

An idempotent in \mathbf{S} is an element $e \in S$ such that $e \cdot e = e$. One denotes by $E(\mathbf{S})$ the set of idempotents in \mathbf{S} . An *ordered semigroup* $\langle S, \cdot, \leq \rangle$ is a semigroup $\langle S, \cdot \rangle$ together with an order \leq over S such that the product \cdot is *compatible* with \leq ; i.e., $a \leq a'$ and $b \leq b'$ implies $a \cdot b \leq a' \cdot b'$. An *ordered monoid* is an ordered semigroup, the underlying semigroup of which is a monoid.

We are now ready to introduce the new notion of stabilisation semigroups and stabilisation monoids.

Definition 1. A stabilisation semigroup $\langle S, \cdot, \leq, \sharp \rangle$ is a finite ordered semigroup $\langle S, \cdot, \leq \rangle$ together with an operator $\sharp: E(\mathbf{S}) \rightarrow E(\mathbf{S})$ (called the stabilisation) such that:

- for all $e \leq f$ in $E(\mathbf{S})$, $e^\sharp \leq f^\sharp$;
- for all $a, b \in S$ with $a \cdot b \in E(\mathbf{S})$ and $b \cdot a \in E(\mathbf{S})$, $(a \cdot b)^\sharp = a \cdot (b \cdot a)^\sharp \cdot b$;⁵

⁵ This equation states that \sharp is a *consistent mapping* in the sense of [23, 25].

- for all $e \in E(\mathbf{S})$, $e^\# \leq e$;
- for all $e \in E(\mathbf{S})$, $(e^\#)^\# = e^\#$.

It is called a stabilisation monoid if furthermore $\langle S, \cdot \rangle$ is a monoid and $1^\# = 1$ in which 1 is the neutral element of ‘ \cdot ’.

The intuition is that $e^\#$ represents what is the value of e^n when n becomes ‘very large’. Some consequences of the definitions, namely

$$\text{for all } e \in E(\mathbf{S}), \quad e^\# = e \cdot e^\# = e^\# \cdot e = e^\# \cdot e^\# = (e^\#)^\#,$$

make perfect sense in this respect: repeating ‘a lot of e s’ is equivalent to see one e followed by ‘a lot of e s’, etc...

However, this view is incompatible with the classical view on monoids, in which by induction, if $e \cdot e = e$, then $e^n = e$ for all $n \geq 1$. The idea in stabilisation monoids is that the product is something that cannot be iterated ‘a lot of times’. This implies that thinking that for all $n \geq 1$, e^n is the same as e becomes something ‘incorrect’. The value of e^n is e if n is ‘small’, and it is $e^\#$ if n is ‘big’. Most of the remaining of the section is devoted to the formalisation of this intuition. This requires the development of a suitable mathematical framework. This approach is then validated by Theorem 2 which associates unique semantics to stabilisation monoids.

However, even if the material necessary for working with stabilisation monoids has not been provided, it is already possible to give some examples of stabilisation monoids that are constructed from an informal idea of their intended meaning.

Example 6. In this example we start from an informal idea of what we would like to compute, and construct a stabilisation monoid from it. At this point in the exposition of the theory, the explanations have to remain informal. However, this examples should illustrate how one can already reason at this level of understanding.

Imagine you want, among words over a and b , to separate the ones that possess ‘a lot of occurrences of a ’s’ from the ones that have only ‘a few occurrences of a ’s’, i.e., imagine you want to describe a stabilisation monoid which would ‘count’ the number of occurrences of a ’s.

For doing this, one should separate three ‘kinds’ of words:

- the kind of words with no occurrence of a ; let b be the corresponding element in the stabilisation monoid,
- the kind of words with at least one occurrence of a , but only ‘a few’ such occurrences; let a be the corresponding element in the stabilisation monoid,
- the kind of words with ‘a lot of occurrences of a ’s’; let 0 be the corresponding element in the stabilisation monoid.

As one can check the word b is of kind b , while the word a is of kind a . The words that we intend to separate – the one with a lot of a ’s – are the one of kind 0 . Once this three elements known, let us complete the definition of the stabilisation monoid.

Of course, iterating twice, or a lot of time, words that contains no occurrences of a yield words with no occurrence of the letter a . We capture this with the equalities $b = b \cdot b = b^\sharp$.

Now words that have at least one a , but only a few number of occurrences of a 's, should not be affected by appending b letters to their left or to their right. I.e., we set $a = b \cdot a = a \cdot b$. Even more, appending a word with 'few a 's' to another word with 'few a 's' does also give a word with 'few a 's'. I.e., we set $a \cdot a = a$.

However, if we iterate 'a lot of times' a word with at least one occurrence of a , it yields a word with 'a lot of a 's'. Hence we set $a^\sharp = 0$. We also easily get the equations $b \cdot 0 = 0 \cdot b = a \cdot 0 = 0 \cdot a = 0 \cdot 0 = 0^\sharp = 0$ by inspecting all situations. We reach the following description of the stabilisation monoid:

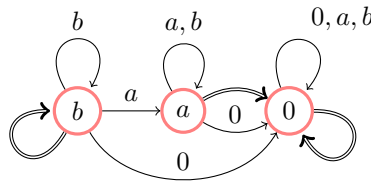
	b	a	0	\sharp
b	b	a	0	b
a	a	a	0	0
0	0	0	0	0

The left part describes the product operation \cdot , while the rightmost column gives the value of stabilisation \sharp (in general, this column may be partially defined since stabilisation is defined only for idempotents).

To complete the definition, we need to define the ordering over $\{1, a, 0\}$. The least we can do is setting $0 \leq a$ and $x \leq x$ for all $x \in \{1, a, 0\}$. This is mandatory, since by definition of a stabilisation monoid $a^\sharp \leq a$, and we have $a^\sharp = 0$. The reader can check that all the properties that we expect from a stabilisation monoid are now satisfied.

The intuition behind the ordering is that depending on what we mean by 'a lot of a 's', the same word can be of kind a or of kind 0 . For instance the word a^{100} is of kind a if we consider that 100 is 'a few', while it is of kind 0 if we consider that 100 is 'a lot'. For this reason, there is a form of continuum that allows to go from a to 0 . The order \leq captures this relationship between elements.

Another way to present the same stabilisation monoid is by its Cayley graph:



As in a standard Cayley graph, there is an edge labeled by y going from every vertex x to $x \cdot y$. Furthermore, there is a double arrow linking every idempotent x to its stabilised version x^\sharp .

Example 7. Imagine this time that one wants to compute the size of the longest sequence of consecutive a 's, once more over the alphabet $\{a, b\}$. Then we would separate four 'kinds' of words:

- the kind consisting only of the empty word; let it be 1,
- the kind of words, containing only occurrences of a , at least one occurrence of it, but only ‘a few’ of them; let the corresponding element be a ,
- the kind of words containing at least one b , but no long sequence of consecutive a ’s; let the corresponding element be b ,
- the kind of words that contain a long sequence of consecutive a ’s; let the corresponding element be 0.

It is clear that the words that have a long sequence of consecutive a ’s are the ones of kind 0. It is clear also that the word a is of kind a and the word b is of kind b .

The table of product and stabilisation is then naturally the following:

	1	a	b	0	#
1	1	a	b	0	1
a	a	a	b	0	0
b	b	b	b	0	b
0	0	0	0	0	0

We complete the definition of this stabilisation monoid by defining the ordering. For this, we let $x \leq x$ hold for any $x \in \{1, a, b, 0\}$, and we further set $0 \leq a$ and $0 \leq b$. Indeed $0 = a^\# \leq a$ is mandatory since $0 = a^\# \leq a$ must be satisfied, and $0 \leq b$ is mandatory since $0 = a^\# \cdot b \leq a \cdot b = b$ must be satisfied. Once more the ordering corresponds to the intuition that there exist words that can be of kind a (e.g., the word a^{100}) or b (e.g., the word ba^{100}), and that have kind 0 if we change what we intend by ‘a lot of’.

Remark 3. The notion of stabilisation monoids extends the standard definition of monoids. This is more than a formal concept: a lot of standard results concerning monoids have counterparts in the world of stabilisation monoids. For making this relationship more precise, let us describe the canonical way to translate a monoid into a stabilisation monoid.

Let $\mathbf{M} = \langle M, \cdot \rangle$ be a monoid. The corresponding stabilisation monoid is:

$$\mathbf{M}_\# = \langle M, \cdot, =, id_{E(M)} \rangle.$$

In other words, the monoid is extended with a trivial ordering (the equality), and the stabilisation is simply the identity over idempotents. The reader can easily check that this object indeed respects the definition of a stabilisation monoid.

If we refer to the intuition we gave above, one extends the monoid by a identity stabilisation. This means that for all idempotents, one does not make the distinction between iterating it ‘a few times’, and ‘a lot of times’. Said differently, one never has to count the number of occurrences of this idempotents. This is consistent with the principle that a standard monoid has no quantitative capabilities.

We have seen through the above examples how easy it is to work with stabilisation monoids at an informal level. An important part of the remaining of the

section is dedicated to give formal definitions to this informal reasoning. In the above explanations, we worked with the imprecise terminology ‘a few’ and ‘a lot of’. Of course, the value (what we referred to as ‘the kind’ in the examples) of a word depends on what is the frontier we fix for separating ‘a few’ from ‘a lot’. We introduce below the notion of cost sequences that allow to work simultaneously with all possible ‘frontiers’.

3.2 Structural properties of stabilisation monoids

We will make use of Green’s relations. This is a very important ingredient in different proofs. Given two elements $a, b \in M$, $a \leq_{\mathcal{J}} b$ if $a = xby$ for some $x, y \in M$. If $a \leq_{\mathcal{J}} b$ and $b \leq_{\mathcal{J}} c$, then $a \leq_{\mathcal{J}} c$. We write $a <_{\mathcal{J}} b$ to denote $a \leq_{\mathcal{J}} b$ and $b \not\leq_{\mathcal{J}} a$. The relation \mathcal{J} is one of Green’s relation among others. Thanks to Lemma 7, we will never have to refer to the other relations of Green nor to the relationship between them. The interest we have in the relation $\leq_{\mathcal{J}}$ is that it provides an induction parameter for the proofs.

A *regular element* in a monoid is an element a such that $a \cdot s \cdot a = a$ for some $s \in M$. A \mathcal{J} -class is regular if it contains a regular element. The following is standard.

Proposition 9. *The following items are equivalent for a \mathcal{J} -class J :*

- J is regular,
- J contains an idempotent,
- there exists a, b in J such that $a \cdot b \in J$.

In particular either all elements in a \mathcal{J} -class are regular (and we name the \mathcal{J} -class *regular*), or none is regular (and we name the \mathcal{J} -class *irregular*).

Example 8. The above examples are particularly simple in terms of structure of \mathcal{J} -classes. In Example 6, each \mathcal{J} -class is trivial (namely restricted to a single element) and regular. The classes are ordered as follows:

$$0 <_{\mathcal{J}} a <_{\mathcal{J}} b .$$

The situation is similar in Example 7. Each \mathcal{J} -class is also trivial and regular. The classes are ordered as follows:

$$0 <_{\mathcal{J}} b <_{\mathcal{J}} a <_{\mathcal{J}} 1 .$$

It happens in these examples that $\leq_{\mathcal{J}}$ ‘refines’ the order \leq . This is a pure accident, and the interplay between the two orderings can become quite complex in practice. The only thing that we can say is that for all idempotent e , we have both $e^{\#} \leq_{\mathcal{J}} e$ and $e^{\#} \leq e$. Since the above examples consisted only of idempotents, we get the fake impression that a similar relationship can happen in general.

Lemma 7 below is a very important tool in the proofs. It allows to use stabilisation not only on idempotents, but more generally on all regular element. It is due to Kirsten [23, 25], who established it in the more general context of consistent mappings (stabilisations are a special instance of consistent mappings; a consistent mapping needs not satisfy in general the equation $(e^\#)^\# = e^\#$). This is a generalisation of similar results due to Simon and Leung for particular mappings [26, 36, 37, 27, 28, 38, 29]. We state here a version adapted to stabilisation monoids.

Lemma 7 (Kirsten [23, 25]). *One can consistently extend $\#$ to all regular elements in S in such a way that for all regular a in S ,*

$$a^\# = (a^\#)^\# , \quad a^\# \leq a , \quad a^\# \leq_{\mathcal{J}} a ,$$

and for all regular a, b in S such that $a\mathcal{J}b\mathcal{J}a \cdot b$,

$$a^\# \cdot b = a \cdot b^\# = a^\# \cdot b^\# = (a \cdot b)^\# .$$

Proof. Let a be a regular element. It is \mathcal{J} -equivalent to an idempotent e . Let x, y be such that $x \cdot e \cdot y = a$. One defines $a^\#$ to be $x \cdot e^\# \cdot y$. Kirsten (in [23, 25]) proves that this definition does not depend of the choice of e, x and y . He also establishes all the facts above but the one concerning the order \leq . However, since, $e^\# \leq e$ by definition of a $\#$ -monoid, one has $a^\# = x \cdot e^\# \cdot y \leq x \cdot e \cdot y = a$. \square

Corollary 1. *Given two \mathcal{J} -equivalent regular elements a, b , we have $a^\# \mathcal{J} b^\#$.*

Proof. Since a is regular, $a = a \cdot s \cdot a$ for some s . Since $a \leq_{\mathcal{J}} b$, $a = x \cdot b \cdot y$ for some $x, y \in M$. We have naturally, $a \cdot s \cdot y \leq_{\mathcal{J}} a$. Reciprocally, $a = a \cdot s \cdot y \cdot b \cdot x$, and hence $a \leq_{\mathcal{J}} a \cdot s \cdot y$. Overall $a\mathcal{J}a' = a \cdot s \cdot y$, and similarly $a\mathcal{J}a'' = y \cdot s \cdot a$. We now compute using the above lemma and $a'\mathcal{J}a''\mathcal{J}b$:

$$a^\# = (a \cdot s \cdot a \cdot s \cdot a)^\# = (a' \cdot b \cdot a'')^\# = a' \cdot b^\# \cdot a''.$$

This means $a^\# \leq_{\mathcal{J}} b^\#$, and since a and b play the same role $a\mathcal{J}b$. \square

Let us remark that for a regular element a , the value of $a^\#$ provided by Lemma 7 should not be thought as ‘the value of a^n when n becomes big’. The correct way to see it is: ‘ $a^\#$ is the value of $a_1 a_2 \dots a_n$ when n becomes big, under the assumption that $a_1 \mathcal{J} a_2 \dots \mathcal{J} a_n$ and $a_1 \dots a_n = a$ ’. So in particular, since a is \mathcal{J} -equivalent to an idempotent e is can be written as $a = x \cdot e \cdot y$, and one obtains that: ‘ $a^\#$ is the value of $x e^n y$ when n becomes big’.

Given a regular \mathcal{J} -class J in a stabilisation monoid we say that J is *stable* if $a^\# \in J$ for some $a \in J$ (equivalently for all $a \in J$ by Corollary 1). Otherwise it is said *unstable*.

Lemma 8. *For all regular element a , if a belongs to a regular \mathcal{J} -class, $a^\# = a$; otherwise $a^\# <_{\mathcal{J}} a$.*

Proof. Let us assume first that e is an idempotent such that $e\mathcal{J}e^\sharp$. Then⁶, since $e^\sharp = e \cdot e^\sharp = e^\sharp \cdot e$ we get $e\mathcal{H}e^\sharp$. It follows that the \mathcal{H} -class of e is a group, and as a consequence, contains exactly one idempotent. We have $e = e^\sharp$.

Consider now a regular such that $a^\sharp\mathcal{J}a$. Remind from the proof of Lemma 7, that a can be written as $x \cdot e \cdot y$ in which e is an idempotent \mathcal{J} -equivalent to a , and that $a^\sharp = x \cdot e^\sharp \cdot y$. We have then, $a\mathcal{J}a^\sharp \leq_{\mathcal{J}} e^\sharp \leq_{\mathcal{J}} e\mathcal{J}a$. Hence $e\mathcal{J}e^\sharp$, and by the above remark, $e = e^\sharp$. We conclude that $a^\sharp = x \cdot e^\sharp \cdot y = x \cdot e \cdot y = a$. \square

3.3 Cost sequences

The subject of this section is to introduce cost sequences, and the relations \sim_α and \preceq_α that are used in order to work with them. This mathematical framework formalise the terminology ‘a few’ and ‘a lot’.

Principle of cost sequences. The semantic of a stabilisation monoid has been informally stated so far using terms such as ‘there is a few number of’, or ‘there is a lot of occurrences of’... To make such statements meaningful, one needs to give a threshold value allowing to separate what should be considered as ‘a few’ from what should be considered as ‘a lot’. This is done using cost sequences.

A cost sequence over an ordered set E is a sequence $\mathbf{a} \in E^\omega$. The idea is that given a non-negative integer n , $\mathbf{a}(n)$ represents the value which corresponds to the choice of the threshold n . The intention being that E is a stabilisation monoid, we intend some monotonicity with respect to the order. Namely we require cost sequences to be α -non-decreasing, i.e., quasi-non-decreasing, the error to non-decreasingness being controlled by a correction function α .

In the explanations, we were using statements such as ‘a few’+‘a few’=‘a few’. Of course, such an equality cannot hold for any value of the threshold. That is why we do not use the relation ‘=’ nor ‘ \leq ’ to compare cost sequences, but rather the relations \sim_α and \prec_α defined below, in which α measures the ‘imprecision’ allowed in the equations. The subject of this section is to introduce the suitable notion for working with cost sequences, i.e., to define and describe how to use the relations \preceq_α and \sim_α .

As in the automata part, α, α', \dots range over non-decreasing mappings from ω to ω such that $\alpha(n) \geq n$ for all $n \in \omega$. Those mappings are called *correction functions*. From now, m and n implicitly range over ω . Recall that we take the convention of using bold symbols for ω -sequences over some set, i.e., for cost sequences.

We give here first a definition to \preceq_α and \sim_α restricted to non-decreasing sequences. This is a safe approximation which conveys a lot of intuition. Let (E, \leq) be an ordered set.

Given two non-decreasing sequences \mathbf{a} and \mathbf{b} , and a correction function α , set $\mathbf{a} \preceq_\alpha \mathbf{b}$ to hold if $\mathbf{a} \leq \mathbf{b} \circ \alpha$. In other words, \mathbf{b} ‘dominates’ \mathbf{a} when its coordinates are ‘shrunk’ by α . One defines also $\mathbf{a} \sim_\alpha \mathbf{b}$ to hold if both $\mathbf{a} \preceq_\alpha \mathbf{b}$ and $\mathbf{b} \preceq_\alpha \mathbf{a}$.

⁶ We use here elementary Green’s relation, though we did not introduce them in detail. See for instance [34] for more details.

For the sake of the description, assume that two sequences \mathbf{a} and \mathbf{b} have respective limits a and b in E . Then for all α , $\mathbf{a} \preceq_\alpha \mathbf{b}$ implies $a \leq b$. To this extent, \preceq_α refines the ordering of limits. The parameter α allows to further compare the respective speed at which the limits (when applicable) are reached.

It is very good to keep this incomplete definition in mind. The reader is welcome to go once through the section and check that almost all the statements do hold very naturally with this definition. Consider for instance, Fact 10 below which is obvious.

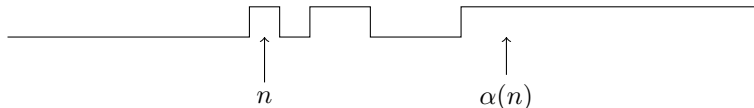
The real definitions of \preceq_α and \sim_α . Though it would be possible to work with the above definitions, we believe that it is mathematically better to directly work with sequences that are not necessarily non-decreasing. One reason for that is that some very natural operations do fail to produce non-decreasing sequences. The complete definition can be stated as follows.

Definition 2. *Given an ordered set (E, \leq) , a correction function α , and two sequences $\mathbf{a}, \mathbf{b} \in E^\omega$, define $\mathbf{a} \preceq_\alpha \mathbf{b}$ to hold when:*

$$\forall m. \forall n. \quad \alpha(m) \leq n \rightarrow \mathbf{a}(m) \leq \mathbf{b}(n) .$$

A sequence \mathbf{a} is said α -non-decreasing if $\mathbf{a} \preceq_\alpha \mathbf{a}$. We set \sim_α to be $\preceq_\alpha \cap \succeq_\alpha$.

To have an intuitive meaning of what it is to be α -non-decreasing, consider the following drawing representing some sequence:



The sequence takes only two values: low and high. In this case, the notion of α -non-decreasingness state that for every position n for which the value of the sequence is high, then every position greater or equal to $\alpha(n)$ has to have the value high. In other words, this definition does not constrain what should be the value of the sequence in the interval $(n, \alpha(n))$. Globally, the functions looks like non-decreasing, however, some temporary imprecision can appear.

Let us immediatly show that this definition is consistent with the explanations we gave before.

Lemma 9. *If \mathbf{a} and \mathbf{b} are non-decreasing, then for all α , $\mathbf{a} \preceq_\alpha \mathbf{b}$ iff $\mathbf{a} \leq \mathbf{b} \circ \alpha$. If \mathbf{a} is non-decreasing, then it is α -non-decreasing for all α .*

Proof. First statement. Let \mathbf{a} and \mathbf{b} be non-decreasing.

$$\begin{aligned} & \mathbf{a} \leq \mathbf{b} \circ \alpha \\ \text{iff } & \forall m. \mathbf{a}(m) \leq \mathbf{b}(\alpha(m)) \\ \text{iff } & \forall m. \forall n \geq \alpha(m). \mathbf{a}(m) \leq \mathbf{b}(n) \quad (\text{since } \mathbf{b} \text{ and } \alpha \text{ are non-decreasing}) \\ \text{iff } & \mathbf{a} \preceq_\alpha \mathbf{b} . \quad (\text{definition of } \preceq_\alpha) \end{aligned}$$

Second statement. Let \mathbf{a} be non-decreasing and α be a correction function. This implies that for all $n \in \omega$, $\alpha(n) \geq n$, and thus $\mathbf{a} \leq \mathbf{a} \circ \alpha$. According to the first statement, we obtain that $\mathbf{a} \preceq_{\alpha} \mathbf{a}$, i.e., that \mathbf{a} is α -non-decreasing. \square

Our second result shows that α -non-decreasing sequences can be thought as (in fact transformed into) non-decreasing sequences.

Proposition 10. *Every α -non-decreasing sequence is $\sim_{\alpha \circ \alpha}$ -equivalent to a non-decreasing sequence.*

Proof. Let \mathbf{a} be an α -non-decreasing sequence. Define $\mathbf{a}'(n)$ inductively as follows. The base case is $\mathbf{a}'(0) = \mathbf{a}(0)$. Then, assuming $\mathbf{a}'(n-1)$ is defined, set:

$$\mathbf{a}'(n) = \begin{cases} \mathbf{a}(n) & \text{if } \mathbf{a}(n) \geq \mathbf{a}'(n-1) , \\ \mathbf{a}'(n-1) & \text{otherwise.} \end{cases}$$

Of course, \mathbf{a}' defined in this way is non-decreasing. Let us prove that $\mathbf{a} \sim_{\alpha \circ \alpha} \mathbf{a}'$.

For this we first claim that \mathbf{a} and \mathbf{a}' often ‘coincide’. Namely, we claim that for all $m \in \omega$, there exists $k \in [m, \alpha(m)]$ such that $\mathbf{a}(k) = \mathbf{a}'(k)$. For proving this, let $l \leq \alpha(m)$ be the index of the latest use before or at $\alpha(m)$ of the first case in the definition of \mathbf{a}' (or 0 if the first case is never used before or at $\alpha(m)$). We have by definition of l , $\mathbf{a}(l) = \mathbf{a}'(l)$. Hence, if $l \geq m$, the claim is established for $k = l$. Otherwise, since the value of \mathbf{a}' keeps unmodified as long as the second case of the definition is used, we have $\mathbf{a}'(\alpha(m)) = \mathbf{a}(l)$. Now, since α is non-decreasing we have $\alpha(l) \leq \alpha(m)$, and using the \sim_{α} -monotonicity of \mathbf{a} , we obtain $\mathbf{a}'(\alpha(m)) = \mathbf{a}(l) \leq \mathbf{a}(\alpha(m))$. This means that the definition of $\mathbf{a}'(\alpha(m))$ used the first case in the construction, and by consequence $\mathbf{a}'(\alpha(m)) = \mathbf{a}(\alpha(m))$. This establishes the claim for $k = \alpha(m)$.

Let now $m, n \in \omega$ be such that $\alpha(\alpha(m)) \leq n$. By the above claim, there exists $k \in [\alpha(m), \alpha(\alpha(m))]$ such that $\mathbf{a}(k) = \mathbf{a}'(k)$. Hence, using the \sim_{α} -non-decreasingness of \mathbf{a} , and the non-decreasingness of \mathbf{a}' we get:

$$\mathbf{a}(m) \leq \mathbf{a}(k) = \mathbf{a}'(k) \leq \mathbf{a}'(n) .$$

This shows that $\mathbf{a} \preceq_{\alpha \circ \alpha} \mathbf{a}'$.

Conversely, let $m, n \in \omega$ be such that $\alpha(\alpha(m)) \leq n$. By the above claim, there exists $k \in [m, \alpha(m)]$ such that $\mathbf{a}(k) = \mathbf{a}'(k)$. Furthermore, by non-decreasingness of α , $\alpha(k) \leq n$. Hence, using the non-decreasingness of \mathbf{a}' and the \sim_{α} -non-decreasingness of \mathbf{a} , we get:

$$\mathbf{a}'(m) \leq \mathbf{a}'(k) = \mathbf{a}(k) \leq \mathbf{a}(n) .$$

This shows that $\mathbf{a}' \preceq_{\alpha \circ \alpha} \mathbf{a}$. \square

Let us now gain more knowledge about those relations \preceq_{α} and \sim_{α} . The following fact follows from the definitions.

Fact 10 *If $\alpha \leq \alpha'$ and $\mathbf{a} \preceq_{\alpha} \mathbf{b}$, then $\mathbf{a} \preceq_{\alpha'} \mathbf{b}$. If $\mathbf{a} \preceq_{\alpha} \mathbf{b} \preceq_{\alpha'} \mathbf{c}$, then $\mathbf{a} \preceq_{\alpha' \circ \alpha} \mathbf{c}$.*

The mapping α is used as a parameter of ‘precision’ for \sim and \preceq . The above fact states that using one transitivity step costs some ‘precision’.

Remark 4. It can give some intuition to view α as a form of ‘distance’. This idea works with the relation \sim_α . If you interpret $\mathbf{a} \sim_\alpha \mathbf{b}$ as ‘ \mathbf{a} is at distance at most α of \mathbf{b} ’, and $\mathbf{b} \sim_\alpha \mathbf{c}$ as ‘ \mathbf{b} is at distance at most α of \mathbf{c} ’, then you deduce using Fact 10 that $\mathbf{a} \sim_{\alpha \circ \alpha} \mathbf{c}$, i.e., ‘ \mathbf{a} is at distance at most $\alpha \circ \alpha$ of \mathbf{c} ’. This can be seen as a form of triangular inequality.

One usually also requires a distance to satisfy the equality $d(x, x) = 0$. However, there is no equivalent to a zero in our framework. Indeed for all α , one can find a sequence \mathbf{a} which is not α -non-decreasing, i.e., such that $\mathbf{a} \not\sim_\alpha \mathbf{a}$. An equivalent to zero can however be recovered, for instance by considering only non-decreasing sequences. Then we would have $\mathbf{a} \sim_{id} \mathbf{a}$ for all non-decreasing sequence \mathbf{a} . This means that id would behave as a zero for a distance. However, taking this solution would mean to add normalisation steps – thanks to Proposition 10 – each time we perform an operation which does not yield a non-decreasing sequence. We prefer here to simply not consider such a zero since it does not play any role in the framework.

α -monotonicity. We introduce now the important tool of α -monotonic mappings.

Definition 3. *Given two ordered sets (E, \leq) and (F, \leq) , a mapping f from E to F^ω is said α -monotonic if*

$$\forall a, b \in E. \quad a \leq b \rightarrow f(a) \preceq_\alpha f(b) .$$

We say that a mapping from E to F , in which E and F are ordered sets, is *monotonic* if for all $x, y \in E$, $x \leq y$ implies $f(x) \leq f(y)$. The notion of α -monotonicity naturally extends this notion of monotonicity as shown by the following fact.

Fact 11 *Let f be a monotonic mapping from E to F , then \mathbf{f} defined as:*

$$\begin{aligned} \mathbf{f} &: E \rightarrow F^\omega \\ a &\mapsto \mathbf{f}(a) = \text{the sequence constant equal to } f(a) , \end{aligned}$$

is α -monotonic for all α .

Let f be a mapping from E to F^ω . It can be turned into a mapping \tilde{f} from E^ω to F^ω defined by:

$$\text{for all } \mathbf{a} \in E^\omega \text{ and all } m \in \omega, \quad \tilde{f}(\mathbf{a})(m) = f(\mathbf{a}(m))(m) .$$

This definition behaves well with respect to composition.

Lemma 12. *For all $f : E \rightarrow F^\omega$ and $g : F \rightarrow G^\omega$, $\widetilde{(g \circ f)} = \tilde{g} \circ \tilde{f}$.*

Proof. For all $\mathbf{a} \in E^\omega$ and $n \in \omega$ we have:

$$\begin{aligned} \widetilde{(\tilde{g} \circ f)}(\mathbf{a})(n) &= (\tilde{g} \circ f)(\mathbf{a}(n))(n) \\ &= g(f(\mathbf{a}(n))(n))(n) \\ &= g(\tilde{f}(\mathbf{a})(n))(n) \\ &= \tilde{g}(\tilde{f}(\mathbf{a}))(n) = (\tilde{g} \circ \tilde{f})(\mathbf{a})(n) . \end{aligned} \quad \square$$

The following proposition discloses some key properties of α -monotonicity:

Proposition 11. *Let $f : E \rightarrow F^\omega$ be α -monotonic and $\mathbf{a}, \mathbf{b} \in E^\omega$, then:*

$$\mathbf{a} \preceq_\alpha \mathbf{b} \quad \text{implies} \quad \tilde{f}(\mathbf{a}) \preceq_\alpha \tilde{f}(\mathbf{b}) .$$

Proof. Let $u \preceq_\alpha v$, and let $m, n \in \omega$ be such that $\alpha(m) \leq n$. By definition $\tilde{f}(u)(m) = f(u(m))(m)$. From $u \preceq_\alpha v$, one gets $u(m) \leq v(n)$. Then by using α -monotonicity, one gets $f(u(m)) \preceq_\alpha f(v(n))$. This implies by definition of \preceq_α that $f(u(m))(m) \leq f(v(n))(n)$. Which means by definition of \tilde{f} , $\tilde{f}(u)(m) \leq \tilde{f}(v)(n)$. Overall we get $\tilde{f}(u) \preceq_\alpha \tilde{f}(v)$. \square

Corollary 2. *If $f : E \rightarrow F^\omega$ and $g : F \rightarrow G^\omega$ are α -monotonic, then $\tilde{g} \circ \tilde{f}$ is α -monotonic.*

Proof. Assume both f, g are α -monotonic, and consider $a \leq b$, then by α -monotonicity $g(a) \preceq_\alpha g(b)$. By the first statement, one deduces that $\tilde{f}(g(a)) \preceq_\alpha \tilde{f}(g(b))$. Hence $\tilde{f} \circ g$ is α -monotonic. \square

Relationship between \preceq_α and \preccurlyeq_α . As mentioned above, \preceq_α and \preccurlyeq_α are tightly connected. This section, we formalise this relationship.

Recall that given two functions f, g from some set E to $\omega + 1$, we have defined $f \preccurlyeq g$ to hold when $f \leq \bar{\alpha} \circ g$ (in which $\bar{\alpha}$ extends α by $\bar{\alpha}(\omega) = \omega$). Recall also that for F an ordered set and \mathbf{a}, \mathbf{b} non-decreasing sequences in F^ω , $\mathbf{a} \preceq \mathbf{b}$ holds iff $\mathbf{a} \leq \mathbf{b} \circ \alpha$. Keeping those formal facts in mind, imagine for the sake of the explanations that \mathbf{a}, \mathbf{b} and α are in fact increasing bijections from the reals to the reals, and let \mathbf{a}^{-1} and \mathbf{b}^{-1} be their inverses (for the composition). Then we would easily have $\mathbf{a} \preceq_\alpha \mathbf{b}$ iff $\mathbf{a} \leq \mathbf{b} \circ \alpha$ iff $\mathbf{b}^{-1} \leq \alpha \circ \mathbf{a}^{-1}$ iff $\mathbf{b}^{-1} \preccurlyeq_\alpha \mathbf{a}^{-1}$. Even if this has no meaning in the present context, this chain of equivalences conveys the important idea that \preccurlyeq and \preceq are two facets of the same phenomenon.

Before disclosing the complete relationship between \preceq_α and \preccurlyeq_α , let us do it in a particularly simple case. Given an ordered set (E, \leq) and $a \leq b$ in E , denote by $a|_k b$ for $k \in \omega$ the sequence in E^ω defined by:

$$(a|_k b)(n) = \begin{cases} a & \text{if } n \leq k , \\ b & \text{otherwise.} \end{cases}$$

Lemma 13. *For all $k, l \in \omega$ and $a < b$ in an ordered set E ,*

$$(a|_k b) \preceq_\alpha (a|_l b) \quad \text{iff} \quad l \preccurlyeq_\alpha k .$$

Proof. Remark that $a|_k b$ and $a|_l b$ are non-decreasing. Hence, by Lemma 9 and from the fact that $(a|_k b)(n) = b$ iff $n \geq k$, we have $a|_k b \preceq_\alpha a|_l b$ iff $\forall n. (n \geq k) \rightarrow (\alpha(n) \geq l)$ (\star). Assume \star , then in particular for $n = k$, we have $\alpha(k) \geq l$, i.e., $l \preceq_\alpha k$. Conversely assume $l \preceq_\alpha k$, i.e., $\alpha(k) \geq l$, and consider some $n \geq k$, then since α is non-decreasing, $\alpha(n) \geq \alpha(k) \geq l$. Hence \star holds. \square

Given an ordered set (E, \leq) , an *ideal* is a subset $I \subseteq E$ such that for all $a \in I$ and $b \leq a$, $b \in I$. Its complement in E is \bar{I} . Let us denote by $\downarrow(E)$ the set of ideals of E .

One goes back and forth between \preceq_α and \preceq_α using Proposition 12:

Proposition 12. *For all $\mathbf{a}, \mathbf{b} \in E^\omega$ and all correction function α , the following items are equivalent:*

- $\mathbf{a} \preceq_\alpha \mathbf{b}$,
- for all ideal $I \subseteq E$, $\sup\{n + 1 : \mathbf{b}(n) \in I\} \preceq_\alpha \inf\{n : \mathbf{a}(n) \notin I\}$.

Proof. Remark first that for all $a, b \in E$, $a \leq b$ iff $a \in I \leftarrow b \in I$ for all ideal $I \subseteq E$. Hence, item 1 and 2 are equivalent by:

$$\mathbf{a} \preceq_\alpha \mathbf{b}$$

$$\begin{aligned} &\text{iff } \forall n. \forall m. \alpha(n) \leq m \rightarrow \mathbf{a}(n) \leq \mathbf{b}(m) && \text{(definition of } \preceq_\alpha) \\ &\text{iff } \forall n. \forall m. \alpha(n) \leq m \rightarrow (\forall I \in \downarrow(E). \mathbf{a}(n) \in I \leftarrow \mathbf{b}(m) \in I) && \text{(above remark)} \\ &\text{iff } \forall I \in \downarrow(E). \forall n. \forall m. \alpha(n) \leq m \rightarrow (\mathbf{a}(n) \in I \vee \mathbf{b}(m) \notin I) \\ &\text{iff } \forall I \in \downarrow(E). \forall m. \forall n. (\mathbf{b}(m) \in I \wedge \mathbf{a}(n) \notin I) \rightarrow m < \alpha(n) && \text{(contraposition)} \\ &\text{iff } \forall I \in \downarrow(E). \sup\{n + 1 : \mathbf{b}(n) \in I\} \preceq_\alpha \inf\{n : \mathbf{a}(n) \notin I\}. && \text{(Fact 3)} \end{aligned}$$

\square

Corollary 3. *Let E be an ordered set, \mathbf{a} be an α -non-decreasing sequence over E , and I be an ideal of E , then:*

$$\sup\{n + 1 : \mathbf{a}(n) \in I\} \approx_\alpha \inf\{n : \mathbf{a}(n) \notin I\}.$$

Proof. It is clear that $\sup\{n + 1 : \mathbf{a}(n) \in I\} \geq \inf\{n : \mathbf{a}(n) \notin I\}$. Indeed, if the supremum of $\{n : \mathbf{a}(n) \in I\}$ is ω , then the claim holds. Otherwise it is N , and $N + 1 \in \{n : \mathbf{a}(n) \notin I\}$. For the converse, since $\mathbf{a} \preceq_\alpha \mathbf{a}$, we have $\sup\{n + 1 : \mathbf{a}(n) \in I\} \preceq_\alpha \inf\{n : \mathbf{a}(n) \notin I\}$ by Proposition 12. \square

Constant sequences, products, words, and various notations. We have seen so far all the important notions concerning the relations \preceq_α and \sim_α . What we need now is to provide convenient notations. In general, what we need is to be able to *lift* notations from elements to sequences.

The easiest such lifting is the following. For every element $a \in E$, we will identify a with the constant sequence \mathbf{a} defined by $\mathbf{a}(n) = a$ for all n . In the

same way, if f is a monotonic mapping from E to F , then f is implicitly lifted into a mapping from E to F^ω . Thus, an equation of the form:

$$a = f(b) ,$$

in which $a \in E$, $b \in F$ and f is a monotonic mapping from F to E , can be seen either as an equation in E , or an equation in E^ω . Of course, those two views are rigorously equivalent. Remark also that if α is a correction function then this above equation is equivalent to $a \sim_\alpha f(b)$ (since $=$ and \sim_α coincide over constant sequences). Similarly, $a \leq f(b)$ is equivalent to $a \preceq_\alpha f(b)$ (since \leq and \preceq_α coincide over constant sequences).

The second kind of notations we need is for products. Though we could introduce those notations for any product, we only do it for words, i.e., products of finite non-fixed length over a single set. Given an ordered set (E, \leq) , we order (implicitly) the words over E^* by $a_1 \dots a_m \leq b_1 \dots b_m$ if $m = n$ and $a_i \leq b_i$ for all $i = 1 \dots m$. Furthermore, for all $\mathbf{a}_1, \dots, \mathbf{a}_n \in E^\omega$, we implicitly identify $\mathbf{a}_1 \dots \mathbf{a}_n$ with the sequence which to m associates $\mathbf{a}_1(m) \dots \mathbf{a}_n(m)$, i.e., a sequence of words. The following lemma shows that such an identification is reasonable in our context.

Fact 14 For all $\mathbf{a}_1, \dots, \mathbf{a}_n, \mathbf{b}_1, \dots, \mathbf{b}_n \in E^\omega$,

$$\mathbf{a}_1 \dots \mathbf{a}_n \preceq_\alpha \mathbf{b}_1 \dots \mathbf{b}_n \quad \text{iff} \quad \text{for all } i = 1 \dots n, \mathbf{a}_i \preceq_\alpha \mathbf{b}_i .$$

3.4 Compatible mappings

We are now ready for introducing the key concept of compatible mappings. A mapping compatible with a given stabilisation monoid is an object which provides a quantitative semantic to the stabilisation monoid. It attaches to each word a sequence over the stabilisation monoid. The key result concerning compatible mappings is their existence and uniqueness, which is described in Theorem 2, and proved in the following section.

The principle of a compatible mapping is to be a replacement to the generalised product π in a monoid (or a semigroup). Given a sequence of elements $u \in M^*$ in a monoid $\langle M, \cdot \rangle$, $\pi(u)$ can be seen as the ‘value’ of u . In a similar way, a compatible mapping attaches to each $u \in M^*$ its ‘value’. However, since we deal with a quantitative notion, this value is a more complex object. A compatible mapping ρ attaches to each word u a sequence in M^ω . The intuition is that all the reasoning were performed so far using a terminology such as ‘a few’ or ‘a lot’. The value $\rho(u)(n)$ for a given n is an element M which is the value of u if we choose to consider values less than n as ‘small’, and values above as ‘big’. We start by giving an example of a compatible mapping, illustrating how it captures the informal descriptions done so far.

Example 9 (continuation of Example 6). In Example 6 we have constructed a stabilisation monoid over the elements $M = \{b, a, 0\}$. Let ρ be the mapping from

M^* to M^ω defined for all $u \in M^*$ and $n \in \omega$ by:

$$\rho(u)(n) = \begin{cases} b & \text{if } u \in b^* , \\ a & \text{if } 1 \leq |u|_a < n, \text{ and } |u|_0 = 0 , \\ 0 & \text{otherwise, i.e., if } |u|_0 > 0 \text{ or } |u|_a \geq \max(n, 1) . \end{cases}$$

Over the words over $\{a, b\}$, $\rho(u)(n)$ faithfully defines the ‘kind’ that we had described in our informal description of the stabilisation monoid, in which ‘a few’ is interpreted as ‘less than n ’, and ‘a lot’ as ‘at least n ’:

- ‘ b ’ is the the kind of words with no occurrence of a ;
- ‘ a ’ is the kind of words with at least one occurrence of a , but only ‘a few’ such occurrences;
- ‘ 0 ’ is the kind of words with ‘a lot of occurrences of a ’s’.

Remark that the definition of ρ goes beyond this simple description, since it is defined over words in which the letter 0 occurs. Such a letter can be seen here as any word over a, b which has ‘a lot’ of occurrences of a ’s.

This definition seems to capture the intention we had when defining this stabilisation monoid. The question is:

‘How is this mapping ρ in relation with the stabilisation monoid?’

We will see that the answer is that ρ is a mapping compatible with \mathbf{M} , and as a consequence it is the ‘only one’ which describes the semantics of the stabilisation monoid.

Let us now give the definition of a compatible mapping.

Definition 4. *Given a stabilisation monoid $\mathbf{M} = \langle M, \cdot, \leq, \# \rangle$, a mapping ρ from M^* to M^ω is compatible with \mathbf{M} if for some α we have:*

Monotonicity. ρ is α -monotonic,

(Remind that the order over M is implicitly extended componentwise to M^)*

Letter. for all $a \in M$, $\rho(a) \sim_\alpha a$,

(Remind that a is implicitly identified with the constant sequence equal to a)

Neutral. $\rho(\text{varepsilon}) \sim_\alpha 1$,

Product. for all $a, b \in M$, $\rho(ab) \sim_\alpha (a \cdot b)$,

Stabilisation. for all $e \in E(\mathbf{M})$, $m > 0$, $\rho(\overbrace{e \dots e}^{m\text{-times}}) \sim_\alpha e^\#|_m e$

(Remind that for all $a \leq b$ in M , $(a|_m b) \in S^\omega$ maps $[0, m)$ to a and $[m, \omega)$ to b)

Substitution. for all $u_1, \dots, u_n \in M^*$, $n \geq 0$, $\rho(u_1 \dots u_n) \sim_\alpha \tilde{\rho}(\rho(u_1) \dots \rho(u_n))$.

Remark 5. This remark is at the same time a form of example. Imagine the stabilisation monoid is constructed from a monoid $\mathbf{M} = \langle M, \cdot \rangle$ as explained in Remark 3. Consider now the application ρ defined for all $u \in M^*$ and $n \in \omega$ by:

$$\rho(u)(n) = \pi(u) .$$

(Recall that $\pi : M^* \rightarrow M$ is the extension of the product to any word.) We show below that this ρ is compatible with $\mathbf{M}_\#$, for any α .

Indeed, since \leq is trivial, a sequence is α -non-decreasing iff it is constant. Furthermore, \preceq_α and \sim_α correspond to equality over constant sequences. Hence α -monotonicity amounts to say that for all u , $\rho(u)$ is constant. This is true. The other items, by inlining the definition of ρ , boil down to the following:

Letter: for all $a \in M$, $\pi(a) = a$ and $\pi(\varepsilon) = 1$

Product: for all $a, b \in M$, $\pi(ab) = a \cdot b$

Stabilisation: for all $e \in E(\mathbf{M})$ and $m > 0$, $\pi(e^m) = e$ (since $e^\# = e$)

Substitution: for all $u_1, \dots, u_n \in M^*$, $\pi(u_1 \dots u_n) = \pi(\pi(u_1) \dots \pi(u_n))$.

These facts are obvious from the definition of π and the associativity of \cdot . More generally, the substitution rule in a compatible mapping can be seen as a generalised form of associativity.

Example 10. Consider the stabilisation monoid \mathbf{M} of Example 6, and the mapping ρ defined in Example 9. Let us show that this mapping ρ is compatible with \mathbf{M} :

Monotonicity. We prove *id*-monotonicity. Let $u \leq v$. If $v \in b^+$ then $\rho(u) \leq b = \rho(v)$ (since $\rho(v) = b$ is maximal), i.e., $\rho(u) \preceq_{id} \rho(v)$. If u contains the letter 0 then $\rho(u) = 0 \leq \rho(v)$. Otherwise, since $u \leq v$, u and v contain at least one occurrence of a , and no occurrence of 0. Hence $\rho(u) = (0|_{|u|_a} a)$ and $\rho(v) = (0|_{|v|_a} a)$. But since $u \leq v$, $|u|_a \geq |v|_a$. We get that $\rho(u) \preceq_{id} \rho(v)$.

Letter. We have $\rho(b) = b$, $\rho(0) = 0$ and $\rho(a) = 0|_1 a$. This implies $\rho(a) \sim_\alpha a$ for $\alpha(n) = \max(1, n)$.

Product. The only non trivial case is $\rho(aa) = (0|_2 a) \sim_\alpha (0|_1 a) = a \cdot a$ which holds for $\alpha(n) = \max(2, n)$.

Stabilisation. Every element is an idempotent. Let $m \geq 1$. For all $x \in M$, $\rho(x^m) = x^\#|_m x$ by definition.

Substitution. Let $u_1, \dots, u_n \in M^*$ and $u = u_1 \dots u_n$. If for all i , $u_i \in b^*$, then $\rho(u) = b = \tilde{\rho}(\rho(u_1) \dots \rho(u_n))$. If the letter 0 occurs in some u_i , then $\rho(u) = 0 = \tilde{\rho}(\rho(u_1) \dots \rho(u_n))$. Otherwise u contains no 0, and at least one occurrence of a . Let K be the set of indices i such that u_i contains an occurrence of a . By applying the definition of ρ and $\tilde{\rho}$ we claim that

$$\tilde{\rho}(\rho(u_1) \dots \rho(u_n)) = 0|_{\max(|K|, \max_{i \in K} |u_i|_a)} a,$$

indeed if $m < |u_i|_a$ for some i , then $\rho(u_i)(m) = 0$, and as a consequence $\tilde{\rho}(\rho(u_1) \dots \rho(u_n))(m) = 0$. And otherwise, if $m < |K|$, $(\rho(u_1) \dots \rho(u_n))(m)$ contains no occurrences of 0, but $|K|$ -many occurrences of a , and hence once more $\tilde{\rho}(\rho(u_1) \dots \rho(u_n))(m) = 0$. Finally, if $m \geq \max(|K|, \max_{i \in K} |u_i|_a)$, then $(\rho(u_1) \dots \rho(u_n))(m)$ contains no occurrences of 0 and at most m occurrences of a . We obtain $\tilde{\rho}(\rho(u_1) \dots \rho(u_n))(m) = a$. Using Example 4 one has $\max(|K|, \max_{i \in K} |u_i|_a) \approx_\alpha |u|_a$ (for $\alpha(m) = m^2$), and consequently using Proposition 12, $\rho(u) \sim_\alpha \tilde{\rho}(\rho(u_1) \dots \rho(u_n))$.

Lemma 15. *Let ρ be a mapping compatible with some stabilisation monoid \mathbf{M} , there exists α such that for all $u, v \in M^*$, $\rho(uv) \sim_\alpha \rho(u) \tilde{\rho}(v)$.*

Proof. Let α be the correction function witnessing the compatibility of ρ with respect to \mathbf{M} . Remark that following the notations of α -monotonic mappings, $\tilde{\cdot}$ is a mapping from $(M \times M)^\omega$ to M^ω . By definition $\mathbf{a}\tilde{\mathbf{b}}$ is the sequence of n 'th element $\mathbf{a}(n) \cdot \mathbf{b}(n)$: i.e., the componentwise product of sequences.

The product rule states that for all $a, b \in M$, $\rho(ab) \sim_\alpha a \cdot b$. Using Proposition 11, we can lift this to (\star) : for all α -non-decreasing $\mathbf{a}, \mathbf{b} \in M^\omega$, $\tilde{\rho(\mathbf{a}\mathbf{b})} \sim_\alpha \mathbf{a}\tilde{\mathbf{b}}$. Once those preliminary comments done, the reasoning is obvious. We have:

$$\rho(uv) \sim_\alpha \tilde{\rho(\rho(u)\rho(v))} , \quad (\text{substitution})$$

and,

$$\tilde{\rho(\rho(u)\rho(v))} \sim_\alpha \rho(u)\tilde{\rho(v)} \quad (\text{property } \star)$$

We combine those two lines using Fact 10, and obtain $\rho(uv) \sim_{\alpha \circ \alpha} \rho(u)\tilde{\rho(v)}$ \square

Remark 6. Below, for simplifying proofs, we do not mention explicitly the correction functions, and just leave them in the statements. Hence, the proof of the above lemma would simply be written (we just mention ‘product and monotonicity’ in replacement of property \star):

$$\begin{aligned} \rho(uv) &\sim \tilde{\rho(\rho(u)\rho(v))} && (\text{substitution}) \\ &\sim \rho(u)\tilde{\rho(v)} . && (\text{product and monotonicity}) \end{aligned}$$

What is important to notice is that this sequence of equivalences is ‘short’ (here of length 2). Indeed, all the symbols ‘ \sim ’ are shorthands for \sim_α for a correction function α that depends only on the stabilisation monoid and the compatible mapping, but not on u and v . Hence, by using Fact 10, one can by transitivity obtain that $\rho(uv) \sim_{\alpha'} \rho(u)\tilde{\rho(v)}$ for an α' that depends only on the stabilisation monoid and the compatible mapping. This way of presenting proofs is only valid if the sequence of equivalences is bounded in terms of the stabilisation monoid and the compatible mapping.

Let us provide an example of an incorrect proof. Let us (fakely) prove that there exists α such that for all word $u \in M^*$, $\rho(u) \sim_\alpha \pi(u)$. Let us write u as $a_1 \dots a_k$, and:

$$\begin{aligned} \rho(u) = \rho(a_1 \dots a_k) &\sim a_1 \tilde{\rho}(a_2 \dots a_k) && (\text{Lemma 15 and letter}) \\ &\sim a_1 \tilde{a_2} \tilde{\rho}(a_3 \dots a_k) && (\text{Lemma 15 and letter}) \\ &\vdots \\ &\sim a_1 \tilde{\dots} \tilde{a_k} = \pi(u) . \end{aligned}$$

This proof is incorrect because the number of transitivity steps used is unbounded, since it is linear in the length of u . For this reason, it would be invalid to do as if the symbol \sim was transitive. What we have proved in fact is that for all word u , there exists α such that $\rho(u) \sim_\alpha \pi(u)$. Up to a slight transformation in the statement, we have obtained Lemma 16 just below.

Lemma 16. *Let ρ be compatible with \mathbf{M} , then for all positive integer m , there exists α such that $\rho(u) \sim_\alpha \pi(u)$ for all $u \in M^*$ with $|u| \leq k$.*

Corollary 4. *For all ρ compatible with \mathbf{M} , and all $u \in M^*$, $\lim(\rho(u)) = \pi(u)$.*

Proof. It is sufficient to remark that if $\mathbf{a} \sim b$ for some sequence $\mathbf{a} \in M^\omega$ and element $b \in M$, then $\lim \mathbf{a} = b$. According to the previous corollary, $\rho(u) \sim \pi(u)$ for all word $u \in M^*$. Hence we get $\lim \rho(u) = \pi(u)$. \square

Corollary 5. *For all word $u \in M^*$ and all $n \in \omega$, $\rho(u)(n) \leq \pi(u)$.*

All those definitions are not really meaningful until we prove Theorem 2.

Theorem 2. *For every stabilisation monoid \mathbf{M} , there exists a mapping ρ compatible with it.*

Furthermore it is unique up to \sim . More precisely, for any stabilisation monoid \mathbf{M}' , of which \mathbf{M} is a sub-stabilisation monoid, and every two mappings ρ, ρ' compatible with \mathbf{M} and \mathbf{M}' respectively, there exists α such that $\rho(u) \sim_\alpha \rho'(u)$ for all word u over M .

This key result states that for all stabilisation monoid, there exists a unique mapping compatible with it. In the context of standard monoids, this states the obvious fact that the binary product can be uniquely extended into the generalised product π . It is reminiscent of the existence of unique extensions of finite Wilke algebras into ω -semigroups in the theory of regular languages of infinite words [41]. The approach is rigorously the same: provide a finite algebraic object and extend it to an infinite semantics. But the similarity stops here, and the proof methods are radically different.

The proof of Theorem 2 is the subject of the next section, namely, it aggregates Lemmas 17, and 18. For technical reasons, the proof is performed is performed for stabilisation semigroup. Let us show first how the notion of compatible mapping can be translated to the framework of stabilisation semigroups.

The stabilisation semigroup case As we already mentioned, we will have sometimes to use stabilisation semigroups during the proofs. There exists also a notion of compatible mappings or stabilisation semigroup. We present it here, and then establish a proposition showing how the two notions can be converted one into the other (Proposition 13).

Definition 5. *Given a stabilisation semigroup $\mathbf{S} = \langle S, \cdot, \leq, \# \rangle$, a mapping ρ from S^+ to S^ω is compatible with \mathbf{S} if for some α we have:*

Monotonicity. ρ is α -monotonic,

Letter. for all $a \in S$, $\rho(a) \sim_\alpha a$,

Product. for all $a, b \in S$, $\rho(ab) \sim_\alpha (a \cdot b)$,

Stabilisation. for all $e \in E(\mathbf{S})$, $m > 0$, $\rho(\overbrace{e \dots e}^{m\text{-times}}) \sim_\alpha e^\#|_m e$

Substitution. for all $u_1, \dots, u_n \in S^+$, $n > 0$, $\rho(u_1 \dots u_n) \sim_\alpha \tilde{\rho}(\rho(u_1) \dots \rho(u_n))$.

This definition differs only in three places from the one for stabilisation monoids, corresponding to the removal of the possibility to use the empty word. First, the application ρ is only defined over non-empty words. Second, the neutral rule has disappeared. Third, in the substitution rule, the words u_1, \dots, u_n are required now to be non-empty, and n to be positive.

However, those two notions are very close. Indeed, if ρ is an application compatible with a stabilisation monoid \mathbf{M} , its reduction to non-empty words is obviously compatible with \mathbf{M} seen this time as a stabilisation semigroup. Conversely, there is an easy way to transform a mapping compatible with a stabilisation semigroup into a mapping compatible with a stabilisation monoid. This is the subject of the following proposition.

Proposition 13. *Given a stabilisation monoid \mathbf{M} over the set M , and an application ρ from M^+ to M^ω which is compatible with \mathbf{M} considered as a stabilisation semigroup, then the application ρ extended with $\rho(\varepsilon) = 1$ is compatible with the stabilisation monoid \mathbf{M} .*

Proof. We just need to establish the neutral rule and the substitution rule which is stronger in the case of stabilisation monoids. The neutral rule is by definition. Let us concentrate on the substitution rule.

Let us first prove $\rho(1 \dots 1) \sim 1$. This comes from the definition if the length of $1 \dots 1$ is null. Otherwise using the stabilisation rule and the equality $1^\sharp = 1$ in stabilisation monoids:

$$\rho(\overbrace{1 \dots 1}^m) \sim 1^\sharp|_m 1 = 1 .$$

Let us prove now $\rho(a1 \dots 1) \sim a$. If $a1 \dots 1$ has length 1, this is the letter rule. Otherwise, we compute:

$$\begin{aligned} \rho(a1 \dots 1) &\sim \tilde{\rho}(\rho(a)\rho(1 \dots 1)) && \text{(substitution)} \\ &\sim \tilde{\rho}(a1) && \text{(letter rule, above case, and Proposition 11)} \\ &= \rho(a1) \sim a. && \text{(product rule and neutrality of 1)} \end{aligned}$$

Let us prove now that $\rho(a_1 1 \dots 1 a_2 \dots a_n 1 \dots 1) \sim \rho(a_1 \dots a_n)$ (i.e., we show that the value of a word $a_1 \dots a_n$ is unchanged when one inserts arbitrary many occurrences of 1). Let us compute:

$$\begin{aligned} \rho(a_1 1 \dots 1 a_2 \dots a_n 1 \dots 1) &\sim \tilde{\rho}(\rho(a_1 1 \dots 1) \dots \rho(a_n 1 \dots 1)) && \text{(substitution rule)} \\ &\sim \tilde{\rho}(a_1 \dots a_n) && \text{(above case and Proposition 11)} \\ &= \rho(a_1 \dots a_n) . \end{aligned}$$

Let us finally establish the substitution rule. Let u_1, \dots, u_n be words in M^* . Let $i_1 < \dots < i_k$ be the indices such that $u_{i_j} \neq \varepsilon$. We compute:

$$\begin{aligned}
\rho(u_1 \dots u_n) &= \rho(u_{i_1} \dots u_{i_k}) \\
&\sim \tilde{\rho}(\rho(u_{i_1}) \dots \rho(u_{i_k})) && \text{(substitution)} \\
&\sim \tilde{\rho}(1 \dots 1 \rho(u_{i_1}) 1 \dots 1 \rho(u_{i_k}) 1 \dots 1) \\
&\hspace{15em} \text{(above case with Proposition 11)} \\
&\sim \tilde{\rho}(\rho(u_1) \dots \rho(u_n)) . && (\rho(\varepsilon) \sim 1 \text{ and Proposition 11})
\end{aligned}$$

□

3.5 Existence and uniqueness of compatible mappings

The goal of this section is to prove Theorem 2: Every stabilisation monoid admits a mapping compatible with it. Furthermore, it is unique (up to \sim). We make this proof for stabilisation semigroups. The conversion can be done using Proposition 13. Since we are dealing with stabilisation semigroups, till the end of the section, words will always be assumed to be non-empty, unless specified otherwise.

Hence, we simultaneously prove the following two lemmas.

Lemma 17. *For all stabilisation semigroup, there exists a mapping compatible with it.*

Given two stabilisation semigroup $\mathbf{S} = \langle S, \cdot, \leq, \sharp \rangle$ and $\mathbf{T} = \langle T, \cdot', \leq', \sharp' \rangle$, \mathbf{T} is a sub-stabilisation monoid of \mathbf{S} if $T \subseteq S$, and \cdot, \leq and \sharp coincide over T with \cdot', \leq' and \sharp' respectively.

Lemma 18. *For all stabilisation semigroup \mathbf{S} , all sub-stabilisation semigroup \mathbf{T} of \mathbf{S} , and all mappings ξ and ρ compatible with \mathbf{S} and \mathbf{T} respectively, ξ and ρ are \sim -equivalent over T^+ , where T is the set of elements of \mathbf{T} .*

Our first step consists in constructing a mapping β which is a ‘candidate’ to be the compatible mapping restricted to so called ‘smooth words’ (see definition below). The proof then proceeds into extending this construction to all words along an induction on the \mathcal{J} -classes of the stabilisation monoid. During each induction step, there are two sub-steps: first treat the case of ‘trivial words’ (see definition below), and then the general case.

Definition over smooth words: the mapping β . Our first step consists in defining a mapping β that satisfies all the properties of a compatible mapping over smooth words. A word $u = a_1 \dots a_n$ in S^+ is said *J-smooth* for some \mathcal{J} -class J if a_1, \dots, a_n and $\pi(u)$ belong to J . Equivalently, each non-empty factor v of u is such that $\pi(v) \in J$. A word is said simply *smooth* if it is J -smooth for some \mathcal{J} -class J .

Given a J -smooth word $u \in S^+$, define $\beta(u)$ to be:

$$\beta(u) = \begin{cases} \pi(u) & \text{if } J \text{ is irregular} \\ \pi(u)^\#_{|u|} \pi(u) & \text{otherwise.} \end{cases}$$

You can remark that if u is of the form $e \dots e$ for some idempotent e , then this definition coincide with the stabilisation rule. Remark also that if J is irregular, then u has to be of length 1, and the definition of β coincides this time with the letter rule.

Lemma 19. *For all $k > 0$, there exists α such that for all smooth word u of length at most k , $\beta(u) \sim_\alpha \pi(u)$.*

Proof. This comes directly from the definition and from $\pi(u)^\# \leq \pi(u)$, with α such that for all $n \in \omega$, $\alpha(n) = \max(n, k)$. \square

Lemma 20. *There exists α such that β is α -monotonic.*

Proof. Let k be $|J|^2$ for J the \mathcal{J} -class of maximal size in \mathbf{M} . We prove that if u and v are smooth and $u \leq v$, then $\beta(u) \preceq_\alpha \beta(v)$ for a suitable α .

If $|u| \leq k$, by Lemma 19, we have $\beta(u) \sim_\alpha \pi(u) \leq \pi(v) \sim_\alpha \beta(v)$ (for the α of Lemma 19).

Else if $|u| > k$, by standard combinatoric techniques, one can decompose u as $u'eu''$ and v as $v'fv''$ with $u' \leq v'$, $e \leq f$, and $u'' \leq v''$ where e and f are non-empty, and both $\pi(e)$ and $\pi(f)$ are idempotents. Hence, we have $\pi(u') \leq \pi(v')$, $\pi(e) \leq \pi(f)$, $\pi(u'') \leq \pi(v'')$ and $\pi(e)^\# \leq \pi(f)^\#$ (property of a stabilisation monoid) from which we directly get $\pi(u)^\# \leq \pi(v)^\#$ (compatibility of the product with respect to the order). Using $\pi(u) \leq \pi(v)$ and $\pi(u)^\# \leq \pi(v)^\#$, we get $\beta(u) \preceq_{id} \beta(v)$.

Combining the two cases together, we get $\beta(u) \preceq_{\alpha \circ \alpha} \beta(v)$. \square

Lemma 21. *There exists α such that for all S -words u, v , such that uv is smooth, $\beta(uv) \sim_\alpha \beta(u) \cdot \beta(v)$.*

Proof. If u or v is empty, then the result is obvious.

Otherwise, since $\pi(u)$, $\pi(v)$ and $\pi(uv)$ are \mathcal{J} -equivalent by hypothesis. As a consequence, by Proposition 9, this \mathcal{J} -class is regular. By a case analysis, using Lemma 7, one gets:

$$\begin{aligned} \beta(u) \cdot \beta(v) &= \pi(uv)^\#_{|\max(|u|, |v|)} \pi(uv) \\ &\sim \pi(uv)^\#_{|u|+|v|} \pi(uv) && \text{(Lemma 13)} \\ &= \beta(uv) \end{aligned}$$

in which we use the equivalence $x + y \approx_{\times 2} \max(x, y)$. \square

Let us now turn to the uniqueness of this construction.

Lemma 22. *There exists α such that for all smooth word $u \in S^+$, $\beta(u) \sim_\alpha \xi(u)$.*

Proof. Let $u = a_1 \dots a_n$ be J -smooth. If $n = 1$, this comes from the letter rule.

Otherwise according to Proposition 9, J is regular. Using standard semigroup techniques (see e.g. [34]), one can write each a_i as $b_i \cdot c_i$ and choose an idempotent $e \in J$ such that $c_i \cdot b_{i+1} = e$ for all $i = 1, \dots, n-1$. And we compute:

$$\begin{aligned}
\xi(u) &= \xi((b_1 \cdot c_1) \dots (b_n \cdot c_n)) \\
&\sim \xi(b_1 c_1 \dots b_n c_n) \\
&\sim \xi(b_1 (c_1 \cdot b_2) \dots (c_{n-1} \cdot b_n) c_n) \\
&\sim b_1 \cdot \xi(e^{n-1}) \cdot c_n \\
&\sim \pi(u)^\sharp|_{n-1} \pi(u) \\
&\sim \pi(u)^\sharp|_n \pi(u) \\
&= \beta(u)
\end{aligned}$$

in which we use only the compatibility rules for ξ . □

Structure of the induction. We present here the general structure of the proof of Lemmas 17 and 18.

The induction parameter is a non-negative integer K .

The induction hypothesis states that for all stabilisation semigroup \mathbf{T} of size K , there exists a mapping ρ_T and a correction function α_T such that:

- the mapping ρ_T is α_T -compatible with \mathbf{T} .

Furthermore, whenever \mathbf{T} is a sub-stabilisation semigroup of a stabilisation semigroup \mathbf{S} and for all mapping ξ compatible with \mathbf{S} , there exists α such that:

- for all smooth $u \in S^+$ and all $v \in T^+$ such that $u \leq v$, $\beta(u) \preceq_\alpha \rho_T(v)$,
- for all smooth $u \in S^+$ and all $v \in T^+$ such that $v \leq u$, $\rho_T(v) \preceq_\alpha \beta(u)$,
- for all $u \in T^+$, $\xi(u) \sim_\alpha \rho_T(u)$.

The first part proves the compatibility of the mapping. The two following items are necessary for the induction to go through. The last item establishes the uniqueness of ρ_T . It is clear that for $T = S$, the first item establishes Lemma 17, and the fourth Lemma 18. Hence, what remains to be done is the induction itself.

The base case is when $K = 0$. In this case the nowhere defined mapping ρ_\emptyset is compatible with \mathbf{T} , and the three other items trivially hold. Hence, let us consider now a stabilisation semigroup $\mathbf{T} = \langle T, \cdot, \leq, \sharp \rangle$ such that the induction hypothesis hold for every $K < |T|$. We aim at constructing a mapping ρ_T compatible with T , and establish the induction property for it.

Let J be a \mathcal{J} -class of \mathbf{T} which is maximal for $\leq_{\mathcal{J}}$. We set T' be $T \setminus J$. Using the maximality of J , the set T' satisfy $S \cdot T' \cdot S \subseteq T'$. We then resort to the following lemma:

Lemma 23. *Let $T' \subseteq T$ be such that $T \cdot T' \cdot T \subseteq T'$, then \mathbf{T} restricted to T is a sub-stabilisation semigroup of \mathbf{T} .*

Proof. By hypothesis, T induces an ordered sub-semigroup of \mathbf{S} . Furthermore, according to Lemma 7, for all idempotent $e \in E(\mathbf{S}) \cap T$, $e^\# = e \cdot e^\# \cdot e$. Hence, $e^\# \in T$. \square

Hence let us denote \mathbf{T}' the stabilisation semigroup \mathbf{S} restricted to T' . It follows from the induction hypothesis applied to T' that there exists a mapping $\rho_{T'}$ and a correction function $\alpha_{T'}$ which satisfy all the statements of the induction hypothesis.

The remaining of the section is devoted to constructing a mapping ρ_T satisfying the requirements of the induction hypothesis. The parameters $T, J, T', \rho_{T'}$ and $\alpha_{T'}$ are fixed from now as explained above. This construction is divided into two subparts. We first show that we can extend $\rho_{T'}$ to so called J, k -trivial words (see below), yielding the mapping η_k . We then combine this first extension with β for constructing the final mapping ρ_T . We finally prove that this mapping ρ_T fulfils the induction hypothesis.

Extension to trivial words: the mapping η_k . Let us consider a positive integer k . We first consider the case of J, k -trivial words: an S -word u is J, k -trivial if for all factor v of u of length at least k , $\pi(v) \notin J$. From the definition, every J, k -trivial word u can be (uniquely) decomposed into $u_1 u_2 \dots u_n u'$ in which each u_i has length k and u' has length less than k (possibly null). Using the k -triviality of u , for all i , $\pi(u_i) \in S'$. Hence it makes sense to define:

$$\eta_{T, J, k}(u) = \begin{cases} \pi(u') & \text{if } n = 0 \\ \rho_{T'}(\pi(u_1) \dots \pi(u_k)) \cdot \pi(u') & \text{if } u' \neq \varepsilon \\ \rho_{T'}(\pi(u_1) \dots \pi(u_k)) & \text{otherwise.} \end{cases}$$

Since T and J are fixed from now, we abbreviate $\eta_{T, J, k}$ by just η_k . We also abbreviate $\rho_{T'}$ by simply ρ' . One can remark that the $J, 1$ -trivial S -words are exactly the T' -words, and $\eta_1(u)$ is nothing but $\rho_{T'}(u)$.

We now aim at establishing some properties of η_k . Essentially we prove that the η_k mappings faithfully extend ρ' (Lemma 25) and are consistent one with each other (Corollary 6). We also establish that each η_k satisfy all of the properties of a compatible mapping but stabilisation (properties restricted to J, k -trivial words): monotonicity (Lemma 24), letter and product (a consequence of Lemma 26) and finally substitution (Lemma 29).

Lemma 24. *The mapping η_k is α -monotonic for some α .*

Proof. Let $u \leq v$ be two J, k -trivial words, and let $u_1 \dots u_n u'$ and $v_1 \dots v_n v'$ be their decomposition as for the definition of η_k . For all $i = 1 \dots n$, since $u_i \leq v_i$, we get $\pi(u_i) \leq \pi(v_i)$. In the same way $\pi(u') \leq \pi(v')$. Hence using the definition of η_k and the α -monotonicity of ρ' , we directly obtain $\eta_k(u) \preceq_\alpha \eta_k(v)$. \square

Lemma 25. *For all k , there exists α such that for all S' -word u , $\eta_k(u) \sim_\alpha \rho'(u)$.*

Proof. Let u be an S' -word (in particular it is J, k -trivial). u can be decomposed into $u_1 \dots u_n v$ in which the u_i 's have length k and v' has length less than k . Let us compute:

$$\begin{aligned}\eta_k(u) &= \rho'(\pi(u_1) \dots \pi(u_n)) \cdot \pi(v') \\ &\sim \tilde{\rho}'(\rho'(u_1) \dots \rho'(u_n)) \cdot \rho'(v') \\ &\sim \rho'(u_1 \dots u_n) \cdot \rho'(v') \\ &\sim \rho'(u_1 \dots u_n v') = \rho'(u),\end{aligned}$$

in which we successively use the definition of η_k , Lemma 16 (inside ρ' which is α -monotonic), the substitution rule for ρ' , and Lemma 15. \square

Lemma 26. *There exists α such that for all J, k -trivial word uv ,*

$$\eta_k(uv) \sim_\alpha \eta_k(u) \cdot \eta_k(v) .$$

Proof. (A) We first claim that for all J, k -trivial word ua , $\eta_k(ua) \sim_\alpha \eta_k(u) \cdot a$. If ua is not a multiple of k , we have $\eta_k(ua) = \eta_k(u) \cdot a$ by simply unfolding the definition of η_k . Else, $u = u_1 \dots u_n v$ in which each u_i has length k and v has length $k - 1$. We compute using the substitution rule of ρ' :

$$\begin{aligned}\eta_k(u) \cdot a &= \rho'(\pi(u_1) \dots \pi(u_n)) \cdot \pi(v) \cdot a \\ &\sim \rho'(\pi(u_1) \dots \pi(u_n) \pi(va)) \\ &= \eta_k(ua) .\end{aligned}$$

This completes the proof of (A).

(B) We prove now that $\eta_k(au) \sim_\alpha a \cdot \eta_k(u)$ for all J, k -trivial word au (for a suitable α). Let us assume first that u has a length multiple of $3k$ (B1). In this case, we decompose au as $a_1 u_1 a_2 u_2 \dots a_{3n} u_{3n} a_{3n+1}$ in which every u_i has length $k-1$ (of course $a_1 = a$). We then use the following sequence of equivalences (essentially making successive uses of the substitution rule for ρ' and Lemma 16):

$$\begin{aligned}\eta_k(au) &= \rho'(\pi(a_1 u_1) \dots \pi(a_{3n} u_{3n})) \cdot a_{3n+1} \\ &\sim \rho'(\pi(a_1 u_1 a_2 u_2 a_3 u_3) \dots \pi(a_{3n-2} u_{3n-2} a_{3n-1} u_{3n-1} a_{3n} u_{3n})) \cdot a_{3n+1} \\ &\sim \rho'(\pi(a_1 u_1 a_2) \pi(u_2 a_3 u_3) \dots \pi(a_{3n-2} u_{3n-2} a_{3n-1}) \pi(u_{3n-1} a_{3n} u_{3n})) \cdot a_{3n+1} \\ &\sim \pi(a_1 u_1 a_2) \cdot \rho'(\pi(u_2 a_3 u_3) \dots \pi(a_{3n-2} u_{3n-2} a_{3n-1})) \cdot \pi(u_{3n-1} a_{3n} u_{3n} a_{3n+1}) \\ &\sim \pi(a_1 u_1 a_2) \cdot \rho'(\pi(u_2 a_3 u_3 a_4 u_4 a_5) \dots \pi(u_{3n-4} \dots a_{3n-1})) \cdot \pi(u_{3n-1} a_{3n} u_{3n} a_{3n+1}) \\ &\sim a_1 \cdot \rho'(\pi(u_1 a_2) \dots \pi(u_{3n} a_{3n+1})) \\ &= a \cdot \eta_k(u)\end{aligned}$$

This proves (B1). In order to establish (B) for any word au , one uses $(6k - 2)$ applications of (A) in order to translate the problem to a word of length multiple of $3k$ (this requires up to $3k - 1$ shortenig steps, and as many lengthening steps); and we use (B1).

We now prove the statement of the lemma. Let uv be a J, k -trivial word. If u has a length multiple of k , we have $\eta_k(uv) \sim \eta_k(u) \cdot \eta_k(v)$ by simply using the

definition of η_k and the substitution rule over ρ' . The general case is obtained by using up to $2(k-1)$ -many applications of (B) in order to reduce the computation to the previous case (i.e., up to $(k-1)$ shortening steps in order to reduce u to a length multiple of k , followed by up to $(k-1)$ lengthening step for coming back to the original word uv). We get the expected $\eta_k(uv) \sim \eta_k(u) \cdot \eta_k(v)$. \square

Lemma 27. *For all k , and $u_1 \dots u_n$ a J, k -trivial word such that each u_i has a length multiple of k ,*

$$\eta_k(u_1 \dots u_n) \sim_{\alpha'} \tilde{\rho}'(\eta_k(u_1) \dots \eta_k(u_n)) .$$

Proof. Since each u_i has a length multiple of k , it can be decomposed as $v_i^1 \dots v_i^{n_i}$ in which each v_i^j has a length k . We compute:

$$\begin{aligned} \eta_k(u_1 \dots u_n) &= \eta_k(v_1^1 \dots v_1^{n_1} \dots v_n^1 \dots v_n^{n_n}) \\ &= \rho'(\pi(v_1^1) \dots \pi(v_1^{n_1}) \dots \pi(v_n^1) \dots \pi(v_n^{n_n})) \\ &\sim_{\alpha'} \tilde{\rho}'(\rho'(\pi(v_1^1) \dots \pi(v_1^{n_1})) \dots \rho'(\pi(v_n^1) \dots \pi(v_n^{n_n}))) \\ &= \tilde{\rho}'(\eta_k(u_1) \dots \eta_k(u_n)) \end{aligned}$$

in which we successively use the decomposition of each u_i into v_i^j 's, the definition of η_k , the substitution rule for ρ' and the definition of η_k again. \square

Lemma 28. *For all m , there exists α such that for all J, k -trivial word $u = u_1 \dots u_n$ in which $1 \leq |u_i| \leq m$ for all $i = 1 \dots n$,*

$$\eta_k(u) \sim_{\alpha} \eta_k(\pi(u_1) \dots \pi(u_n)) .$$

Proof. We first establish it the particular case of $m = 2$ (\star). We rephrase it as follows. Let us assume that u is of the form $v_0 a_1 b_1 v_1 \dots a_n b_n v_n$ (the v_i 's are possibly empty), we claim that $\eta_k(u) \sim \eta_k(v_0(a_1 \cdot b_1)v_1 \dots (a_n \cdot b_n)v_n)$.

(A) Let us assume first that $n \leq k$. In this case, it is sufficient to make use up to $(2n-1)$ times of Lemma 26 for establishing the claim.

(B) Let us assume now that $|v_i| \geq k$ for all $i = 1 \dots n-1$ and n is a multiple of k . We prove that $\eta_k(u) \sim \eta_k(v_0(a_1 \cdot b_1)v_1 \dots (a_n \cdot b_n)v_n)$. In this case, one can decompose u as $u_1 \dots u_n$ in such a way that each u_i has a length multiple of k , and that exactly k -many factors $a_i b_i$ occur in each u_j . Let u'_i be the word u_i in which each occurrence of a factor $a_i b_i$ is replaced by $(a_i \cdot b_i)$. Remark that each u'_i has a length multiple of k by construction. Furthermore, using case (A), $\eta_k(u_i) \sim \eta_k(u'_i)$. We compute:

$$\begin{aligned} \eta_k(u) &= \eta_k(u_1 \dots u_n) \\ &\sim \tilde{\rho}'(\eta_k(u_1) \dots \eta_k(u_n)) \\ &\sim \tilde{\rho}'(\eta_k(u'_1) \dots \eta_k(u'_n)) \\ &\sim \eta_k(u'_1 \dots u'_n) = \eta_k(v_0(a_1 \cdot b_1) \dots (a_n \cdot b_n)v_n) \end{aligned}$$

in which we successively use Lemma 27, the equivalence of $\eta_k(u_i)$ and $\eta_k(u'_i)$, and Lemma 27 again.

(C) If we assume $|v_i| \geq k$ but this time not necessarily n being a multiple of k , then the property (\star) holds by applying (A), (B), and Lemma 26.

(D) We now prove (\star) in the general case. For this, we prove by induction on $l = 0 \dots k+1$ that $\eta_k(u_l) \sim \eta_k(u)$ where $u_l = v_0 w_{1,l} v_1 \dots w_{n,l} v_n$ in which $w_{i,l}$ is the word $a_i b_i$ if $(i \bmod k + 1) \geq l$, otherwise $(a_i \cdot b_i)$. It is clear that $u_0 = u$ (base case). It is clear also that transforming u_l into u_{l+1} fulfills the hypothesis of (C), and hence $\eta_k(u) \sim \eta_k(u_l) \sim \eta_k(u_{l+1})$. This completes the induction. Finally, one can remark that $u_{k+1} = v_0(a_1 \cdot b_1)v_1 \dots (a_n \cdot b_n)v_n$, and hence we obtained $\eta_k(u) \sim \eta_k(v_0(a_1 \cdot b_1)v_1 \dots (a_n \cdot b_n)v_n)$. This completes the proof of the claim (\star) .

Let us turn now to the statement of the lemma. Let $u = u_1 \dots u_n$ be J, k -trivial in which $1 \leq |u_i| \leq m$ for all $i = 1 \dots n$. The result is obtained by m successive use of the claim (\star) . \square

Corollary 6. *For all k , there exists α such that $\eta_{k'}(u) \sim_\alpha \eta_k(u)$ for all $k' \leq k$ and all J, k' -trivial word u .*

Proof. Let α be obtained from the above lemma for $m = k$. Let u be a J, k' -trivial word. It is decomposed into $u_1 \dots u_n v$ in which each u_i has length k and v has length less than k' .

$$\begin{aligned} \eta_{k'}(u) &= \rho'(\pi(u_1) \dots \pi(u_n)) \cdot \pi(v) \\ &\sim \eta_k(\pi(u_1) \dots \pi(u_n)) \cdot \eta_k(\pi(v)) \\ &\sim \eta_k(u_1 \dots u_n) \cdot \eta_k(v) \\ &\sim \eta_k(u) , \end{aligned}$$

in which we use the definition of $\eta_{k'}$, Lemma 25, Lemma 28, and Lemma 26. \square

One can finally establish the key lemma concerning η_k .

Lemma 29. *There exists α such that for all J, k -trivial word $u_1 \dots u_n$,*

$$\eta_k(u_1 \dots u_n) \sim_\alpha \tilde{\eta}_k(\eta_k(u_1) \dots \eta_k(u_n)) .$$

Proof. We first establish the statement for products of big factors, i.e., $u = u_1 \dots u_n$ is a J, k -trivial word such that $|u_i| \geq k$ for all $i = 1 \dots n$. In this situation, each u_i can be decomposed in $u_i^1 \dots u_i^{k_i}$ in which $k \leq |u_i^j| < 2k$ for any i, j . By Lemma 28 and its corollary (twice each one), and using the α -monotony of η_k and the substitution rule for ρ' , we compute:

$$\begin{aligned} \eta_k(u) &\sim \eta_k(\pi(u_1^1) \dots \pi(u_1^{k_1}) \dots \pi(u_n^{k_n}) \dots \pi(u_n^{k_n})) \\ &\sim \rho'(\pi(u_1^1) \dots \pi(u_1^{k_1}) \dots \pi(u_n^{k_n}) \dots \pi(u_n^{k_n})) \\ &\sim \tilde{\rho}'(\rho'(\pi(u_1^1) \dots \pi(u_1^{k_1})) \dots \rho'(\pi(u_n^1) \dots \pi(u_n^{k_n}))) \\ &\sim \tilde{\rho}'(\eta_k(u_1) \dots \eta_k(u_n)) \\ &\sim \tilde{\eta}_k(\eta_k(u_1) \dots \eta_k(u_n)) . \end{aligned}$$

Let us turn ourselves no to general case $u = u_1 \dots u_n$. Without loss off generality (i.e., up to $2(k-1)$ uses of Lemma 26), we assume that n is a multiple of k ; $n = mk$, and we set $v_l = u_{(l-1)k+1} \dots u_{lk}$ for all $l = 1 \dots m$. We obtain:

$$\begin{aligned}
\eta_k(u) &= \eta_k(v_1 \dots v_m) \\
&\sim \tilde{\eta}_k(\eta_k(v_1) \dots \eta_k(v_m)) \\
&\sim \tilde{\eta}_k(\pi(\eta_k(u_1) \dots \eta_k(u_k)) \dots \pi(\eta_k(u_{n-k+1}) \dots \eta_k(u_n))) \\
&\sim \tilde{\eta}_k(\eta_k(u_1) \dots \eta_k(u_n)) .
\end{aligned}$$

This concludes the proof. \square

We now establish the second and third item of the induction hypothesis restricted to trivial words.

Lemma 30. *For all k there exists α such that for all smooth S -word u and J, k -trivial word v ,*

- If $u \leq v$, $\beta(u) \preceq_\alpha \eta_k(v)$, and;
- If $v \leq u$, $\eta_k(v) \preceq_\alpha \beta(u)$.

Proof. Assume $u \leq v$. Let us decompose u as $u_1 \dots u_n u'$ and v as $v_1 \dots v_n v'$ in which $|u_1| = |v_1| = \dots = |u_n| = |v_n| = k$, and $|u'| = |v'| < k$.

$$\begin{aligned}
\beta(u) &\sim \beta(\pi(u_1) \dots \pi(u_n)) \cdot \pi(u') \\
&\preceq \rho'(\pi(v_1) \dots \pi(v_n)) \cdot \pi(v') \\
&= \eta_k(v)
\end{aligned}$$

in which one successively use the definition of β , the induction hypothesis for ρ' , and the definition of η_k . The second item is proved in exactly the same way. \square

And we terminate with the uniqueness of the definition.

Lemma 31. *For all k and all mapping ξ compatible with \mathbf{S} , there exists α such that for all J, k trivial word u ,*

$$\eta_k(u) \sim_\alpha \xi(u) .$$

Proof. Let u be decomposed as $u_1 \dots u_n v$ as in the definition of η_k . We compute:

$$\begin{aligned}
\xi(u) &\sim \tilde{\xi}(\xi(u_1) \dots \xi(u_n)) \cdot \xi(v) \\
&\sim \xi(\pi(u_1) \dots \pi(u_n)) \cdot \pi(v) \\
&\sim \rho'(\pi(u_1) \dots \pi(u_n)) \cdot \pi(v) \\
&= \eta_k(u) .
\end{aligned}$$

in which we use the compatibility of ξ , Lemma 16, and the induction hypothesis. \square

Extension to all T -words: the mapping ρ_T . We turn now to the definition of ρ_T itself. Two cases have to be distinguished. If J is not regular, then every S word is $J, 2$ -trivial, and by consequence one can set ρ to be η_2 . The resulting mapping ρ satisfies the induction hypothesis according to the previous lemmas.

Otherwise, the class J is regular. In this case one uses the fact that each word $u \in T^+$ can be uniquely decomposed into $u = u_1 u_2 \dots u_n$ in which each u_i is either J -smooth and maximal, or consists of a single letter in T' . One defines ρ_T over u as:

$$\rho_T(u) = \tilde{\eta}_2(\beta(u_1) \dots \beta(u_n)) .$$

Since T is fixed, we simply denote ρ_T by ρ . Let us first stress the fact that this definition is meaningful. Indeed, since the u_i 's are maximal, $u_i u_{i+1}$ is not J -smooth, and as a consequence, for all m , $(\beta(u_i) \cdot \beta(u_{i+1}))(m)$ belongs to S' , i.e., $\beta(u_1)(m) \dots \beta(u_n)(m)$ is $J, 2$ -trivial. The remaining of the section is devoted to establishing that ρ is compatible with **S**.

Lemma 32. *Let u be a T -word such that $\pi(u) \in T'$, for all m , $\rho(u)(m) \in T'$.*

Proof. A direct consequence of the maximality of smooth words in the decomposition. \square

Lemma 33. *There exists α such that for all T -word $u = u_1 \dots u_n$ with $\pi(u_i) \in T'$ for all $i = 1 \dots n$, then:*

$$\rho(u) \sim_\alpha \tilde{\rho}'(\rho(u_1)\rho(u_2) \dots \rho(u_n))$$

Proof. Let us first remark that since the u_i 's are not J -smooth, no u_i can be a factor of a J -smooth factor of u . As a consequence one can write each u_i as $u_i = v_i u_i^1 \dots u_i^{k_i} v_i'$ in which the v_i 's and v_i' 's are possibly empty, and

$$u_1^1, u_1^2, \dots, u_1^{k_1}, (v_1' v_2), u_2^1, \dots, u_2^{k_2}, (v_2' v_3), \dots, (v_{n-1}' v_n), u_n^1, \dots, u_n^{k_n}$$

is the sequence of maximal J -smooth factors of u (in particular, $v_1 = v_n' = \varepsilon$ and for every $i = 1 \dots, n-1$, $v_i' v_{i+1}$ is nonempty).

We use below the convention that $\beta(\varepsilon) = \varepsilon$ (we can check that one never tries to apply ρ, η_2 and η_3 to the empty word). Let us compute:

$$\begin{aligned} \rho(u) &= \tilde{\eta}_2(\beta(u_1^1) \dots \beta(u_1^{k_1}) \beta(v_1' v_2) \dots \beta(v_{n-1}' v_n) \beta(u_n^1) \dots \beta(u_n^{k_n})) \\ &\sim \tilde{\eta}_3(\beta(u_1^1) \dots \beta(u_1^{k_1}) \beta(v_1' v_2) \dots \beta(v_{n-1}' v_n) \beta(u_n^1) \dots \beta(u_n^{k_n})) \\ &\sim \tilde{\eta}_3(\beta(u_1^1) \dots \beta(u_1^{k_1}) \beta(v_1') \beta(v_2) \dots \beta(v_{n-1}') \beta(v_n) \beta(u_n^1) \dots \beta(u_n^{k_n})) \\ &\sim \tilde{\eta}_3(\eta_3(\beta(u_1^1) \dots \beta(u_1^{k_1}) \beta(v_1')) \eta_3(\beta(v_2) \dots \beta(v_2')) \dots \eta_3(\beta(v_n) \beta(u_n^1) \dots \beta(u_n^{k_n}))) \\ &\sim \tilde{\eta}_2(\eta_2(\beta(u_1^1) \dots \beta(u_1^{k_1}) \beta(v_1')) \eta_2(\beta(v_2) \dots \beta(v_2')) \dots \eta_2(\beta(v_n) \beta(u_n^1) \dots \beta(u_n^{k_n}))) \\ &= \tilde{\eta}_2(\rho(u_1) \rho(u_2) \dots \rho(u_n)) \\ &\sim \tilde{\rho}'(\rho(u_1) \rho(u_2) \dots \rho(u_n)) \end{aligned}$$

in which we successively use the definition of ρ , Corollary 6, Lemma 21 (naturally extended to ε words), Lemma 29, Corollary 6 again together with Lemma 24, the definition of ρ , and finally Lemma 25. \square

Corollary 7. *There exists α such that for all T -word $u = u_1 \dots u_n$ with $\pi(u_i u_{i+1}) \in T'$ for all $i = 1 \dots n - 1$, then:*

$$\rho(u) \sim_\alpha \tilde{\eta}'_2(\rho(u_1)\rho(u_2) \dots \rho(u_n)) .$$

Lemma 34. *For all T -word u and all n , $\rho(u)(n) \leq \pi(u)$.*

Proof. The property holds for η_2 by definition and induction hypothesis (use Corollary 5). In combination with $\beta(v) \leq \pi(v)$ (directly from the definition) for all J -smooth words (directly from the definition), we get the expected result. \square

Lemma 35. *There exists α such that for all J -smooth word $u = u_1 \dots u_n$*

$$\beta(u_1 \dots u_n) \sim_\alpha \tilde{\rho}(\beta(u_1) \dots \beta(u_n))$$

Proof. If J is irregular, then u has length 1, and the statement is obvious.

If J is regular and stable, then $\beta(u_i) = \pi(u_i)$ for all i . Furthermore $\pi(u_1) \dots \pi(u_n)$ is also J -smooth. Hence $\beta(u) = \pi(u) = \pi(\pi(u_1) \dots \pi(u_n)) = \rho(\beta(u_1) \dots \beta(u_n))$.

The interesting case is when J is unstable. Let $u = u_1 \dots u_n$, and let U be the set of words $(\pi(u_1) + \pi(u_1)^\sharp) \dots (\pi(u_n) + \pi(u_n)^\sharp)$. (Remark that $\beta(u_1) \dots \beta(u_n)$ is an id -nondecreasing sequence over $(U, <)$).

Let $v \in U$. We claim first that $\pi(u)^\sharp \preceq \rho(v)$. Indeed, let us factorize v into $v_1 \dots v_k$ according to the definition of ρ . One has $\pi(u)^\sharp = \pi(\pi(v_1)^\sharp \dots \pi(v_k)^\sharp)$ by Lemma 7. Since J^\sharp is stable, using the above case, $\pi(\pi(v_1)^\sharp \dots \pi(v_k)^\sharp) \sim \eta_2(\pi(v_1)^\sharp \dots \pi(v_k)^\sharp)$. Finally, we have $\pi(v_i)^\sharp \leq \pi(v_i)$ for all i , and using Lemma 24 we get $\eta_2(\pi(v_1)^\sharp \dots \pi(v_k)^\sharp) \preceq \eta_2(\pi(v_1) \dots \pi(v_k)) = \rho(v)$. Summing up, $\pi(u)^\sharp \preceq \rho(v)$.

Let now $v \in U$. If $v \neq \pi(u_1) \dots \pi(u_n)$, then at least one letter in v is of the form $\pi(u_i)^\sharp$. It follows by Lemma 7 that $\pi(v) = \pi(u)^\sharp$. Using Lemma 34, we get $\rho(v) \leq \pi(v)$. Together with the above claim, we obtain $\rho(v) \sim \pi(v)$. If $v = \pi(u_1) \dots \pi(u_n)$, then v is J -smooth and as a consequence, $\rho(v) \sim \beta(v)$. Overall we get that for all $v \in U$,

$$\rho(v) \sim \begin{cases} \beta(v) & \text{if } v = \pi(u_1) \dots \pi(u_n) \\ \pi(u)^\sharp & \text{otherwise.} \end{cases}$$

And the latter mapping is id -monotonic (from $(U, <)$ to S^ω).

Using Proposition 11, one gets:

$$\begin{aligned} \tilde{\rho}(\beta(u_1) \dots \beta(u_n)) &\sim \begin{cases} \beta(v) & \text{if } v = \pi(u_1) \dots \pi(u_n) \\ \pi(u)^\sharp & \text{otherwise.} \end{cases} \\ &= \pi(u)^\sharp|_{\max(n, \max_{i=1 \dots n} |u_i|)} \pi(u) \\ &= \beta(u) \\ &\sim \tilde{\eta}'_2(\beta(u)) = \rho(u) \end{aligned}$$

(in which we use Example 4 together with Proposition 12). This concludes the proof. \square

Lemma 36. *There exists α such that for all $u = u_1 \dots u_n$ be a T -word in which each u_i is either J -smooth, or restricted to a single letter.*

$$\rho(u_1 \dots u_n) \sim_\alpha \tilde{\rho}(\beta(u_1) \dots \beta(u_n))$$

Proof. Let us decompose u into $v_1 \dots v_k$ as in the definition of ρ . By assumption on the u_i 's, each u_i is a factor of one of the v_j 's. Hence, one can write v_j as $u_{h(j-1)+1} \dots u_{h(j)}$ in which $0 = h(0) < h(1) \dots < h(k) = n$. Let us compute:

$$\begin{aligned} \rho(u) &= \tilde{\eta}_2(\beta(v_1) \dots \beta(v_n)) \\ &\sim \tilde{\eta}_2(\tilde{\rho}(\beta(u_0) \dots \beta(u_{h(1)}) \dots \tilde{\rho}(\beta(u_{h(k-1)+1}) \dots \beta(u_{h(k)})))) \\ &\sim \rho'(\tilde{\rho}(\beta(u_0) \dots \beta(u_{h(1)}) \dots \tilde{\rho}(\beta(u_{h(k-1)+1}) \dots \beta(u_{h(k)})))) \\ &\sim \rho(\beta(u_1) \dots \beta(u_n)) \end{aligned}$$

in which we successively use the definition of ρ , Lemma 35 together with the monotonicity of η_2 , Lemma 25, and Lemma 33. \square

Lemma 37. *There exists α such that for all smooth S -word u and all T -word v ,*

- *If $u \leq v$, $\beta(u) \preceq_\alpha \rho(v)$, and;*
- *If $v \leq u$, $\rho(v) \preceq_\alpha \beta(u)$.*

Proof. We remark first that if $\pi(u)$ is irregular, u has length 1, and both statements are obvious. Let us consider the case when $\pi(u)$ regular.

First item. Let $v = v_1 \dots v_k$ be the decomposition of v as in the definition of ρ , and $u = u_1 \dots u_k$ be the corresponding decomposition of u . We assume $u \leq v$.

We first prove that $\pi(u)^\sharp \leq \rho(v)$. In this case, for all $i = 1 \dots k$, $\pi(u_i)^\sharp \leq \beta(v_i)$ according to Lemma 20. Let us compute:

$$\begin{aligned} \pi(u)^\sharp &= \beta(\pi(u_1)^\sharp \dots \pi(u_k)^\sharp) \\ &\preceq \tilde{\eta}_2(\beta(v_1) \dots \beta(v_k)) \\ &= \rho(v) , \end{aligned}$$

in which we use Lemma 30.

In the general case, let α be taken from Lemma 30. Let $\alpha(m) \leq n$. We prove that $\beta(u)(m) \leq \rho(v)(n)$. Two cases can happen. If $\beta(u)(m) = \pi(u)^\sharp$, then $\beta(u)(m) = \pi(u)^\sharp \leq \rho(v)(n)$ by the case above. Otherwise, $|u| \leq m$. This implies $|u_i| \leq m$ for all $i = 1, \dots, k$ (which itself implies $\beta(v_i)(n) = \pi(v_i) \geq (\pi(u_i))$), and $k \leq m$. Let us compute.

$$\begin{aligned} \beta(u)(m) &= \pi(u) \leq \eta_2(\pi(v_1) \dots \pi(v_k))(n) \\ &= \rho(u)(n) \end{aligned}$$

by Lemma 30.

Second item. Let $v = v_1 \dots v_k$ be the decomposition of v as in the definition of ρ , and $u = u_1 \dots u_k$ be the corresponding decomposition of u . We assume $v \leq u$. We first remark that $\rho(v) \leq \pi(v) \leq \pi(u)$.

Let α be taken from Lemma 30. Let $\alpha(\alpha(m))^2 \leq n$. We aim at $\rho(v)(m) \leq \beta(u)(n)$. Two cases can happen. Either $|u| \leq n$, and in this case $\beta(u)(n) = \pi(u)$. In this case the inequality follows from the above remark. Otherwise, either $n \geq \alpha(\alpha(m))$ or $|u_i| \geq \alpha(m)$ for some $i = 1, \dots, k$. If $n \geq \alpha(\alpha(m))$, then

$$\begin{aligned} \rho(v)(m) &\leq \eta_2(\pi(v_1) \dots \pi(v_k))(\alpha(m)) \\ &\preceq \beta(\pi(u_1) \dots \pi(u_k))(\alpha(\alpha(m))) \\ &= \pi(u)^\sharp \end{aligned}$$

Otherwise $|u_i| \geq \alpha(m)$ for some i .

$$\begin{aligned} \rho(v)(m) &\leq \eta_2(\pi(v_1) \dots \pi(v_{i-1})\pi(v_i)^\sharp(\pi(v_{i+1}) \dots))(\alpha(m)) \\ &\leq \pi(\pi(v_1) \dots \pi(v_{i-1})\pi(v_i)^\sharp(\pi(v_{i+1}) \dots)) \\ &\leq \pi(u)^\sharp. \end{aligned}$$

□

Lemma 38. *There exists α such that ρ is α -monotonic.*

Proof. Let $u \leq v$ be two T -words. We decompose simultaneously u as $u_1 \dots u_n$ and $v = v_1 \dots v_n$ in which u_i and v_i are maximal such that either $\pi(u_i) \in D$ or $\pi(v_i) \in D$. By Lemma 37, $\rho(u_i) \preceq \rho(v_i)$. Furthermore $(\rho(u_1) \dots \rho(u_n))(m)$ is $J, 2$ -trivial for all m (and the same goes for v). Hence we easily compute:

$$\begin{aligned} \rho(u) &= \rho(u_1 \dots u_n) \\ &\sim \tilde{\eta}'_2(\rho(u_1) \dots \rho(u_n)) \\ &\preceq \tilde{\eta}'_2(\rho(v_1) \dots \rho(v_n)) \\ &\sim \rho(v) \end{aligned}$$

in which we use Corollary 7, the monotonicity of η_2 , and Corollary 7 again. □

Lemma 39. *There exists α such that for all S -word $u = u_1 \dots u_n$,*

$$\rho(u) \sim_\alpha \tilde{\rho}(\rho(u_1) \dots \rho(u_n)).$$

Proof. Let $0 = h(0) < h(1) < \dots < h(k) = n$ be such that for all i , i is maximal such that $\pi(u_{h(i-1)+1} \dots u_{u_i}) \in D$, or $h(i-1) + 1 = h(i)$. One knows by Lemma 35 that for all i , $\rho(u_{h(i-1)+1} \dots u_{u_i}) = \tilde{\rho}(\rho(u_{h(i-1)+1}) \dots \rho(u_{u_i}))$.

$$\begin{aligned} \rho(u) &= \rho(u_1 \dots u_n) \\ &\sim \tilde{\eta}_2(\rho(u_{h(0)+1} \dots u_{h(1)}) \dots \rho(u_{h(k-1)+1} \dots u_{h(k)})) \\ &\sim \tilde{\eta}_2(\tilde{\rho}(\rho(u_{h(0)+1}) \dots \rho(u_{h(1)})) \dots \tilde{\rho}(\rho(u_{h(k-1)+1}) \dots \rho(u_{h(k)}))) \\ &\sim \tilde{\rho}(\rho(u_1) \dots \rho(u_n)) \end{aligned}$$

in which we successively use Corollary 7, the above equivalence together with the monotonicity of η_2 , and Corollary 7 again. □

What remains to be proved is the uniqueness of the compatible mapping. This is the subject of Lemma 40.

Lemma 40. *Let ξ be a mapping compatible with \mathbf{S} , there exists α such that for all T -word u , $\rho(u) \sim_\alpha \xi(u)$.*

Proof. Let u be a T -word decomposed as $u_1 \dots u_n$ as in the definition of ρ .

$$\begin{aligned} \xi(u) &= \tilde{\xi}(\xi(u_1) \dots \xi(u_n)) \\ &\sim \tilde{\xi}(\beta(u_1) \dots \beta(u_n)) \\ &\sim \tilde{\eta}_2(\beta(u_1) \dots \beta(u_n)) \\ &= \rho(u) \end{aligned}$$

in which we use the compatibility of ξ together with Lemma 22, Lemma 32 and Lemma 31. □

4 Recognisable cost functions

In this section, we put in action the notions developed in the previous section, and use them for an algebraic description of cost functions: recognisable cost functions.

The section is organised as follows. In Section 4.1 we define recognisable cost functions. We present some elementary closure properties for them in Section 4.2. In Section 4.3 we introduce the very useful tool of \sharp -expressions, and present some results on sub-stabilisation monoids. In Section 4.4, we show how we can prove decidability results using those arguments. We prove in Section 4.5 that recognisable cost functions are closed under inf-projection and sup-projection. We then show how one can transform a cost automaton into an equivalent recognisable cost function in Section 4.6. We establish the converse translation in Section 4.7.

4.1 Recognisability

We now define the notion of recognisability for cost-functions.

Given a stabilisation monoid $\mathbf{M} = \langle M, \cdot, \leq, \sharp \rangle$, a length-preserving morphism h from \mathbb{A}^* to M^* , and an ideal $I \subseteq M$, the triple \mathbf{M}, h, I *recognises* the mapping $f : \mathbb{A}^* \rightarrow \omega + 1$ if there exists α such that for all $u \in \mathbb{A}^*$,

$$f(u) \approx_\alpha \sup\{n + 1 : \rho(h(u))(n) \in I\}$$

in which ρ is a mapping compatible with \mathbf{M} . A cost function from \mathbb{A}^* to $\omega + 1$ is *recognisable* if some (equivalently all) function in the class is recognised by some \mathbf{M}, h, I . Remark that by Corollary 3, this definition would have been the same if we used:

$$f(u) \approx_\alpha \inf\{n : \rho(h(u))(n) \notin I\}.$$

Example 11. For $\mathbb{A} = \{a, b\}$, the function $|\cdot|_a$ which counts the number of occurrences of a in a word is recognisable. For this, consider the monoid of Example 10, the morphism defined by $h(a) = a, h(b) = b$, and the ideal $I = \{\perp\}$. Then we have $|u|_a = \inf\{n : \rho(h(u))(n) \notin I\}$ for all $u \in \mathbb{A}^*$. This means that $|\cdot|_a$ is recognisable.

Remark 7. Pursuing Remark 5, imagine you construct the stabilisation monoid $\mathbf{M}_\#$ out of a monoid $\mathbf{M} = \langle M, \cdot \rangle$. This monoid can serve for recognising the language $L = h^{-1}(A) \subseteq \mathbb{A}^*$ in which $A \subseteq M$, and h is a morphism from \mathbb{A}^* to M (\mathbb{A} being some alphabet).

We can construct the length preserving morphism h' from \mathbb{A}^* to M^* which to each letter $a \in \mathbb{A}$ associates $h(a) \in M$. The morphism h and h' are related by the relation $h(u) = \pi(h'(u))$ for all $u \in \mathbb{A}^*$. We have also seen in Remark 5 that the mapping ρ which to every $v \in M^*$ and $n \in \omega$ associates $\rho(v)(n) = \pi(v)$ is compatible with $\mathbf{M}_\#$. Finally, since the order in $\mathbf{M}_\#$ is reduced to the equality, every subset of M is an ideal; in particular A is an ideal. Hence it makes sense to consider the cost function recognised by $\mathbf{M}_\#, h', A$:

$$\sup\{n : \rho(h'(u))(n) \in A\} = \sup\{n : \pi(h(u)) \in A\} = \chi_{\mathbb{A}^* \setminus L}.$$

In other words, if a language is recognised by a monoid, the same monoid translated into a stabilisation monoid recognises the characteristic function of the language.

4.2 Elementary closure properties.

We show here some straightforward operations on recognisable cost functions: first under composition with a morphism, then under min and max.

Composition with morphism The closure of recognisable cost functions under composition with a morphism is the counterpart to the preservation of recognisable language under inverse morphism in standard language theory. The construction is the same:

Lemma 41. *Let f be a recognisable cost function over \mathbb{A}^* , let \mathbb{B} be an alphabet, and p be a morphism from \mathbb{B}^* to \mathbb{A}^* , then the cost function $f \circ p$ over \mathbb{B}^* is recognisable.*

Proof. Assume f is recognised by \mathbf{M}, h, I , then construct h' from \mathbb{B} to \mathbf{M} defined by $h'(a) = \pi(h(p(a)))$. It is easy to see that \mathbf{M}, h', I recognises $f \circ p$. \square

Closure under min and max. The composition under min and max of recognisable cost functions is the counterpart to the closure under intersection and union in standard language theory. As a consequence it is based on the product of algebraic structures.

Let $\mathbf{M}_1 = \langle M_1, \cdot_1, \leq_1, \#_1 \rangle$ and $\mathbf{M}_2 = \langle M_2, \cdot_2, \leq_2, \#_2 \rangle$ be stabilisation monoids, then the *product* $\mathbf{M}_1 \times \mathbf{M}_2$ is the tuple $\langle M_1 \times M_2, \cdot, \leq, \# \rangle$ such that for all $a_1, b_1 \in$

M_1 and $a_2, b_2 \in M_2$, $(a_1, a_2) \cdot (b_1, b_2) = (a_1 \cdot b_1, a_2 \cdot b_2)$, $(a_1, a_2) \leq (b_1, b_2)$ if $a_1 \leq b_1$ and $a_2 \leq b_2$, and for all idempotents $e_1 \in M_1$ and $e_2 \in M_2$, $(e_1, e_2)^\sharp = (e_1^{\sharp_1}, e_2^{\sharp_2})$. We have:

Lemma 42. *If \mathbf{M}_1 and \mathbf{M}_2 are stabilisation monoids, then $\mathbf{M}_1 \times \mathbf{M}_2$ is a stabilisation monoid.*

Furthermore, if ρ_1 is a mapping compatible with \mathbf{M}_1 and ρ_2 is mapping compatible with \mathbf{M}_2 then ρ defined for all words $u \in (M_1 \times M_2)^$ and $n \in \omega$ by:*

$$\rho(u)(n) = (\rho_1(u_1)(n), \rho_2(u_2)(n)) ,$$

in which u_i for $i = 1, 2$ is obtained from u by projecting each letter to its i th component, is compatible with $\mathbf{M}_1 \times \mathbf{M}_2$.

Proof. Everything follows from the definitions. □

From this we can derive that any two recognisable cost functions can be recognised by the same monoid.

Corollary 8. *If f_1 and f_2 are recognisable cost functions over the alphabet \mathbb{A}^* , there exists a stabilisation monoid \mathbf{M} of elements M , a length preserving morphism from \mathbb{A}^* to M^* and two ideals I_1 and I_2 such that f_1 is recognised by \mathbf{M}, h, I_1 and f_2 by \mathbf{M}, h, I_2 .*

Proof. Assume f_1 is recognised by \mathbf{M}_1, h_1, J_1 and f_2 by \mathbf{M}_2, h_2, J_2 , then using Lemma 42, we easily obtain that f_1 is recognised by $\mathbf{M}_1 \times \mathbf{M}_2, h, J_1 \times J_2$ and f_2 is recognised by $\mathbf{M}_1 \times \mathbf{M}_2, h, M_1 \times J_2$, in which $h(a) = (h_1(a), h_2(a))$ for all $a \in \mathbb{A}$, M_1 is the set of elements of \mathbf{M}_1 and M_2 is the set of elements of \mathbf{M}_2 . □

Corollary 9. *Recognisable cost functions are closed under min and max.*

Proof. Let f_1 and f_2 be cost functions over the alphabet \mathbb{A}^* recognised by \mathbf{M}, h, I_1 and \mathbf{M}, h, I_2 respectively (thanks to Corollary 8). By definition of recognisability, we directly have that $\min(f_1, f_2)$ is recognised by $\mathbf{M}, h, I_1 \cap I_2$, and $\max(f_1, f_2)$ is recognised by $\mathbf{M}, h, I_1 \cup I_2$. □

4.3 On \sharp -expressions and sub-stabilisation monoids

\sharp -expressions are a very useful tool in the analysis of stabilisation monoids. Those expressions were introduced by Hashiguchi [17].

A \sharp -expression over A is an expression constructed over the set A using formal operators of concatenation and stabilisation. I.e., all $a \in A$ is a \sharp -expression over A ; if ϕ, ψ are \sharp -expressions over A , then so is $\phi\psi$; and if ϕ is a \sharp -expression over A , then so is ϕ^\sharp . The set of \sharp -expression over A is $\text{exp}(A)$. A \sharp -expression is *strict* if it contains at least one \sharp operator. Non-strict \sharp -expressions are identified with words over A . The *size* of a \sharp -expression is the number of operators it contains.

For an integer k , we define $\text{unfold}_k(\phi) \in A^*$ inductively by $\text{unfold}_k(a) = a$, $\text{unfold}_k(\phi\psi) = \text{unfold}_k(\phi)\text{unfold}_k(\psi)$, and $\text{unfold}_k(\phi^\sharp) = (\text{unfold}_k(\phi))^{k+1}$.

We now introduce typed \sharp -expressions, i.e., \sharp -expressions which can be evaluated in a stabilisation monoid. Let $\mathbf{M} = \langle M, \cdot, \leq, \sharp \rangle$ be a stabilisation monoid, and $A \subseteq M$. One defines $\text{type}(\phi)$ as the partial mapping from $\text{exp}(A)$ to M defined by:

- $\text{type}(a) = a$,
- $\text{type}(\phi\psi) = \text{type}(\phi) \cdot \text{type}(\psi)$,
- $\text{type}(\phi^\sharp) = (\text{type}(\phi))^\sharp$ if $\text{type}(\phi)$ is an idempotent, otherwise it is undefined.

A \sharp -expression ϕ is *typed* if $\text{type}(\phi)$ is defined.

Lemma 43. *For all stabilisation monoid $\mathbf{M} = \langle M, \cdot, \leq, \sharp \rangle$, all mapping ρ compatible with \mathbf{M} , and all typed \sharp -expression ϕ over M , there exists α such that for all $k \in \omega$,*

$$\rho(\text{unfold}_k(\phi)) \sim_\alpha \text{type}(\phi)|_k \pi(\text{unfold}_k(\phi)) .$$

Proof. The proof is by an elementary induction on the \sharp -expression relying on the properties of compatible mappings. \square

Lemma 44. *For all stabilisation monoid $\mathbf{M} = \langle M, \cdot, \leq, \sharp \rangle$ and all mapping ρ compatible with \mathbf{M} , there exists a correction function α such that for all word $u \in M^*$, there exists two strict typed \sharp -expressions $\phi, \psi \in \text{exp}(\text{letters}(u))$ such that:*

$$\text{type}(\phi)|_{|u|} \pi(u) \preceq_\alpha \rho(u) \preceq_\alpha \text{type}(\psi)|_{|u|} \pi(u) .$$

Given a stabilisation monoid $\mathbf{M} = \langle M, \cdot, \leq, \sharp \rangle$, a *sub-stabilisation monoid* is a sub-stabilisation semigroup which contains the neutral element. Given some subset A of a stabilisation monoid, we denote by $\langle A \rangle^\sharp$ the least sub-stabilisation monoid which contains A . It is easy to see that:

Fact 45 $\langle A \rangle^\sharp = \{ \text{type}(\phi) : \phi \in \text{exp}(A), \phi \text{ is typed} \} .$

Our last lemma shows that we can make every computation intern to sub-stabilisation monoids.

Lemma 46. *For all stabilisation monoid $\mathbf{M} = \langle M, \cdot, \leq, \sharp \rangle$ there exists a compatible mapping ρ such that for all word $u \in A^*$ for some $A \subseteq M$,*

$$\rho(u) \in (\langle A \rangle^\sharp)^\omega .$$

Proof. For all word $u \in M^*$, denote by $\text{letters}(u)$ the set of letters appearing in u . For all sub-stabilisation monoid $H \subseteq M$, let ρ_H be a mapping compatible with M restricted to H (this is possible according to Theorem 2).

We construct the mapping ρ as follows. For all $u \in M^*$,

$$\rho(u) = \rho_{\langle \text{letters}(u) \rangle^\sharp}(u) .$$

By construction, for all $u \in A^*$ for some $A \subseteq M$ we have $\rho(u) \in (\langle \text{letters}(u) \rangle^\sharp)^\omega \subseteq (\langle A \rangle^\sharp)^\omega$.

Let now ξ be some mapping compatible with \mathbf{M} (there exists some according to Theorem 2). By Lemma 18, for all sub-stabilisation monoid H , there is a correction function α_H such that $\xi(u) \sim_{\alpha_H} \rho_H(u)$ for all $u \in H^*$. Since there are only finitely many stabilisation monoids, we can take α to be the supremum of the α_H 's correction functions and obtain that $\xi(u) \sim_{\alpha} \rho(u)$ for all $u \in M^*$. Hence, ρ is equivalent to a compatible mapping. This means that it is compatible itself. \square

4.4 Decidability

In this section, we establish that the relation \preceq is decidable over recognizable cost functions, using a very simple saturation argument.

Theorem 3. *The relation \preceq is decidable over recognisable cost functions.*

Proof. Let f_1 and f_2 be mappings recognisable by \mathbf{M}, h, I_1 and \mathbf{M}, h, I_2 respectively (this is possible according to Corollary 8). Let ρ be a mapping compatible with \mathbf{M} . Let $A = h(\mathbb{A})$, i.e., the elements in the stabilisation monoid that are images of letters. We claim that

$$f_1 \preceq f_2 \quad \text{iff} \quad (I_1 \setminus I_2) \cap \langle A \rangle^\sharp = \emptyset$$

Assuming the claim is true, we directly derive from it a decision procedure for the relation \preceq : compute $\langle A \rangle^\sharp$ by a natural saturation argument, and check the emptiness of $(I_1 \setminus I_2) \cap \langle A \rangle^\sharp$. Let us prove the claim.

First direction. Assume $(I_1 \setminus I_2) \cap \langle A \rangle^\sharp = \emptyset$. Using Lemma 46, one assumes that ρ maps A^* to $(\langle A \rangle^\sharp)^\omega$. Consider now a word $u \in \mathbb{A}^*$. Since $\rho(h(u)) \in \langle A \rangle^\sharp$, and $(I_1 \setminus I_2) \cap \langle A \rangle^\sharp = \emptyset$, for all $n \in \omega$ we have:

$$\rho(h(u))(n) \in I_1 \quad \rightarrow \quad \rho(h(u))(n) \in I_2 .$$

As a consequence,

$$\begin{aligned} f_1(u) &\approx_{\alpha} \sup\{n : \rho(h(u)) \in I_1\} && \text{(def recognisability)} \\ &\leq \sup\{n : \rho(h(u)) \in I_2\} && \text{(above remark)} \\ &\approx_{\alpha} f_2(u) . && \text{(def recognisability)} \end{aligned}$$

Hence $f_1 \preceq f_2$.

Other direction. Assume that $(I_1 \setminus I_2) \cap \langle A \rangle^\sharp \neq \emptyset$. By Fact 45, there exists a \sharp -expression ϕ over A such that $\text{type}(\phi) \in I_1 \setminus I_2$. One can turn ϕ into a \sharp -expression ψ over \mathbb{A} by replacing each letter $a \in A$ in ϕ by some $a' \in \mathbb{A}$ such that $h(a') = a$ (this is possible by definition of A). We quiet naturally have for all $k \in \omega$, $h(\text{unfold}_k(\psi)) = \text{unfold}_k(\phi)$. We can estimate the value of $f_1(\text{unfold}_k(\psi))$ for $k \in \omega$, and get:

$$\begin{aligned} f_1(\text{unfold}_k(\psi)) &\approx \sup\{n : \rho(\text{unfold}_k(\phi))(n) \in I_1\} && \text{(def recognisability)} \\ &\approx \sup\{n : (\text{type}(\phi)|_k \pi(\text{unfold}_k(\phi)))(n) \in I_1\} && \text{(by Lemma 43)} \\ &\geq k . && \text{(since } \text{type}(\phi) \in I_1) \end{aligned}$$

Similarly, we can estimate the value of $f_1(\text{unfold}_k(\psi))$ for $k \in \omega$, and get:

$$\begin{aligned} f_2(\text{unfold}_k(\psi)) &\approx \inf\{n : \rho(\text{unfold}_k(\phi))(n) \notin I_2\} && \text{(def recognisability)} \\ &\approx \inf\{n : (\text{type}(\phi)|_k \pi(\text{unfold}_k(\phi)))(n) \notin I_2\} && \text{(by Lemma 43)} \\ &= 0 . && \text{(since } \text{type}(\phi) \notin I_2 \text{)} \end{aligned}$$

Hence, f_2 is bounded over $\{\text{unfold}_k(\psi) : k \in \omega\}$, while f_1 is not bounded over the same set. According to Proposition 6, this is a contradiction to $f_1 \preceq f_2$. \square

This decidability result extends previous results. For instance, the *boundedness problem* (deciding the existence of $n \in \omega$ such that $f(u) \leq n$ for all words u) corresponds to $f \preceq 0$. The standard *limitedness problem* (the boundedness over the support of the function) corresponds to $f \preceq \chi_{\text{support}(f)}$ (remind that $\text{support}(f) = \{u : f(u) < \omega\}$ and is regular according to Proposition 3).

4.5 Closure under inf-projection and sup-projection

In this section, we devise two forms of powerset construction for stabilisation monoid: the stabilisation monoid of ideals and the stabilisation monoid of coideals. Those two constructions allow to prove the closure of recognisable cost functions under inf-projection and sup-projection respectively.

The stabilisation monoid of ideals and inf-projection. Consider a stabilisation monoid $\mathbf{M} = \langle M, \cdot, \leq, \# \rangle$. We aim at constructing a new stabilisation monoid $\downarrow(\mathbf{M})$, each element of which is an ideal of M . The validity of the construction is established by Lemma 47. We then construct the corresponding compatible mapping, the correction of which is the subject of Lemma 49. We finally conclude with Proposition 14 which shows that this construction can be used for computing the inf-projection of a recognisable cost function.

Remind that an ideal of M is a subset $I \subseteq M$ such that for all $x \leq y$, if $y \in I$ then $x \in I$, and that we denote by $\downarrow(M)$ the set of ideals of M . Given a set $A \subseteq M$, one denotes by $A\downarrow$ the least ideal which contains it, i.e., the set $\{y : \bar{y} \leq x \in A\}$. Given ideals $A, B \in \downarrow(M)$, their product is defined in the usual way by:

$$A \cdot B = \{a \cdot b : a \in A, b \in B\}\downarrow ,$$

Let $E \in \downarrow(M)$ be an idempotent, one defines $E^\#$ by:

$$E^\# = \{a \cdot e^\# \cdot b : a, e, b \in E, e = e \cdot e\}\downarrow .$$

Our first result state the correction of this construction.

Lemma 47. $\downarrow(\mathbf{M}) = \langle \downarrow(M), \subseteq, \cdot, \# \rangle$ is a stabilisation monoid.

Proof. The fact that $\langle \downarrow(M), \subseteq, \cdot \rangle$ is an ordered monoid is standard and with no difficulty, the neutral element being $\{1\}\downarrow$. The fact that for all idempotent ideals $E \subseteq F$, $E^\# \subseteq F^\#$ is also obvious from the definition (call this *Claim 0*).

Let $E \in \downarrow(M)$ be an idempotent.

Claim 1: $E^\# \subseteq E$. Indeed, let $a \in E^\#$, i.e., $a \leq x \cdot e^\# \cdot y$ for $x, e, y \in E$ with e idempotent. Since E is idempotent, $x \cdot e \cdot y \in E$. Hence $a \leq x \cdot e^\# \cdot y \leq x \cdot e \cdot y \in E$ which implies that $a \in E$ since E is an ideal.

Claim 2: $E^\# \cdot E \subseteq E^\#$. Let $a \in E^\#$ and $a' \in E$. By definition, $a \leq b \cdot e^\# \cdot c$ for some $b, e, c \in E$ with e idempotent. Since E is an idempotent, $(c \cdot a') \in E$. Hence, $a \cdot a' \leq b \cdot e^\# \cdot (c \cdot a') \in E^\#$.

Claim 3: $E^\# \subseteq E^\# \cdot E^\#$. Let $a \in E^\#$. This means that $a \leq b \cdot e^\# \cdot c$ for some $a, e, b \in E$. Let $x = b \cdot e^\# \cdot e$ and $y = e \cdot e^\# \cdot c$, then both $x \in E^\#$ and $y \in E^\#$. Hence $a \leq x \cdot y \in E^\# \cdot E^\#$.

Claim 4: $E^\# \cdot E^\# = E^\# \cdot E = E^\#$. Indeed, $E^\# \cdot E^\# \subseteq E^\# \cdot E \subseteq E^\#$ (by claim 1, compatibility of product, and claim 2), and $E^\# \subseteq E^\# \cdot E^\#$ (by claim 3).

Claim 5: $(E^\#)^\# = E^\#$. We know that $E^\# \subseteq E$ (by claim 1), and hence $(E^\#)^\# \subseteq E^\#$ (by claim 0). Conversely, let $a \in E^\#$. This means $a \leq b \cdot e^\# \cdot c$ for some $a, e, b \in E$ with e idempotent. We know that $a \cdot e^\# = a \cdot e^\# \cdot e \in E^\#$, $e^\# = e \cdot e^\# \cdot e^\# \in E^\#$, and $e^\# \cdot c = e \cdot e^\# \cdot e \in E^\#$. Furthermore, $e^\#$ is an idempotent and $e^\# = (e^\#)^\#$. Hence $a \leq b \cdot e^\# \cdot c = (a \cdot e^\#) \cdot (e^\#)^\# \cdot (e^\# \cdot c) \in (E^\#)^\#$.

Let us now consider $A, B \in \downarrow(M)$ such that $A \cdot B$ and $B \cdot A$ are idempotents.

Claim 6: $(A \cdot B)^\# \subseteq A \cdot (B \cdot A)^\# \cdot B$. Let $x \in (A \cdot B)^\#$. This means that $x \leq (a \cdot b) \cdot (a' \cdot b')^\# \cdot (a'' \cdot b'')$ for some $a, a', a'' \in A, b, b', b'' \in B$ with $a' \cdot b'$ idempotent. Consider the element $e = b' \cdot a' \cdot b' \cdot a'$. We have $e \cdot e = (b' \cdot a' \cdot b' \cdot a') \cdot (b' \cdot a' \cdot b' \cdot a') = e$ since $a' \cdot b'$ is an idempotent. It follows that

$$\begin{aligned} x &\leq (a \cdot b) \cdot (a' \cdot b')^\# \cdot (a'' \cdot b'') && \text{(choice of } a, b, a', b', a'', b'') \\ &= a \cdot b \cdot (a' \cdot b' \cdot a' \cdot b')^\# \cdot a'' \cdot b'' && \text{(idempotency of } a' \cdot b') \\ &= a \cdot b \cdot a' \cdot (b' \cdot a' \cdot b' \cdot a')^\# \cdot b' \cdot a' \cdot b' \cdot a'' \cdot b'' && \text{(consistency)} \\ &\in A \cdot B \cdot A \cdot (B \cdot A \cdot B \cdot A)^\# \cdot B \cdot A \cdot B \cdot A \cdot B && \text{(def of } \cdot \text{ and } \#) \\ &= A \cdot (B \cdot A)^\# \cdot B. && \text{(idempotency of } B \cdot A, \text{ and claim 4)} \end{aligned}$$

Claim 7: $(A \cdot B)^\# \subseteq A \cdot (B \cdot A)^\# \cdot B$. By exchanging the roles of A and B in claim 6, we have $(B \cdot A)^\# \subseteq B \cdot (A \cdot B)^\# \cdot A$. Hence by claim 4, we obtain $A \cdot (B \cdot A)^\# \cdot B \subseteq A \cdot B \cdot (A \cdot B)^\# \cdot A \cdot B = (A \cdot B)^\#$.

Claim 8: $\{1\}^\# = (\{1\}^\#)^\#$. Thanks to claim 1, it is sufficient to prove $1 \in (\{1\}^\#)^\#$. But this is obvious since $1 = 1 \cdot 1^\# \cdot 1 \in (\{1\}^\#)^\#$.

At this point, the fact that $\downarrow(\mathbf{M})$ is a stabilisation monoid is a consequence of the fact that $(\downarrow(M), \subseteq, \cdot)$ is an ordered monoid, and claims 0 to 8. \square

We need now to construct the corresponding compatible mapping. For this, let us introduce some notations. For $X \subseteq M^\omega$, denote by $[X]_\downarrow \in (\downarrow(M))^\omega$ the sequence which to $n \in \omega$ associates:

$$[X]_\downarrow(n) = \{\mathbf{a}(n) : \mathbf{a} \in X\}^\#.$$

For a word $U = A_1 \dots A_n \in (\downarrow(M))^*$ and a word $u = a_1 \dots a_n \in M^*$, one abbreviates $a_1 \in A_1, \dots, a_n \in A_n$ by simply writing $u \in U$. Similarly, we write $A_1 \dots A_k \subseteq B_1 \dots B_l$ to denote that $k = l, A_1 \subseteq B_1, \dots, A_k \subseteq B_k$.

Let ρ be a mapping compatible with \mathbf{M} . We construct the mapping ρ' from $(\downarrow(M))^*$ to $\downarrow(M)$ defined for all $U \in (\downarrow(M))^*$ by:

$$\rho'(U) = [\{\rho(u) : u \in U\}]_{\downarrow} .$$

For making the proof to a little bit simpler, we use the following fact:

Fact 48 *Let $X, Y \subseteq M^\omega$, if for all $\mathbf{a} \in X$ there exists $\mathbf{b} \in Y$ such that $\mathbf{a} \preceq_\alpha \mathbf{b}$, then $[X]_{\downarrow} \preceq_\alpha [Y]_{\downarrow}$.*

Proof. Let $n, m \in \omega$ such that $\alpha(n) \leq m$. Let $a \in [X]_{\downarrow}(n)$. This means that $a = \mathbf{a}(n)$ for some $\mathbf{a} \in X$. Thus there exists $\mathbf{b} \in Y$ such that $\mathbf{a} \preceq_\alpha \mathbf{b}$. We thus have $\mathbf{a}(n) \leq \mathbf{b}(m)$, and then $a \leq \mathbf{b}(m) \in [Y]_{\downarrow}(m)$. We obtain, $[X]_{\downarrow} \preceq_\alpha [Y]_{\downarrow}$. \square

We are now ready for proving:

Lemma 49. *The mapping ρ' is compatible with $\downarrow(\mathbf{M})$.*

Proof. We denote by α the correction function witnessing the compatibility of ρ . As above, we do not precisely state the computation of correction functions.

Monotonicity Let $U \subseteq V$ be words in $(\downarrow(M))^*$. We can straightforwardly apply Fact 48 to $\{\rho(u) : u \in U\}$ and $\{\rho(v) : v \in V\}$, and get $\rho'(U) \preceq_{id} \rho'(V)$.

Letter, and product Let $A \in \downarrow(M)$. The sequence equal to A can also be written as $[a : a \in A]_{\downarrow}$. Hence, one can directly apply Fact 48 to it, and get $\rho'(A) \sim_\alpha A$. The equation $\rho'(\varepsilon) \sim_\alpha \{1\}_{\downarrow}$ is obtained in the same way.

Product As for item ‘letter’.

Substitution Let $U_1, \dots, U_k \in \downarrow(M)^*$. We will begin by giving an equivalent description to $\tilde{\rho}'(\rho'(U_1) \dots \rho'(U_k))$. For all $n \in \omega$ we have:

$$\begin{aligned} \tilde{\rho}'(\rho'(U_1) \dots \rho'(U_k))(n) &= \{\rho(a_1 \dots a_k)(n) : \forall i, a_i \in \rho'(U_i)(n)\}_{\downarrow} \\ &= \{\rho(a_1 \dots a_k)(n) : \forall i, a_i \leq \rho(u_i)(n), u_i \in U_i\}_{\downarrow} \\ &= \{\tilde{\rho}(\mathbf{a}_1 \dots \mathbf{a}_k)(n) : \forall i, \mathbf{a}_i \leq \rho(u_i), u_i \in U_i\}_{\downarrow} . \end{aligned}$$

Hence $\tilde{\rho}'(\rho'(U_1) \dots \rho'(U_k)) = [\{\tilde{\rho}(\mathbf{a}_1 \dots \mathbf{a}_k) : \forall i, \mathbf{a}_i \leq \rho(u_i), u_i \in U_i\}]_{\downarrow}$.

Let now $\mathbf{a}_1 \leq \rho(u_1), \dots, \mathbf{a}_k \leq \rho(u_k)$ for some $u_1 \in U_1, \dots, u_k \in U_k$. We have:

$$\begin{aligned} \tilde{\rho}(\mathbf{a}_1 \dots \mathbf{a}_k) &\preceq \tilde{\rho}(\rho(u_1) \dots \rho(u_k)) && \text{(def and monotonicity)} \\ &\sim \rho(u_1 \dots u_k) && \text{(substitution)} \end{aligned}$$

Hence by Fact 48 we get $\tilde{\rho}'(\rho(U_1) \dots \rho(U_k)) \preceq \rho(U_1 \dots U_k)$.

Conversely, let $u \in U_1 \dots U_k$, it can be decomposed into $u = u_1 \dots u_n$ with $u_1 \in U_1, \dots, u_k \in U_k$. We have:

$$\begin{aligned} \rho(u) &= \rho(u_1 \dots u_k) \\ &\sim \tilde{\rho}(\rho(u_1) \dots \rho(u_k)) && \text{(substitution)} \\ &= \tilde{\rho}(\mathbf{a}_1 \dots \mathbf{a}_k) \quad \text{with } \mathbf{a}_i = \rho(u_i) \text{ for } i = 1 \dots k \end{aligned}$$

From which, by Fact 48, we get $\rho(U_1 \dots U_k) \preceq \tilde{\rho}'(\rho(U_1) \dots \rho(U_k))$.

Stabilisation Let $E \in \downarrow(M)$ be an idempotent. Remark first that:

$$E^\sharp|_k E = [\{a|_k b : a \in E^\sharp, b \in E\}]_\downarrow .$$

Let $u \in E^k$. Since E is an idempotent, it is a sub-stabilisation monoid of \mathbf{M} , and use Lemma 44. We get that there exists a typed strict \sharp -expression ϕ over E such that $\rho(u) \preceq \text{type}(\phi)|_k \pi(\text{unfold}_k(\phi))$. Since $\text{type}(\phi) \in E^\sharp$ and $\pi(\text{unfold}_k(\phi)) \in E$, we can use the preliminary remark and Fact 48 to obtain $\rho'(E^k) \preceq E^\sharp|_k E$.

Conversely, remark that every element $c \in E$ is such that $c \leq a \cdot e \cdot b$ for some $a, e, b \in E$ with e idempotent. Thus:

$$E^\sharp|_k E = [\{(a \cdot e^\sharp \cdot b)|_k(a \cdot e \cdot b) : a, b, e \in E, e \cdot e = e\}]_\downarrow .$$

Hence, fix $a, e, b \in E$ with e idempotent. Consider the word $u = (a \cdot e)^{k-2}(e \cdot b)$ in E^k (if $k = 1$, then use the one-letter word $a \cdot e \cdot a$). We have:

$$\begin{aligned} \rho(u) &= \rho((a \cdot e)^{k-2}(e \cdot b)) \\ &\sim \tilde{\rho}(\rho(ae)\rho(e)^{k-2}\rho(ea)) && \text{(letter, product and monotonicity)} \\ &\sim \rho(ae^k b) && \text{(substitution)} \\ &\sim a \cdot \rho(e^k) \cdot b && \text{(product, substitution and letter)} \\ &\sim a \cdot (e^\sharp|_k e) \cdot b && \text{(stabilisation)} \\ &= (a \cdot e^\sharp \cdot b)|_k(a \cdot e \cdot b) . \end{aligned}$$

Hence by Fact 48, $E^\sharp|_k E \preceq \rho'(E^k)$. □

Proposition 14. *Recognisable cost-functions are closed under inf-projection.*

Proof. Let \mathbb{A} be an alphabet, $\mathbf{M} = \langle M, \cdot, \leq, \sharp \rangle$ be a stabilisation monoid, h be a length-preserving morphism from \mathbb{A}^* to M^* , and I be an ideal of \mathbf{M} . Let ρ be a mapping compatible with \mathbf{M} . By Proposition 10, one assumes without loss of generality that $\rho(u)$ is non-decreasing for all $u \in M^*$. Let f be defined by $f(u) = \sup\{n \in \omega : \rho(h(u)) \in I\}$, i.e., a representative of the cost function recognised by \mathbf{M}, h, I . Remark that, according to the choice of $\rho(h(u))$ non-decreasing, we have $n \leq f(u)$ iff $\rho(h(u)) \in I$ (prop \star). Let \mathbb{B} be an alphabet and p be a length preserving morphism from \mathbb{A} to \mathbb{B} . One aims at showing that $f_{\text{inf}, p}$ is recognisable.

For this, let us consider the monoid $\downarrow(\mathbf{M})$, the mapping h' from \mathbb{B} to $\downarrow(M)$ defined for all $b \in \mathbb{B}$ by:

$$h'(b) = \{h(a) : p(a) = b\}_\downarrow = h(p^{-1}(b))_\downarrow ,$$

and the ideal $I' \subseteq \downarrow(M)$ defined by:

$$I' = \{A \in \downarrow(M) : A \subseteq I\}.$$

Let $f'(v)$ be $\sup\{n \in \omega : \rho'(h'(v))(n) \in I'\}$, i.e., f' is a representative of the cost function recognised by $\downarrow(\mathbf{M}), h', I'$. Let $v = b_1 \dots b_k \in \mathbb{B}^*$, we have:

$$\begin{aligned}
f'(v) &= \sup\{n \in \omega : \rho'(h'(v))(n) \in I'\} && \text{(def of } f') \\
&= \sup\{n \in \omega : \rho'(h'(v))(n) \subseteq I\} && \text{(def of } I') \\
&= \sup\{n \in \omega : \rho'(h(p^{-1}(b_1)) \downarrow \dots h(p^{-1}(b_k)) \downarrow)(n) \subseteq I\} && \text{(def of } h') \\
&= \sup\{n \in \omega : \forall u. p(u) = v \rightarrow \rho(h(u))(n) \in I\} && \text{(def of } \rho') \\
&= \sup\{n \in \omega : \forall u. p(u) = v \rightarrow n \leq f(u)\} && \text{(prop } \star) \\
&= \inf\{f(u) : p(u) = v\} \\
&= f_{\text{inf}, p}
\end{aligned}$$

Hence, $\downarrow(\mathbf{M}), h', I'$ recognizes the inf-projection of f . □

The stabilisation monoid of coideals and sup-projection. Let $\mathbf{M} = \langle M, \cdot, \leq, \# \rangle$ be a stabilisation monoid. We perform a construction very similar to the one for inf-projection. This time we construct a stabilisation monoid $\uparrow(\mathbf{M})$, each element of which is a coideal of M . Since the model of stabilisation monoids is not invariant under reversing the order (because of the rule $e^\# \leq e$), it is not possible to reuse the above proof. The validity of the construction is established by Lemma 50. We then construct the corresponding compatible mapping, the correction of which is the subject of Lemma 52. We finally conclude with Proposition 15 which shows that this construction can be used for computing the sup-projection of a recognisable cost function.

Recall that a coideal of M is a subset $I \subseteq M$ such that for all $x \geq y$, if $x \in I$ then $y \in I$. Given a set $A \subseteq M$, one denotes by $A\uparrow$ the least coideal which contains it, i.e., the set $\{y : y \geq x \in A\}$. We denote by $\uparrow(M)$ the set of coideals of M . Given coideals $A, B \in \uparrow(M)$, their product is defined in the usual way by:

$$A \cdot B = \{a \cdot b : a \in A, b \in B\}\uparrow,$$

Given an idempotent coideal E , one defines $E^\#$ to be:

$$E^\# = \langle E \rangle^\#\uparrow = \{\text{type}(\phi) : \phi \in \text{exp}(E), \phi \text{ is typed}\}\uparrow.$$

Let us prove now the correction of this construction.

Lemma 50. $\uparrow(\mathbf{M}) = \langle \uparrow(M), \supseteq, \cdot, \# \rangle$ is a stabilisation monoid.

Proof. The fact that $\langle \uparrow(M), \supseteq, \cdot \rangle$ is an ordered monoid is standard and with no difficulty, the neutral element being $\{1\}\uparrow$. The fact that for all idempotent coideals $E \supseteq F$, $E^\# \supseteq F^\#$ is also obvious from the definition (call this *Claim 0*).

Let E be an idempotent coideal. We clearly have $E^\# \supseteq E$. It is also clear from the definition that $E^\# \cdot E^\# \subseteq E^\# \cdot E \subseteq E^\#$. For the converse, let $a \in E^\#$. This means $a \geq \text{type}(\phi)$ for some typed $\phi \in \text{exp}(E)$. If ϕ is of the form $\psi\psi'$, then $a \leq \text{type}(\psi) \cdot \text{type}(\psi') \in E^\# \cdot E^\#$. If ϕ is of the form $\psi^\#$, then it could be rewritten as $\psi^\# \cdot \psi$, and the above case be applied. Finally, if ϕ is simply a' ,

we use the fact that E is an idempotent, and get that there are $b, c \in E$ such that $a \leq a' \leq b \cdot c \in E^\sharp \cdot E^\sharp$. Hence $E^\sharp \cdot E^\sharp = E^\sharp \cdot E = E^\sharp$.

It remains the consistency. Let A, B be coideals such that $A \cdot B$ and $B \cdot A$ are idempotent. Let $\phi \in \exp(A \cdot B)$. We prove by induction on ϕ that there exists $a \in A$, $b \in B$, and $\psi \in \exp(B \cdot A)$ such that $\text{type}(\phi) \geq \text{type}(a\psi b)$. If $\phi = c \in A \cdot B$, then by idempotency of $A \cdot B$, there exists $a, a' \in A$ and $b, b' \in B$ such that $c \leq a \cdot b' \cdot a' \cdot b$. The induction hypothesis is satisfied for $\psi = b' \cdot a'$. If $\phi = \phi' \phi''$, one applies the induction hypothesis, and get $a', a'' \in A$, $b', b'' \in B$, and $\psi', \psi'' \in \exp(B \cdot A)$ such that $\text{type}(\phi') \leq a' \cdot \text{eval}(\psi') \cdot b'$, and $\text{type}(\phi'') \leq a'' \cdot \text{type}(\psi'') \cdot b''$. Let us set $a = a'$, $b = b''$, and $\psi = \psi' \cdot (b' \cdot a') \cdot \psi''$. This witnesses the validity of the induction hypothesis for ϕ . If finally $\phi = (\phi')^\sharp$, one applies the induction hypothesis, and get $a' \in A$, $b' \in B$, and $\psi' \in \exp(B \cdot A)$ such that $\text{eval}(\phi') \leq a' \cdot \text{eval}(\psi') \cdot b'$. Our problem is that $a \cdot \text{eval}(\psi')$ may not be an idempotent. However, let us set $K = |M|!$, it is classical that both $\text{eval}((a \cdot \psi' \cdot b)^K)$ and $\text{eval}((\psi' \cdot b \cdot a)^K)$ are idempotents. Furthermore, since $\text{eval}(\phi')$ is an idempotent, $\text{eval}(\phi') \leq \text{eval}((a \cdot \psi' \cdot b)^K)$. Using consistency, we get:

$$\begin{aligned} \text{eval}(\phi) &= \text{eval}((\phi')^\sharp) \\ &\leq \text{eval}((a \cdot \psi' \cdot b)^K) \\ &= a \cdot \underbrace{\text{eval}(((\psi' \cdot (b \cdot a))^K)^\sharp \cdot (\psi' \cdot (b \cdot a))^{K-1} \cdot \psi')}_{\in \exp(B \cdot A)} \cdot b . \end{aligned}$$

From this statement, we directly get that $(A \cdot B)^\sharp \subseteq A \cdot (B \cdot A)^\sharp \cdot B$. For the converse, we swap the roles of A and B , and get $A \cdot (B \cdot A)^\sharp \cdot B \subseteq A \cdot B \cdot (A \cdot B)^\sharp \cdot B \cdot A = (A \cdot B)^\sharp$. \square

We need now to construct the corresponding compatible mapping. For this, let us introduce some notations. For $X \subseteq M^\omega$, denote by $[X]_\uparrow \in (\uparrow(M))^\omega$ the sequence which to $n \in \omega$ associates:

$$[X]_\uparrow(n) = \{\mathbf{a}(n) : \mathbf{a} \in X\}_\uparrow .$$

For a word $U = A_1 \dots A_n \in (\uparrow(M))^*$ and a word $u = a_1 \dots a_n \in M^*$, one abbreviates $a_1 \in A_1, \dots, a_n \in A_n$ by simply writing $u \in U$. Similarly, we write $A_1 \dots A_k \subseteq B_1 \dots B_l$ to denote that $k = l$, $A_1 \subseteq B_1, \dots, A_k \subseteq B_k$.

Let ρ be a mapping compatible with \mathbf{M} . We construct the mapping ρ' from $(\uparrow(M))^*$ to $(\uparrow(M))^\omega$ defined for all $U \in \uparrow(M)^*$ by:

$$s\rho'(U) = [\{\rho(u) : u \in U\}]_\uparrow .$$

For making the proof to a little bit simpler, we use the following fact (recall that the ordering in $\uparrow(\mathbf{M})$ is ' \leq ' = ' \supseteq ', i.e., is reversed):

Fact 51 *Let $X, Y \subseteq M^\omega$, if for all $\mathbf{b} \in Y$ there exists $\mathbf{a} \in X$ such that $\mathbf{a} \preceq_\alpha \mathbf{b}$, then $[X]_\uparrow \preceq_\alpha [Y]_\uparrow$.*

Proof. Let $n, m \in \omega$ such that $\alpha(n) \leq m$. Let $b \in [Y]_{\uparrow}(m)$. This means that $b = \mathbf{b}(n)$ for some $\mathbf{b} \in Y$. Thus there exists $\mathbf{a} \in X$ such that $\mathbf{a} \preceq_{\alpha} \mathbf{b}$. We thus have $\mathbf{a}(n) \leq \mathbf{b}(m)$, and then $[X]_{\uparrow}(n) \ni \mathbf{a} \leq \mathbf{b}(m)$. Hence $[X]_{\uparrow}(n) \supseteq [Y]_{\uparrow}(m)$. To sum up, we have $[X]_{\uparrow} \preceq_{\alpha} [Y]_{\uparrow}$. \square

We are now ready for proving:

Lemma 52. *The mapping ρ' is compatible with $\uparrow(\mathbf{M})$.*

Proof. We denote by α the correction function witnessing the compatibility of ρ . As above, we do not precisely state the computation of correction function.

Monotonicity Let $U \supseteq V$ be words in $(\uparrow(M))^*$. We can straightforwardly apply Fact 48 to $\{\rho(u) : u \in U\}$ and $\{\rho(v) : v \in V\}$, and get $\rho'(U) \preceq_{id} \rho'(V)$.

Letter, and product Let $A \in \uparrow(M)$. The sequence equal to A can also be written as $[a : a \in A]_{\uparrow}$. Hence, one can directly apply Fact 48 twice, together with the letter rule in \mathbf{M} , and get $\rho'(A) \sim_{\alpha} A$. The equation $\rho'(\varepsilon) \sim_{\alpha} \{1\}_{\uparrow}$ is obtained in the same way.

Product As for the item ‘letter’.

Substitution Let U_1, \dots, U_k be words in $(\uparrow(M))^*$. Let us first give an equivalent formula for $\tilde{\rho}'(\rho'(U_1) \dots \rho'(U_k))$. For all $n \in \omega$ we have:

$$\begin{aligned} \rho'(\rho'(U_1) \dots \rho'(U_k))(n) &= \{\rho(a_1 \dots a_k)(n) : \forall i, a_i \in \rho'(U_i)(n)\}_{\uparrow} \\ &= \{\rho(a_1 \dots a_k)(n) : \forall i, a_i \geq \rho(u_i)(n), u_i \in U_i\}_{\uparrow} \\ &= \{\tilde{\rho}(\mathbf{a}_1 \dots \mathbf{a}_k)(n) : \forall i, \mathbf{a}_i \geq \rho(u_i), u_i \in U_i\}_{\uparrow}. \end{aligned}$$

Hence $\tilde{\rho}'(\rho'(U_1) \dots \rho'(U_k)) = [\{\tilde{\rho}(\mathbf{a}_1 \dots \mathbf{a}_k) : \forall i, \mathbf{a}_i \geq \rho(u_i), u_i \in U_i\}]_{\uparrow}$.

Let now $\mathbf{a}_1 \geq \rho(u_1), \dots, \mathbf{a}_k \geq \rho(u_k)$ for some $u_1 \in U_1, \dots, u_k \in U_k$. We have:

$$\begin{aligned} \tilde{\rho}(\mathbf{a}_1 \dots \mathbf{a}_k) &\succeq \tilde{\rho}(\rho(u_1) \dots \rho(u_k)) && \text{(def and monotonicity)} \\ &\sim \rho(u_1 \dots u_k) && \text{(substitution)} \end{aligned}$$

Hence by Fact 51 we get $\tilde{\rho}'(\rho(U_1) \dots \rho(U_k)) \succeq \rho(U_1 \dots U_k)$.

Conversely, let $u \in U_1 \dots U_k$, it can be decomposed into $u = u_1 \dots u_k$ with $u_1 \in U_1, \dots, u_k \in U_k$. We have:

$$\begin{aligned} \rho(u) &= \rho(u_1 \dots u_k) \\ &\sim \tilde{\rho}(\rho(u_1) \dots \rho(u_k)) && \text{(substitution)} \\ &= \tilde{\rho}(\mathbf{a}_1 \dots \mathbf{a}_k) \quad \text{with } \mathbf{a}_i = \rho(u_i) \text{ for } i = 1 \dots k \end{aligned}$$

From which, by Fact 51, we get $\rho(U_1 \dots U_k) \succeq \tilde{\rho}'(\rho(U_1) \dots \rho(U_k))$.

Stabilisation Let $E \in \uparrow(M)$ be an idempotent. If we consider $E^{\sharp} \cup \{1\}$, it induces a stabilisation sub-monoid of \mathbf{M} . Hence, there exists a compatible mapping ρ_{\sharp} over $E^{\sharp} \cup \{1\}$. \square *to complete.*

Proposition 15. *Recognisable cost-functions are closed under sup-projection.*

Proof. Let \mathbb{A} be an alphabet, $\mathbf{M} = \langle M, \cdot, \leq, \# \rangle$ be a stabilisation monoid, h be a length-preserving morphism from \mathbb{A}^* to M^* , and I be a coideal of \mathbf{M} . Let ρ be a mapping compatible with \mathbf{M} . By Proposition 10, one assumes without loss of generality that $\rho(u)$ is non-decreasing for all $u \in M^*$. Let f be defined by $f(u) = \sup\{n \in \omega : \rho(h(u)) \in I\}$, i.e., a representative of the cost function recognised by \mathbf{M}, h, I . Remark that, according to the choice of $\rho(h(u))$ non-decreasing, we have $n \leq f(u)$ iff $\rho(h(u)) \in I$ (prop \star). Let \mathbb{B} be an alphabet and p be a length preserving morphism from \mathbb{A} to \mathbb{B} . One aims at showing that $f_{\text{sup}, p}$ is recognisable.

For this, let us consider the monoid $\uparrow(\mathbf{M})$, the mapping h' from \mathbb{B} to $\uparrow(M)$ defined for all $b \in \mathbb{B}$ by:

$$h'(b) = \{h(a) : p(a) = b\}^\uparrow = h(p^{-1}(b))^\uparrow,$$

and the ideal $I' \subseteq \uparrow(M)$ defined by:

$$I' = \{A \in \uparrow(M) : A \cap I \neq \emptyset\}.$$

Let $f'(v)$ be $\sup\{n \in \omega : \rho'(h'(v))(n) \in I'\}$, i.e., f' is a representative of the cost function recognized by $\uparrow(\mathbf{M}), h', I'$. Let $v = b_1 \dots b_k \in \mathbb{B}^*$, we have:

$$\begin{aligned} f'(v) &= \sup\{n \in \omega : \rho'(h'(v))(n) \in I'\} && \text{(def of } f') \\ &= \sup\{n \in \omega : \rho'(h'(v))(n) \cap I \neq \emptyset\} && \text{(def of } I') \\ &= \sup\{n \in \omega : \rho'(h(p^{-1}(b_1))^\uparrow \dots h(p^{-1}(b_k))^\uparrow)(n) \cap I \neq \emptyset\} && \text{(def of } h') \\ &= \sup\{n \in \omega : \exists u. p(u) = v \wedge \rho(h(u))(n) \in I\} && \text{(def of } \rho') \\ &= \sup\{n \in \omega : \exists u. p(u) = v \wedge n \leq f(u)\} && \text{(prop } \star) \\ &= \sup\{f(u) : p(u) = v\} \\ &= f_{\text{sup}, p} \end{aligned}$$

Hence, $\uparrow(\mathbf{M}), h', I'$ recognizes the sup-projection of f . □

4.6 From automata to stabilisation monoids

The purpose of this section is to prove that we can transform a cost automaton (either B or S) accepting some cost function, into a stabilisation monoid recognising this cost function. We first treat the case of simple B -automata (i.e., over the actions ε, ic and r), and in a second step the case of general B - and S -automata.

Simple B -automata. In this section we prove that the functions accepted by simple B -automata are recognisable (Lemma 56). For this, we start by showing that over the set of actions $\text{Act}_{sB1} = \{\varepsilon, ic, r\}$, the function f_{sB1} which to $u \in \text{Act}_{sB1}^*$ associates:

$$f_{sB1}(u) = \sup C(u)$$

is recognisable (Lemma 55).

The elements of the monoid are $M_{sB1} = \{1, ic, r, 0\}$, in which informally, 1 is the neutral element, ic correspond to a small sequence of increments and resets, r corresponds to a sequence without big segments of increments without resets, and containing at least one reset, and 0 correspond to sequences that perform a long sequence of increments without resets. The definition of \mathbf{M}_{sB1} and the stabilisation monoid is defined by the following table:

	1	ic	r	0	#	
1	1	ic	r	0	1	
ic	ic	ic	r	0	0	$0 \leq ic \leq 1 \leq r$.
r	r	r	r	0	r	
0	0	0	0	0	0	

Remark 8. You can remark that we did not use the order defined by the identity augmented by $0 < ic$. The two choices of order are possible, and both choices yield a valid stabilisation monoid. The advantage of having a ‘larger’ order is that the stabilisation monoid has less ideals. For this reason, the ideals and coideals construction – as we perform below – give smaller stabilisation monoids. In the particular case of the order $0 \leq ic \leq 1 \leq r$, the stabilisation monoid of ideals (resp. coideals) coincide with the original stabilisation monoid (i.e., it has 4 elements, as opposed to 24 elements if we would have made the other choice of order). Despite this, we do not try to optimise the complexity in this work.

Lemma 53. \mathbf{M}_{sB1} is a stabilisation monoid.

Proof. By case analysis. □

The corresponding compatible mapping is defined for $u \in \{1, ic, r, 0\}^*$ by:

$$\rho_{sB1}(u)(n) = \begin{cases} 1 & \text{if } u \in 1^* \\ ic & \text{if } u \in (1^*ic)^{<n}1^* \\ r & \text{if } u \in ((1^*ic)^{<n}1^*r)^+(1^*ic)^{<n}1^* \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 54. The mapping ρ_{sB1} is compatible with \mathbf{M}_{sB1} .

Proof. By case analysis. Another argument is simply that this is the result of the construction in the proof of Theorem 2 (which is particularly simple here since every \mathcal{J} -class is reduced to a single idempotent, i.e., regular and trivial). □

Let now h_{sB1} be the (length preserving) morphism from \mathbf{Act}_{sB1}^* to M_{sB1}^* which maps ϵ to 1, ic to ic, and r to r. Let also I_{sB1} be $\{0\}$.

Lemma 55. The function f_{sB1} is recognised by $\mathbf{M}_{sB1}, h_{sB1}, I_{sB1}$.

Proof. From the definition of ρ_{sB1} , we have $f_{sB1}(u) \geq n$ iff $\rho_{sB1}(h_{sB1}(u)) = 0$. □

Consider now a simple B -automaton $\mathcal{A} = \langle Q, \mathbb{A}, In, Fin, \Gamma, \Delta \rangle$. Let $R \subseteq \Delta^*$ be the language of runs, i.e., consisting of words in Δ^* such that the target state of a transition coincides with the source state of the next transition, such that the source state of the first transition is initial, and such that the target of the last transition is final. This set R is regular, and hence the mapping χ_R is recognisable (Remark 7). Let p be the morphism from Δ^* to \mathbb{A}^* which to a transition (p, a, u, q) associates the letter a . For all counter $\iota \in \Gamma$, let p_ι be the morphism from Δ^* to \mathbf{Act}^* which to (p, a, u, q) associates u_ι (i.e., the actions over counter ι done by the transition).

Let $u \in \mathbb{A}^*$ be a word. The value of the automaton over this word is the infimum over all runs of the maximum of the values computed by each counter. We can summarise this in the equation:

$$\llbracket \mathcal{A} \rrbracket_B(u) = \inf \left\{ \max(\chi_R(\sigma), \max_{\iota \in \Gamma} f_{sB\iota}(p_\iota(\sigma))) : \sigma \in \Delta^*, p(\sigma) = u \right\}. \quad (1)$$

We just used the elementary trick that sequences of transitions which do not form a valid run are given a value of ω , i.e., do not impact in the evaluation of the infimum.

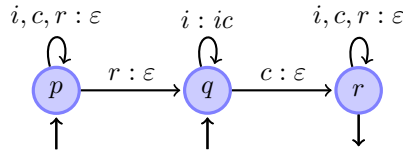
This formula is nothing but an inf-projection of the maximum of $|\Gamma| + 1$ recognisable cost functions (thanks to Remark 7 and Lemmas 55 and 41). Since recognisable cost functions are closed under maximum and inf-projection, we directly get:

Lemma 56. *Cost functions accepted by simple B -automata are recognisable.*

Generic proof for cost automata. The proof of Lemma 56 can be used for all forms of automata, providing that we have stabilisation monoids for treating each counters. Indeed, the equation 1 can be used for B -automaton in general form, as well as for S -automaton just by replacing max by min, and inf by sup.

Hence, using Lemma 56, the cost function f_B is recognisable.

For an S -counter, one uses the following simple B -automaton (one could also use Proposition 5):



This simple B -automaton accepts the function f_S . It works by guessing the ‘check’ in the input that contributes for the least value in $C(u)$: it waits in state p , until reaching a reset and entering state q . In state q it counts the number of increments before the first check, and at this point enters state r , waiting for the end of the word. The best run for this automaton (the one yielding the smallest value) is the one such that the checked value (when going from state q to state r) is as small as possible: i.e., the one that decides the value of $f_S(u)$.

Hence, using Lemma 56, the cost function f_S is recognisable.

From the above facts, we obtain:

Lemma 57. *The cost functions computed by B and S -automata are effectively recognisable.*

4.7 From stabilisation monoids to automata

In this section, we show the converse to the results of the previous section. We show that for every recognisable cost function, one can effectively construct a (history-deterministic, hierarchical and simple) B -automaton and an (history-deterministic, hierarchical and simple) S -automaton that accepts it. The two constructions follow the same principles, but differ slightly. We present them successively.

From stabilisation monoids to history-deterministic B -automata We use the following notation for ideals: for $a \in M$, I_a is the (already defined) set $\{b \in M : b \leq a\}$. We will also use the dual notation $I_{\bar{a}}$ which represents the ideal $\{b \in M : a \not\leq b\}$. (this notation should not be confused with \bar{I} which denotes the co-ideal complement of an ideal I).

The goal of this section is to establish the following lemma.

Lemma 58. *Every recognisable cost function is accepted by an history-deterministic simple hierarchical B -automaton.*

From now, we fix ourselves an alphabet \mathbb{A} , a stabilisation monoid $\mathbf{M} = \langle M, \cdot, \leq, \sharp \rangle$, a length-preserving morphism h from \mathbb{A}^* to M^* , and an ideal $I \subseteq M$. We also fix a mapping ρ compatible with \mathbf{M} . We aim at constructing a B -automaton for the cost function recognised by \mathbf{M}, h, I .

We use for this a enriched form of simple B -automata that relates with more precision with the semantic with the stabilisation monoid: a simple B -automaton is *with output in M* if it is equipped (in replacement of the set of accepting states) with a mapping F from its set of states to M . Intuitively, when reaching the state p after some n -run over some word u , one ‘expects’ the word u to satisfy $\rho(h(u))(n) = F(p)$. In this framework, a run is no more required to end in a final state (since there are no final states): a *run* is simply a partial run, the first state of which is initial.

We prove below the existence of a simple B -automaton over the alphabet M with output in M , and of a correction function α such that:

Correctness For all word $u \in M^*$, all $n \in \omega$, and all n -run over u ending in state q ,

$$(\perp|_n F(q)) \preceq_\alpha \rho(u) ,$$

or equivalently, $\forall m. m \geq \alpha(n) \rightarrow \rho(u)(m) \geq F(q) .$

Completeness For all $n \in \omega$, there exists a translation strategy δ_n such that for all word u , there exists an n -run σ driven by δ_n over u , and

$$\rho(u) \preceq_\alpha (F(q)|_n \top) ,$$

or equivalently, $\forall m. \alpha(m) \leq n \rightarrow \rho(u)(m) \leq F(q) ,$

in which q is the last state assumed by σ .

In this statement, we give two (by definition equivalent) versions of the completeness and the correctness. The left version is the one we establish below: it is much more natural to use in the context of compatible mappings. We use in it the fake elements \perp and \top that represent new elements, below and above respectively every other elements in M (in practice, those elements never interfere with M , i.e., \perp and \top will never appear in a product or a stabilisation). The right version is the good one for concluding as we explain just below.

For the sake of explanation, let us assume for some moment that α is the identity correction function. In this case, the correctness would simply say that for all $n \in \omega$, every n -run which ends in some state q is such that $F(q) \leq \rho(u)(n)$, and the completeness that there exists a specific n -run (the one driven by δ) that ends in a state q such that $\rho(u)(n) \leq F(q)$. This implies:

$$\rho(u)(n) \notin I \iff \text{for all } n\text{-run } \sigma \text{ over } u \text{ ending in } q, F(q) \notin I .$$

From which we deduce that the B -automaton in which the final states are set to be $F^{-1}(\bar{I})$, and that has as transition each (p, a, v, q) such that $(p, h(a), v, q)$ is a transition of \mathcal{A} , accepts the function $u \mapsto \inf\{n : \rho(h(u))(n) \notin I\}$, i.e., a function recognised by \mathbf{M}, h, I .

We use the exact same argument for concluding, assuming the correctness and completeness, the proof of Lemma 58:

Proof. (of Lemma 58) One constructs an automaton $\mathcal{A}_{\mathbf{M},h,I}$ by taking the automaton \mathcal{A} , removing the output mapping F , setting as final each state q such that $F(q) \notin I$, and putting a transition (p, a, v, q) for $a \in \mathbb{A}$ whenever $(p, h(a), v, q)$ is a transition of \mathcal{A} . Let f be the cost function which to $u \in \mathbb{A}^*$ associates $f(u) = \inf\{n : \rho(h(u))(n) \notin I\}$. Let $u \in \mathbb{A}^*$ be a word.

Let us consider an n -run σ of $\mathcal{A}_{\mathbf{M},h,I}$ over u . This run ends in a final state q , i.e., a state such that $F(q) \notin I$. Using correctness, $\rho(h(u))(\alpha(n)) \geq F(q) \notin I$, and hence $\rho(h(u))(\alpha(n)) \notin I$. Hence, using Fact 3, $f(u) \preceq_\alpha \llbracket \mathcal{A}_{\mathbf{M},h,I} \rrbracket_B(u)$.

Conversely, let n be such that $\rho(u)(n) \notin I$. Using the completeness property, there exists a partial $\alpha(n)$ -run driven by $\delta_{\alpha(n)}$ over u which ends in a state q such that $F(q) \geq \rho(u)(n)$. Since I is an ideal and $\rho(u)(n) \notin I$, we deduce that $F(q) \notin I$, i.e., the state q is final. Hence, the partial run driven by $\delta_{\alpha(n)}$ is a run. We get $\llbracket \mathcal{A}_{\mathbf{M},h,I} \rrbracket_B^\delta(u) \preceq_\alpha f(u)$.

This completes the proof of Lemma 58 according to Fact 6. □

Let us turn ourselves to the definition of the automaton \mathcal{A} itself, and to the proof of correctness (Lemma 59) and of completeness (Lemma 60). Let \mathbf{M} be the stabilisation monoid $\langle M, \cdot, \leq, \# \rangle$.

The construction is by induction on a subset $S \subseteq M$ which is \mathcal{J} -closed, i.e., such that $M \cdot S \cdot M \subseteq S$. The induction hypothesis is that there is an automaton over the alphabet S that has the properties of completeness and correctness.

If $S = \emptyset$, then the alphabet is empty, and the empty automaton directly satisfies the statements.

Else, let $J \subseteq S$ be some \mathcal{J} -class different which is maximal for $\leq_{\mathcal{J}}$. Let S' be $S \setminus J$ (by maximality of J , S is also \mathcal{J} -closed). We assume by induction hypothesis the existence of a simple B -automaton \mathcal{A}' with output in S' which satisfies the induction hypothesis for letters in S' .

The construction of the new simple hierarchical B -automaton \mathcal{A} depends upon the nature of the class J . The interesting case is when J is an unstable \mathcal{J} -class (we briefly mention afterward the other cases). Hence we assume now that J is an unstable \mathcal{J} -class. One constructs the new automaton $\mathcal{A} = \langle Q, M, I, \Gamma, \Delta \rangle$ with $Q = Q' \times J \cup \{1\}$, $\Gamma = \Gamma' + \{\iota\}$ in which ι is a fresh counter, $I = I' \times \{1\}$ and Δ is defined as:

$$\begin{aligned} \Delta &= \{((p, a), b, (ic)_{\iota}, (p, a \cdot b)) : a \cdot h(b) \in J\} & (a) \\ &\cup \{((p, a), b, \gamma r_{\iota}, (q, 1)) : a \cdot h(b) \in J, (p, a^{\#} \cdot h(b), \gamma, q) \in \Delta'\} & (b) \\ &\cup \{((p, a), b, \gamma r_{\iota}, (q, 1)) : a \cdot h(b) \notin J, (p, a \cdot h(b), \gamma, q) \in \Delta'\} & (c) \end{aligned}$$

where γr_{ι} represents the extension of γ to Γ that to ι associates r . Finally, for all state (p, a) , one sets $F((p, a)) = F'(p) \cdot a$. One can remark that this construction naturally produces a hierarchical automaton (up to renaming of counters). Indeed, each time some action is performed on a counter of \mathcal{A}' , the new counter ι is reset. Hence if \mathcal{A}' is hierarchical, so is \mathcal{A} .

The intuition behind this construction is that the automaton maintains (1) the state of the automaton \mathcal{A}' and (2) the value of $\pi(v)$ where v is a suffix of the word read so far such that v is J -smooth. Each times the suffix increases in size, the counter is incremented and checked (a). After some time, the counter reaches a too high value; in this case the automaton cannot use the ‘too expensive’ transition (a), and ‘is forced’ into resetting using transition (b): the result is that the value $\pi(v)$ is degraded into $\pi(v)^{\#}$: the suffix v is now ‘considered as long’. Of course, the automaton could take the transition (b) earlier, but this would not help him. The transition (c) is used when the suffix v is prolonged in such a way that $\pi(v)$ does not belong to J anymore. When using transition (b) or (c), the value produced does not belong to J anymore, and the \mathcal{A}' part maintains this information.

In the case of a stable \mathcal{J} -class or an irregular one, the situation is simpler; there is no need for a new counter, and we use the following simplified set of transitions:

$$\begin{aligned} \Delta &= \{((p, a), b, \epsilon, (p, a \cdot h(b))) : a \cdot b \in J\} & (a') \\ &\cup \{((p, a), b, \gamma, (q, 1)) : a \cdot h(b) \notin J, (p, a \cdot h(b), \gamma, q) \in \Delta'\} & (c') \end{aligned}$$

where ϵ is the mapping constant equal to ϵ over Γ' . We will not establish the correctness nor the completeness in this simpler case.

Lemma 59. *There exists α such that for all n -run of \mathcal{A} ending in state q over a word u , $(\perp|_n F(q)) \preceq_\alpha \rho(u)$.*

Proof. Let σ be the n -run over the word u ending in a state q . Our first step consists in decomposing σ . Each transition in σ is of one of the forms (a), (b) or (c). Let us factorise σ into $\sigma_1 \delta_1 \dots \delta_k \sigma_{k+1}$ in such a way that all transitions each σ_i is a (possibly empty) subrun consisting of transitions of the form (a), while each δ_i is a transition of the form (b) or (c). Let $u_1 a_1 \dots a_k u_{k+1}$ be the corresponding factorisation of u . By inspecting the transitions and the set of initial states, one remarks that the states assumed at the beginning of each subrun σ_i is of the form $(p_i, 1)$ for some $p_i \in Q'$. One remarks also that the state assumed at the end of σ_i has to be of the form $(p_i, \pi(u_i))$ (since transitions of the form (a) enforces that the first component of the state remains unchanged, and the second component evolves by a right product with the current letter). In particular $q = (p_{k+1}, \pi(u_{k+1}))$.

Let b_i be $\pi(u_i)^\sharp$ if $i \leq k$ and δ_i is of the form (b), and be $\pi(u_i)$ in all other cases. Let \mathbf{b}_i be $b_i |_{|u_i|} \pi(u_i)$. We can remark that transitions of the form (a) always increment the counter ι . As a consequence, since σ is an n -run, no u_i has a length greater than n . We deduce from that:

$$\mathbf{b}_i = (\pi(u_i)^\sharp |_{|u_i|} \pi(u_i)) \leq (\pi(u_i)^\sharp |_{|u_i|} \pi(u_i)) = \beta(u_i) \sim \rho(u_i) .$$

To each transition of the form (b) or (c), corresponds in its a definition a transition from Δ' . Let $\delta'_1, \dots, \delta'_k$ be the transition corresponding to $\delta_1, \dots, \delta_k$ respectively. According to the remark above, each δ'_i is of the form $(p_{i-1}, b_i \cdot a_i, \gamma_i, p_i)$. Since σ is an n -run and each action on the counter of Γ' are kept in $\sigma' = \delta'_1 \dots \delta'_n$, it follows that σ' is a n -run of \mathcal{A}' over the word $u' = (b_1 \cdot a_1) \dots (b_k \cdot a_k)$. Hence, by induction hypothesis, we know that $\perp|_n F'(p_{k+1}) \preceq_{\alpha'} \rho((b_1 \cdot a_1) \dots (b_k \cdot a_k))$.

Putting all this together we can derive the following chain of inequalities:

$$\begin{aligned} \perp|_n F(q) &= \perp|_n F'(p_{n+1}) \cdot b_{n+1} \\ &\preceq \perp|_n (\rho((b_1 \cdot a_1) \dots (b_n \cdot a_n)) \cdot b_{n+1}) \\ &\leq \rho((\pi(u_i)^\sharp \cdot a_1) \dots (\pi(u_n)^\sharp \cdot a_n)) \cdot \pi(u_{n+1}^\sharp) |_{|u_{n+1}|} \rho((b_1 \cdot a_1) \dots (b_n \cdot a_n)) \cdot b_{n+1} \\ &= \tilde{\rho}((s_1 \cdot a_1) \dots (s_n \cdot a_n)) \cdot s_{n+1} \\ &\sim \rho(s_1 a_1 \dots s_n a_n s_{n+1}) \\ &\preceq \tilde{\rho}(\beta(u_1) a_1 \dots \beta(u_n) a_n \beta(u_{n+1})) \\ &\sim \rho(u) \end{aligned}$$

□

Let us fix n . We need now to define the translation strategy δ_n for n that witnesses the history determinism of the construction. The oracle has to resolve non-determinism conflicts. In this construction, two forms of non-determinism can occur. The first one is the one inherited by the induction hypothesis, and we use for resolving it the translation strategy δ' inherited from the induction

hypothesis. The second form of non-determinism occurs when in a configuration (p, a) and while reading a letter b such that $a \cdot h(b) \in D$; in such a situation, one has to choose between transition (a) and transition (b) . For this, the translation strategy maintains in its memory (there is no memory constraints here) the value of the counter ι , and when in front of an $(a)/(b)$ choice, it always choose transition (a) unless the value of the counter ι is n and in this case, it chooses transition (b) . We will not provide any more formal description of this translation strategy.

Lemma 60. *For all n and all word u , there exists a run driven by δ_n over u exist, it is an n -run, and*

$$\rho(u) \preceq_\alpha (F(q)|_n \top) ,$$

in which q is the last state assumed by this run.

Proof. It is clear that the run exists (there is always a transition available, and the translation strategy always chooses a valid transition). It is clear also that the resulting run is an n -run since as soon the value of the counter ι reaches value n , it is reset during the following transition.

Let σ be the δ_n -driven n -run over u . Let $\sigma = \sigma_1 \delta_1 \dots \sigma_n \delta_k \sigma_{k+1}$ as well as $u = u_1 a_1 \dots a_k u_{k+1}$, p_1, \dots, p_{k+1} , and $\sigma' = \delta'_1 \dots \delta'_k$, b_1, \dots, b_k be as in the proof of Lemma 59. By construction of δ_n , σ' is the run of \mathcal{A}' driven by δ'_n on the word $u' = (b_1 \cdot a_1) \dots (b_k \cdot a_k)$. Hence by induction hypothesis $\rho(u') \preceq_{\alpha'}$ $(F'(p_{k+1})|_n \top)$.

Set \mathbf{b}_i to be $(b_i|_n \pi(u_i))$ (here the definition differs from the one in the proof of Lemma 59). The important point is that in an \mathcal{O} -driven n -run, the transition (b) is never used unless counter ι reaches the value n . As a consequence, if $b_i = \pi(u_i)^\sharp$, then $|u_i| = n$. This implies for a suitable α :

$$\rho(u_i) \sim_\alpha \beta(u_i) = (\pi(u_i)^\sharp|_{|u_i|} \pi(u_i)) \preceq_\alpha (b_i|_n \top) = \mathbf{b}_i .$$

Recall that the last state assumed by σ is $q = (p_{k+1}, \pi(u_{k+1}))$, and as a consequence that $F(q) = F'(p_{k+1}) \cdot b_{k+1}$ by construction of F and definition of b_{k+1} . Let us compute:

$$\begin{aligned} \rho(u) &\sim \tilde{\rho}(\rho(u_1) a_1 \dots a_k \rho(u_{k+1})) \\ &\preceq \tilde{\rho}(s_1 a_1 \dots a_k s_{k+1}) \\ &\sim \tilde{\rho}((s_1 \cdot a_1) \dots (s_k \cdot a_k)) \cdot s_{k+1} \\ &= \rho((b_1 \cdot a_1) \dots (b_k \cdot a_k)) \cdot b_{k+1}|_n \rho(\pi(u_1 a_1) \dots \pi(u_k a_k)) \pi(u_{k+1}) \\ &\preceq F'(p_{k+1}) \cdot b_{k+1}|_n \top \\ &= (F(q)|_n \top) . \end{aligned}$$

□

From stabilisation monoids to history-deterministic S-automata The goal of this section is to establish the following lemma.

Lemma 61. *Every recognisable cost function is accepted by an history-deterministic hierarchical S-automaton.*

The principle of the construction is identical. Everything is to some extent reversed. This time the S-automaton with output in M we construct has the following properties (the equations are reversed):

Correctness For all word u and all n -run over it ending in state q ,

$$\rho(u) \preceq_\alpha (F(q)|_n \top) , \quad \text{or equivalently, } n \preceq_\alpha \overline{I_{F(q)}}[\rho(u)] .$$

Completeness For all n , there exists a translation strategy δ such that for all word u , there exists a n -run driven by δ over u , and

$$(\perp|_n F(q)) \preceq_\alpha \rho(u) , \quad \text{or equivalently } I_{\overline{F(q)}}[\rho(u)] \preceq_\alpha n .$$

in which q is the last state assumed by the n -run.

Another change is that when constructing the automaton \mathcal{A}_f , the set of accepting states is now $F^{-1}(I)$ instead of $F^{-1}(\bar{I})$

The automaton is also constructed by an induction on the size of the stabilisation monoid. Everything is identical, but the transitions which are slightly different (in the unstable case):

$$\begin{aligned} \Delta &= \{((p, a), b, i_\iota, (p, a \cdot b)) : a \cdot h(b) \in D\} & (a) \\ &\cup \{((p, a), b, \gamma r_\iota, (q, 1)) : a \cdot h(b) \in D, (p, a^\# \cdot h(b), \gamma, q) \in \Delta'\} & (b) \\ &\cup \{((p, a), b, \gamma r_\iota, (q, 1)) : a \cdot h(b) \notin D, (p, a \cdot h(b), \gamma, q) \in \Delta'\} & (c) \end{aligned}$$

where i_ι represents the mapping which to every counter in Γ' associates ϵ and to ι associates i , while γr_ι represents the extension of γ to Γ that to ι associates cr (atomic check followed by a reset), and γr_ι represents the extension of γ to Γ that to ι associates r .

The intuition behind this construction is that the automaton maintains (1) the state of the automaton \mathcal{A}' and (2) the value of $\pi(h(v))$ where v is a suffix of the word read so far such that $h(v)$ is D -smooth. Each times the suffix increases in size, the counter is incremented (a). If the length of v is sufficiently long, the counter ι has a high value, and the automaton takes transition (b) and improves the value of $\pi(h(v))$ into $\pi(h(v))^\#$ (this was a degradation in the B -case, this is good here since everything is 'reversed'). If the automaton reads a new letter b such that $\pi(vb) \notin D$, then it uses the transition (c). Of course the automaton could always choose transition (a) and never transition (b), but this would not be a good choice. When using transition (b) or (c), the value produced does not belong to D anymore, and the \mathcal{A}' part maintains this information.

Different slight variations around this construction are possible. Let us stress the fact that it is important to be able to reset without checking the value of

counters for it to work in a history deterministic way. Indeed, it is possible to construct an automaton that uses only the actions on counters $\{\epsilon, i, cr\}$: it has to guess in advance if the following smooth word is short or long, and increment the counter only in the latter case, and perform a atomic cr in the end. This construction is correct, but does not have the history-determinism property: ‘guessing in advance’ is a contradiction to that.

5 Logic

In this section we show how it is possible to use the theory of regular cost functions in deciding extensions of monadic second order logic with cardinality inequalities. Those results strictly extends the classical equivalence established by Büchi between automata and monadic second-order logic over words.

Let us recall that monadic second order logic (MSO for short) is the extension of first-order logic with the ability to quantify about sets (i.e., monadic relations). Formally MSO formulae use *first-order variables* (x, y, \dots), and *monadic variables* (X, Y, \dots), and it is allowed in such formulae to quantify existentially and universally over both first-order and monadic variables, to use every boolean connectives, to use the membership predicate ($x \in X$), and every predicate of the relational structure. It is classical that over words (finite or infinite) as well as trees (finite or infinite), the expressivity of MSO coincide with standard form of automata. We expect from the reader basic knowledge concerning MSO.

Example 12. The MSO formula $\mathbf{reach}(x, y, X)$ over the signature containing the single binary predicate \mathbf{edge} (signature of a digraph):

$$\mathbf{reach}(x, y, X) ::= \forall Z. \\ (x \in Z \wedge \forall z, z'. (z \in Z \wedge z' \in X \wedge \mathbf{edge}(z, z') \rightarrow z' \in Z)) \rightarrow y \in Z$$

describes the existence of a path in a digraph from vertex x to vertex y that uses only vertices from X (but the first one): it expresses that every set Z containing x and closed under taking edges ending in X , also contains y .

We devise two (dual) extensions to the logic MSO, namely $\text{MSO}^{\leq N}$ and $\text{MSO}^{> N}$. In those extensions, one uses a single variable N of a new kind, called the *bound variable*, which ranges over non-negative integers. The logic $\text{MSO}^{\leq N}$ is obtained from MSO by allowing the extra predicate $|X| \leq N$ – in which X is some monadic variable and N the bound variable – if it appears *positively* in the formula (i.e., under the scope of an even number of negations). The logic $\text{MSO}^{> N}$ allows the same predicate but only if it appears *negatively* in the formula (i.e., under the scope of an odd number of negations). Equivalently, the logic $\text{MSO}^{> N}$ is obtained from MSO by allowing the new predicate $|X| > N$ appearing positively in the formula. The semantic of $|X| \leq N$ is, as one may expect, to be satisfied if (the valuation of) X has cardinality at most (the valuation of) N . It is already clear at this point that the negation of a $\text{MSO}^{\leq N}$ -formula is an $\text{MSO}^{> N}$ -formula and vice versa.

Given a relational structure \mathcal{S} and a non-negative integer n , we express by $\mathcal{S}, n \models \phi$ the fact that the formula ϕ is satisfied over the structure \mathcal{S} when the variable N takes the value n .

The positivity (resp. negativity) of the use of the predicate $|X| \leq N$ has straightforward consequences:

Fact 62 *For all $\text{MSO}^{\leq N}$ formula ϕ of only free variable N , all relational structure \mathcal{S} , and all $n \leq m$ in ω ,*

$$\mathcal{S}, n \models \phi \quad \text{implies} \quad \mathcal{S}, m \models \phi .$$

Dually for all $\text{MSO}^{> N}$ formula ϕ of only free variable N , all relational structure \mathcal{S} , and all $n \leq m$ in ω ,

$$\mathcal{S}, m \models \phi \quad \text{implies} \quad \mathcal{S}, n \models \phi .$$

The semantic of an $\text{MSO}^{\leq N}$ -sentence ϕ (*sentence* meaning that N is the only possible free variable in the formula) is to define a cost function over relational structures. Formally, given a relational structure \mathcal{S} , one defines the semantic of ϕ over \mathcal{S} as:

$$\llbracket \phi \rrbracket^{\leq}(\mathcal{S}) = \inf\{n : \mathcal{S}, n \models \phi\} .$$

This value can be either a non-negative integer, or ω if no valuation of N do make the sentence true. Dually, the semantic of an $\text{MSO}^{> N}$ -sentence ϕ is:

$$\llbracket \phi \rrbracket^{>}(\mathcal{S}) = \sup\{n + 1 : \mathcal{S}, n \models \phi\} .$$

This value can be a non-negative integer, or ω if ϕ is satisfied for all valuations of N . Using Fact 62 one directly gets:

Fact 63 *For all $\text{MSO}^{\leq N}$ sentence ϕ , $\llbracket \phi \rrbracket^{\leq} = \llbracket \neg \phi \rrbracket^{>}$. For all $\text{MSO}^{> N}$ sentence ϕ , $\llbracket \phi \rrbracket^{>} = \llbracket \neg \phi \rrbracket^{\leq}$.*

This is also closely related to the standard semantics of MSO, as illustrated by the following fact:

Fact 64 *For all MSO sentence ϕ , and all relational structure \mathcal{S} ,*

$$\llbracket \phi \rrbracket^{\leq}(\mathcal{S}) = \begin{cases} 0 & \text{if } \mathcal{S} \models \phi \\ \omega & \text{otherwise} , \end{cases} \quad \text{and} \quad \llbracket \phi \rrbracket^{>}(\mathcal{S}) = \begin{cases} \omega & \text{if } \mathcal{S} \models \phi \\ 0 & \text{otherwise} . \end{cases}$$

Example 13. The formula $\forall X. |X| \leq N$ calculates the size of a structure. A more interesting example makes use of Example 12. Again over the signature of digraphs, the $\text{MSO}^{\leq N}$ -sentence:

$$\text{diameter} ::= \forall x, y. \exists X. |X| \leq N \wedge \text{reach}(x, y, X).$$

defines the diameter of the di-graph: indeed, the diameter of a graph is the least n such that for all pair of states x, y , there exists a set of size at most N allowing to reach y from x (recall that x does is not necessary in X , hence this is the diameter in the standard sense).

In the standard theory of regular languages, the translation from an *MSO* sentence into a regular languages relies on the closure of regular languages under union, [intersection,] complementation, and projection. In our case, regular cost functions are closed under min, max, inf-projection, and sup-projection. Using exactly the standard technique we easily obtain:

Theorem 4. *The following items are equivalent for a cost function f over words:*

- being regular,
- being definable in $MSO^{\leq N}$,
- being definable in $MSO^{>N}$.

As a consequence, the limitedness is decidable for $MSO^{\leq N}$ and $MSO^{>N}$ -sentences over words.

6 Conclusion

We have introduced the notion of regular cost functions over words: equivalence classes of functions from words to $\omega + 1$. We have shown that those cost functions enjoy many equivalent representations: algebraic, logic and automata theoretic. This paper is mainly oriented toward the algebraic part, in particular with Theorem 2 which shows that stabilisation monoids have a semantics independent from the automata counterpart. From those equivalences we obtain that the class of regular cost functions enjoy closure under min, max, inf-projection and sup-projection. From those closure properties, it is possible to derive an equivalence with a suitable extension of monadic second-order logic (not presented in the paper). We also provide a decision procedures for the \preceq relation, and as a consequence the equivalence of cost functions. This result generalises the decidability of the limitedness problem in [23] and [1].

The results were carefully stated so that the extension of the theory to trees is possible (subject of a following paper). In particular we have introduced the notion of history-determinism, a semantic notion which replaces the classical notion of determinism in this framework. Let us finally remark that this whole framework can be extended without any problem to infinite words.

References

1. Parosh Aziz Abdulla, Pavel Krcál, and Wang Yi. R-automata. In *CONCUR*, pages 67–81. Springer, 2008.
2. Sebastian Bala. Regular language matching and other decidable cases of the satisfiability problem for constraints between regular open terms. In *STACS*, volume 2996 of *Lecture Notes in Computer Science*, pages 596–607. Springer, 2004.
3. Sebastian Bala. Complexity of regular language matching and other decidable cases of the satisfiability problem for constraints between regular open terms. *Theoretical Computer Science*, 39(1):137–163, 2006.

4. Achim Blumensath, Martin Otto, and Mark Weyer. Boundedness of monadic second-order formulae over finite words. In *36th ICALP*, Lecture Notes in Computer Science, pages 67–78. Springer, July 2009.
5. Mikolaj Bojańczyk and Thomas Colcombet. Bounds in ω -regularity. In *LICS 06*, pages 285–296, August 2006.
6. Mikolaj Bojańczyk and Thomas Colcombet. Bounds in ω -regularity. *Logical Methods in Computer Science*, 2009. To appear in the selected papers of LICS 06.
7. Thomas Colcombet and Christof Löding. The nesting-depth of disjunctive μ -calculus for tree languages and the limitedness problem. In *CSL*, number 5213 in Lecture Notes in Computer Science, pages 416–430, Bertinoro, September 2008. Springer.
8. Thomas Colcombet and Christof Löding. The non-deterministic mostowski hierarchy and distance-parity automata. In *35th ICALP*, number 5126 in Lecture Notes in Computer Science, pages 398–409, Reykjavik, July 2008. Springer.
9. Françoise Dejean and Marcel-Paul Schützenberger. On a question of eggan. *Information and Control*, 9(1):23–25, 1966.
10. L. C. Eggan. Transition graphs and the star-height of regular events. *Michigan Math. J.*, 10:385–397, 1963.
11. Gösta Grahne and Alex Thomo. Approximate reasoning in semistructured data. In *KRDB*, 2001.
12. Kosaburo Hashiguchi. A decision procedure for the order of regular events. *Theoretical Computer Science*, 8:69–72, 1979.
13. Kosaburo Hashiguchi. Limitedness theorem on finite automata with distance functions. *J. Comput. Syst. Sci.*, 24(2):233–244, 1982.
14. Kosaburo Hashiguchi. Regular languages of star height one. *Information and Control*, 53(3):199–210, 1982.
15. Kosaburo Hashiguchi. Representation theorems on regular languages. *J. Comput. Syst. Sci.*, 27(1):101–115, 1983.
16. Kosaburo Hashiguchi. Relative star height, star height and finite automata with distance functions. In *Formal Properties of Finite Automata and Applications*, pages 74–88, 1988.
17. Kosaburo Hashiguchi. Improved limitedness theorems on finite automata with distance functions. *Theor. Comput. Sci.*, 72(1):27–38, 1990.
18. Kosaburo Hashiguchi. New upper bounds to the limitedness of distance automata. *Theor. Comput. Sci.*, 233(1–2):19–32, 2000.
19. Karel Culik II and Jarkko Kari. Image compression using weighted finite automata. In *MFCs*, pages 392–402, 1993.
20. Daniel Kirsten. On the complexity of the relative inclusion star height problem. to appear in *Advances in Computer Science and Engineering in 2009*.
21. Daniel Kirsten. Desert automata and the finite substitution problem. In *STACS*, volume 2996 of *Lecture Notes in Computer Science*, pages 305–316. Springer, 2004.
22. Daniel Kirsten. Distance desert automata and the star height one problem. In *FoSSaCS*, pages 257–272, 2004.
23. Daniel Kirsten. Distance desert automata and the star height problem. *RAIRO*, 3(39):455–509, 2005.
24. Daniel Kirsten. A burnside approach to the finite substitution problem. *Theoretical Computer Science*, 39(1):15–50, 2006.
25. Daniel Kirsten. Distance desert automata and star height substitutions. Habilitation, Universität Leipzig, Fakultät für Mathematik und Informatik, 2006.

26. Hing Leung. *An Algebraic Method for Solving Decision Problems in Finite Automata Theory*. PhD thesis, Pennsylvania State University, Department of Computer Science, 1987.
27. Hing Leung. Limitedness theorem on finite automata with distance functions: An algebraic proof. *Theoretical Computer Science*, 81(1):137–145, 1991.
28. Hing Leung. *On some decision problems in finite automata*, pages 509–526. World Scientific, Singapore, 1991.
29. Hing Leung. The topological approach to the limitedness problem on distance automata, 1998.
30. Hing Leung and Viktor Podolskiy. The limitedness problem on distance automata: Hashiguchi’s method revisited. *Theoretical Computer Science*, 310(1-3):147–158, 2004.
31. Robert McNaughton. The loop complexity of pure-group events. *Information and Control*, 11(1-2):167–176, 1967.
32. Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.
33. Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.
34. Jean-Eric Pin. *Varieties of Formal Languages*. North Oxford Academic, London and Plenum, New York, 1986.
35. Imre Simon. Limited subsets of a free monoid. In *FOCS*, pages 143–150. IEEE, 1978.
36. Imre Simon. Recognizable sets with multiplicities in the tropical semiring. In *MFCS*, volume 324 of *Lecture Notes in Computer Science*, pages 107–120. Springer, 1988.
37. Imre Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72:65–94, 1990.
38. Imre Simon. On semigroups of matrices over the tropical semiring. *ITA*, 28(3-4):277–294, 1994.
39. Andreas Weber. Distance automata having large finite distance or finite ambiguity. *Mathematical Systems Theory*, 26(2):169–185, 1993.
40. Andreas Weber. Finite-valued distance automata. *Theoretical Computer Science*, 134(1):225–251, 1994.
41. Thomas Wilke. An eilenberg theorem for ∞ -languages. In *Automata, Languages and Programming*, volume 510 of *Lecture Notes in Computer Science*, pages 588–599. Springer, 1991.