A Complexity Bound for Determinisation of Min-Plus Weighted Automata

Shaull Almagor
Department of Computer Science, Technion, Haifa, Israel

Guy Arbel
Department of Computer Science, Technion, Haifa, Israel

Sarai Sheinvald
Department of Computer Science, Technion, Haifa, Israel

Abstract

The determinisation problem for min-plus (tropical) weighted automata was recently shown to be decidable. However, the proof is purely existential, relying on several non-constructive arguments. Our contribution in this work is twofold: first, we present the first complexity bound for this problem, showing it is primitive recursive. Second, our techniques introduce a versatile framework to analyse runs of weighted automata in a constructive manner. In particular, this simplifies the previous decidability argument and provides a tighter analysis, thus serving as a critical step towards a tight complexity bound.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Theory of computation → Quantitative automata

Keywords and phrases Automata, Weighted Automata, Determinisation, Tropical, Min Plus, Primitive Recursive

Digital Object Identifier 10.4230/LIPIcs.LICS.2026.5

Related Version *Full Version*: https://arxiv.org/abs/2602.01221

Funding *Shaull Almagor*: This research was supported by the ISRAEL SCIENCE FOUNDATION (grant No. 989/22).

1 Introduction

Tropical Weighted Finite automata (WFAs) are a popular quantitative computational model [8, 23, 3, 2], defining functions from words to values over the *tropical semiring* (also called the *min-plus* semiring). There, a WFA $\mathcal{A}$ assigns to a word w the *minimal* weight of a run of $\mathcal{A}$ on w, where the weight of a run is the *sum* of the weights along the run, hence min-plus.

For example, consider the tropical WFA in Figure 1. For every word $w \in \{a, b\}^*$ there are two runs: one cycles in q_a and counts the number of a's, and the other cycles in q_b and counts the number of b's. Thus, the weight of w is the minimum between the number of a's and the number of b's.

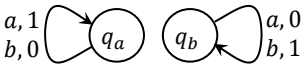

Figure 1 A tropical WFA that assigns to $w \in \{a, b\}^*$ the minimum between the number of a's and b's in w.

Applications of such weighted automata fall on a wide spectrum including verification, rewriting systems, tropical algebra, Boolean language theory, and speech and image processing (in the pre-LLM era). We refer the reader to [14, 11, 12, 18, 6, 8, 2, 1] and references therein for applications and surveys.

Unlike Boolean automata, nondeterministic WFAs are strictly more expressive than their deterministic fragment. Accordingly, a natural problem for WFAs is the *determinisation problem*: given a WFA $\mathcal{A}$, is there a deterministic WFA $\mathcal{D}$ such that $L_{\mathcal{A}} \equiv L_{\mathcal{D}}$. The deterministic fragment of WFAs admits much better algorithmic properties than general WFAs [2], making this problem important in order to increase the practical usages of this model. The determinisability problem was recently shown to be decidable in [1], after being open for over 30 years [20, 21, 16, 15].

Unfortunately, the proof of decidability in [1] is non-constructive (i.e., it is shown via two semi-algorithms). Moreover, it is heavily based on several inherently non-constructive techniques. Specifically, [1] introduces two important concepts that pave the way to decidability: *Cactus letters* and the *Zooming technique*. Cactus letters are a method for (carefully) constructing words in a recursive manner to allow pumping in a "safe" manner (as pumping arguments are notoriously difficult for WFAs [4]). The Zooming technique is a way of untangling the run structure of WFAs in order to find well-behaved infixes that can be used to obtain cactus letters. Both techniques, while useful, rely on non-constructive arguments. In a nutshell, the former constructs an infinite alphabet, and the latter relies on Ramsey-arguments for infinite sequences, and moreover – requires an infinite set of words to be applied.

Determinisability of WFAs has a well-known characterisation by means of runs, and of how far two potentially-minimal runs can get away from one another, which we commonly refer to as a *gap*. Intuitively, $\mathcal{A}$ is determinisable if and only if there is some bound $B \in \mathbb{N}$ such that if two runs on a word x obtain weights that are more than B apart, but the "upper" run can still become minimal over some suffix. For example, consider the WFA in Figure 1 and the word $a^n b^{n+1}$: upon reading a^n, the q_a component accumulates weight n and q_b accumulates 0, making it minimal and far "below" q_a. However, the suffix b^{n+1} makes q_b increase to weight $n+1$ and surpass q_a. Thus, there is a gap of size n for every n, showing that this WFA is not determinisable.

This characterisation (potentially) gives rise to a simple algorithm for determinisability: if we could show a "small-model" property for the B above (i.e., an a-priori upper bound on it, if it exists) it is enough to check whether there is some gap above this upper bound. This can be effectively checked. The only component missing is such an upper bound.

A disappointing surprise from the algorithm in [1] is that it does not provide such an upper bound, rendering the algorithm even stranger.

Our Contribution. Our main result is that the determinisation problem is decidable in primitive-recursive complexity, thus providing the first complexity upper bound for it. Moreover, our algorithm is the "expected" one: an upper bound on the gaps.

Technically, we build upon the outline of [1] for the basic framework, but deviate at the non-constructive steps. Concretely, for cactus letters we specialise the definition so that we only consider a finite alphabet, whereas for the Zooming technique we keep the intuition, but develop a new quantitative approach which enables us to apply the technique on a single word instead of an infinite sequence. The heart of our contribution is technical, and we explore it throughout this extended abstract. We remark that our work is not an "accounting" layer over [1], but rather introduces new structures and concepts.

The new tools we develop in this work present generic ideas, which we hope can be used to tackle other problems and settings. In particular, the quantitative zooming technique shows how to untangle the run tree for a concrete word, rather than an infinite sequence, which is arguably far more useful. We also generalise and demystify some of the ideas in [1] to make them more accessible and applicable.

Paper Organisation. The following extended abstract is an informal survey of the main work (available in the full version). We remark that the full version also contains this extended abstract, with specific references to the sections in the full version, which we omit here for readability. Inevitably, we recall much of the results of [1], as we build upon them and they are not (yet) standardised in literature. As we progress, we present our modifications, until reaching our central technical contributions.

In Section 2 we define the model and state basic properties. In Section 3 we recall the cactus-letters framework, and take the first step towards making it finite in Section 3.3. In Sections 4 and 5 we recall fundamental notions from [1], adapted to our new cactus letters. In Section 5.1 we deviate from these notions and present important elements in the proof. In Section 6 we introduce our new key construction – *separated repeating infix*, and present our tools for reasoning about it. In Section 7 we present the quantitative zooming technique, and in Section 8 we combine all the various elements to construct the main proof. Throughout, we refer to the full version for the precise arguments. Often, the precise details differ wildly from their informal description, due to omitted details.

2 WFAs, Determinisability, Gaps, and the Baseline-Augmented Construction

WFAs. A WFA $\mathcal{A} = \langle Q, \Sigma, q_0, \Delta \rangle$ is a finite automaton where Q is a set of states, Σ is an alphabet, $q_0 \in Q$ is an initial state[1], and $\Delta \subseteq Q \times \Sigma \times \mathbb{Z}_\infty \times Q$ is a *weighted* transition relation, assigning every p, q, σ a weight in $\mathbb{Z}_\infty = \mathbb{Z} \cup \{\infty\}$. Intuitively, weight ∞ means that there is no transition.

A *run* of $\mathcal{A}$ on a word w is a sequence of transitions reading w, and its *weight* is the sum of the weights along its transitions. We denote a run ρ from state q to state p reading w by $\rho : q \xrightarrow{w} p$, and its weight by $\mathsf{wt}(\rho)$. The *weight of a word* w, is the minimal weight of a run on w (where weight ∞ signifies having no run on w). More generally, we denote the minimal weight of a run on w from state q to p by $\mathsf{mwt}(q \xrightarrow{w} p)$, and the minimal weight of a run between sets of states Q' and P' by $\mathsf{mwt}(Q' \xrightarrow{w} P')$. An example of a WFA is depicted in Figure 1.

A *configuration* of $\mathcal{A}$ is a vector $\boldsymbol{c} \in \mathbb{Z}_\infty^Q$ which, intuitively, describes for each $q \in Q$ the weight $\boldsymbol{c}(q)$ of a minimal run to q thus far. We denote by $\mathsf{xconf}(q_0, w)$ the configuration reached after reading w starting from q_0.

We denote by $\delta_{\mathbb{B}} : Q \times \Sigma \to 2^Q$ the $\mathbb{B}$oolean transition function that corresponds to the underlying NFA induced by finite-weight transitions (and naturally extend it to $\delta_{\mathbb{B}} : 2^Q \times \Sigma \to 2^Q$). A WFA is *deterministic* if it has a single initial state, and $|\delta_{\mathbb{B}}(q, \sigma)| \leq 1$ for every $q \in Q, \sigma \in \Sigma$. It is *determinisable* if it has an equivalent deterministic WFA (i.e., they describe the same function) and is otherwise *undeterminisable*. It is well-known that not every WFA is determinisable [3]. The *determinisability* problem asks, given a WFA $\mathcal{A}$, whether it is determinisable. In this paper, we give a complexity bound on the determinisability problem.

[1] It is equivalent to consider a set of initial states, which we sometimes do for illustrations.

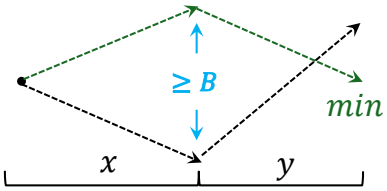

Figure 2 A B-gap witness: upon reading x, the minimal run on x is at least B below another run, but after reading y, the upper run becomes minimal.

We remark that there are several variants of WFAs: with or without initial and final weights, and with or without distinguished accepting states. The determinisability problems for these variations are inter-reducible, as shown in the full version. Thus, our definition is of the simplest variant: no initial and final weights, and all states are accepting.

2.1 Gaps and Determinisability

The starting point for discussing determinisation is the folklore characterisation by means of *gaps* discussed in Section 1 and depicted in Figure 2.

▶ **Definition 2.1** (B-Gap Witness). *For $B \in \mathbb{N}$, a* B-gap witness *consists of a pair of words x, y and a state $q \in Q$ such that*

- $\mathsf{mwt}(q_0 \xrightarrow{x \cdot y} Q) = \mathsf{mwt}(q_0 \xrightarrow{x} q \xrightarrow{y} Q) < \infty$, *but*
- $\mathsf{mwt}(q_0 \xrightarrow{x} q) - \mathsf{mwt}(q_0 \xrightarrow{x} Q) > B$.

Then, the characterisation is as follows.

▶ **Theorem 2.2** ([9, 1]). *Consider a WFA $\mathcal{A}$, then $\mathcal{A}$ is determinisable if and only if there is some $B \in \mathbb{N}$ such that there is no B-gap witness.*

2.2 Baseline-Augmented Construction

The first step in [1], which we now briefly recall, is the introduction of the *Baseline-Augmented Construction* $\widehat{\mathcal{A}}$. Intuitively, $\widehat{\mathcal{A}}$ reads a word w, but is also given a run ρ of $\mathcal{A}$ on w. It then follows the exact same runs as $\mathcal{A}$, but normalises their weight relatively to ρ. In addition, it keeps track of the reachable states.

This simple construction merely makes some information explicit, which is otherwise implicit in the runs of $\mathcal{A}$. However, it turns out that this information greatly assists in manipulating and reasoning about runs. Thus, we put it forward as our starting point of this work.

Technically, the states S of $\widehat{\mathcal{A}}$ are tuples (p, q, T) where $p \in Q$ is the current state of $\mathcal{A}$, $q \in Q$ is the *baseline state*, which is the state in the run that is being read (the *baseline run*), and $T \subseteq Q$ is the reachable set of states. $\widehat{\mathcal{A}}$ actually reads only a run (i.e., a sequence of transitions of $\mathcal{A}$), and the word is extracted from it. Thus, upon reading a transition (q, σ, c, q'), the state is updated to (r, q', T') as follows: q' follows from q, as the transition dictates. r can be any state such that $p \xrightarrow{\sigma} r$, and $T' = \delta_{\mathbb{B}}(T, \sigma)$. The weight of this transition is $d - c$, where $d = \mathsf{mwt}(p \xrightarrow{\sigma} r)$, i.e., we normalise according to the given transition. In particular, the run of $\widehat{\mathcal{A}}$ that follows the baseline in the first two components always has weight 0, and is called the *baseline run*. We denote the alphabet of $\widehat{\mathcal{A}}$ by Γ and its initial state by s_0.

A key property of $\widehat{\mathcal{A}}$ is the ability to *shift the baseline*: instead of feeding $\widehat{\mathcal{A}}$ a run $\rho : s_0 \xrightarrow{x} S$, one may choose any other run $\rho' : s_0 \xrightarrow{x} S$, preserving the gaps between the runs of $\widehat{\mathcal{A}}$, and changing only their concrete weights. This is called *baseline shift*, and we heavily rely on it, as well as develop new tools for it in Section 3.5.

Since $\widehat{\mathcal{A}}$ mirrors the run structure of $\mathcal{A}$, it is easily shown in [1] that $\widehat{\mathcal{A}}$ is determinisable if and only if $\mathcal{A}$ is determinisable. Therefore, we can reason about $\widehat{\mathcal{A}}$.

3 Cactus Letters

The standard approach in reasoning about automata is via "pumping arguments", which come in various forms. For the purpose of determinisability, pumping corresponds to increasing the gap between two runs to obtain unbounded B-gap witnesses. The main reason that decidability of determinisability has remained open for so long until [1] is (arguably) that known pumping techniques do not seem to work for increasing gaps in this manner.

The first novelty in [1] is the introduction of *cactus letters*: these are letters built up from "nesting" words, when these words exhibit certain nice structural properties in $\widehat{\mathcal{A}}$. This nesting then allows us to pump a word while keeping track of which runs can increase unboundedly, and which runs do not. In a way, this is a finer notion than the *stabilisation monoid* [7, 19].

We recall the fundamentals of cactus letters, and then explain where they fall short for obtaining complexity bounds, and how we modify them.

3.1 Reflexive and Stable Cycles

A *reflexive cycle* in $\widehat{\mathcal{A}}$ is a pair (S', w) where $S' \subseteq S$ is a subset of the states of $\widehat{\mathcal{A}}$, and w is a word such that $\delta_{\mathbb{B}}(S', w) \subseteq S'$, i.e., w causes a "cycle" on S'. We assume S' induces a baseline run on w, which therefore has weight 0. We then ask whether there is some $s \in S'$ such that $\mathsf{mwt}(s \xrightarrow{w^k} s) < 0$ for some $k \in \mathbb{N}$. If not, we call (S', w) a *stable cycle*. Thus, in a stable cycle the minimal cycles on w^k for any k are of weight 0.

The fundamental property of stable cycles is that for each pair of states s, t S', either $\mathsf{mwt}(s \xrightarrow{w^k} t)$ tends to ∞ as k increases, or $\mathsf{mwt}(s \xrightarrow{w^k} t)$ is bounded. In the latter case, we say that (s, t) are a *grounded pair*, and the value $\mathsf{mwt}(s \xrightarrow{w^{2k\mathfrak{m}}} t)$ is constant for all k, where $\mathfrak{m} = |S| \cdot |S|!$ is the *stabilisation constant*. Moreover, we can exactly characterise the grounded pairs: (s, t) are grounded if there is some state $r \in S'$ such that $s \xrightarrow{x^{\mathfrak{m}}} r \xrightarrow{x^{\mathfrak{m}}} t$ and there is a run $\tau : r \xrightarrow{x^{\mathfrak{m}}} r$ such that $\mathsf{wt}(\tau) = 0$, i.e., r is a *minimal reflexive state*.

To gain some intuition on stable cycles, consider the runs of $\widehat{\mathcal{A}}$ on some long word. Eventually, the subset S' of states reachable by $\widehat{\mathcal{A}}$ repeats on some infix x. Then, (S', x) is certainly a *reflexive* cycle. The question of whether it is stable depends on whether enough repetitions of x yield any negative cycles in S'. If there are no negative cycles, and the minimal run has weight 0, then it is a stable cycle. As we discuss in Section 3.5, $\widehat{\mathcal{A}}$ allows us to manually change the baseline run in order to guarantee that the minimal run indeed has weight 0, thus obtaining a stable cycle.

3.2 Original Cactus Letters

Consider a stable cycle (S', w). Since we understand the behaviour of runs upon reading $w^{2k\mathfrak{m}}$ for any k, we can define a new letter $\alpha_{S',w}$ such that for every grounded pair (s, t) we have $\mathsf{mwt}(s \xrightarrow{\alpha_{S',w}} t) = \mathsf{mwt}(s \xrightarrow{w^{2\mathfrak{m}}} t)$, and otherwise we have $\mathsf{mwt}(s \xrightarrow{\alpha_{S',w}} t) = \infty$. Intuitively, non-grounded pairs jump to ∞. Thus, in a way, cactus letters capture unboundedly many iterations of w.

Notice that in this way we potentially introduce infinitely many letters (as w is unbounded). The next step, however, is even more dramatic: we again consider stable cycles over this new infinite alphabet, define the corresponding cactus letters (which now nest within them cactus letters), and continue in this fashion ad-infinitum, obtaining the *cactus alphabet* Γ_∞. We define $\mathsf{dep}(\alpha_{S',w})$ to be the depth of nesting of cactus letters in $\alpha_{S',w}$ (with "standard" letters being of depth 0). This is depicted in Figure 3.

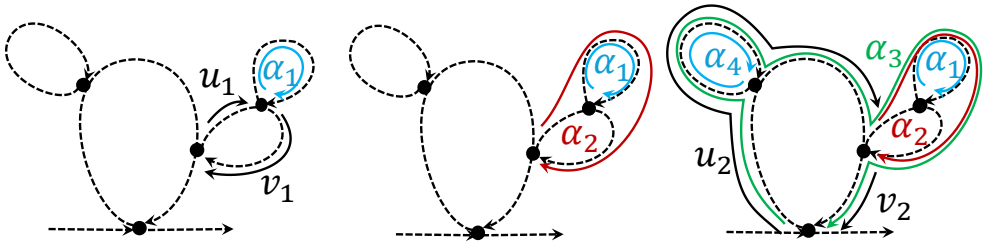

Figure 3 Cactus letters: α_1 is of depth 1 (and its inner word is of depth 0). The word $u_1\alpha_1 v_1$ (left) forms a cactus letter α_2 of depth 2 (center). Then, $u_2\alpha_2 v_2$ forms a cactus letter α_3 of depth 3 (right). Notice that u_2 contains also the cactus letter α_4.

A-priori, the alphabet Γ_∞ is infinite both due to arbitrarily long words in cactus letters, and both due to unbounded depth. Fortunately, the latter is already handled (in a sense) in [1], by showing that for depth greater than $|S|$, any cactus letter contains a *degenerate* cycle – one whose reachable state pairs are all grounded, and therefore does not introduce any "interesting" behaviours. However, the words in the cactus letters may still be arbitrarily long. This is the first place where the proof in [1] becomes non-constructive, and where we begin our contribution.

3.3 Effective Cactus Letters

Our first goal is to limit the alphabet of cactus letters to a finite subset. Naturally, this should be done in a way that facilitates the remainder of the proof. Therefore, we need to bound not only the depth of the letters we consider, but also the length of their "inner" word. That is, bound $|w|$ for the letter $\alpha_{S',w}$. The catch is what bound we can give so that we can still find some structure in the runs that can be used for some form of pumping. Since such a bound depends on the proof structure, we leave it unspecified at this point, and accumulate requirements on it as we progress through the proof, eventually building the concrete bound. At this stage, we only explain what this length bound depends on.

The length of words we allow is not a fixed constant, but depends on the depth of the cactus letters being considered. The technical reason is the inductive nature of the proof, but it is also conceptually sensible: consider e.g., a letter $\alpha_{S_1,ab\alpha_{S_2,w_2}cd}$, then we would expect to be able to "unfold" the inner cactus α_{S_2,w_2} at least to its stable version $w_2^{2\mathfrak{m}}$ and still have a legal cactus letter $\alpha_{S_1,abw^{2\mathfrak{m}}cd}$. To do so, the length bound for the outer letter must be much higher than the inner letter.

In the following, we therefore have a function $L : \mathbb{N} \to \mathbb{N}$ such that for every cactus letter $\alpha_{S',w}$, if $\mathsf{dep}(\alpha_{S',w}) = d$ then $|w| \leq L(d)$. Moreover, we limit ourselves to letters of depth $|S|$. This renders the alphabet finite, as we required. We denote this alphabet by Υ.

Unfortunately, we recommend not getting too attached to L, since in Section 7.1 we actually need *two* such bounding functions and two corresponding alphabets in order to obtain our results, but we defer this problem for now.

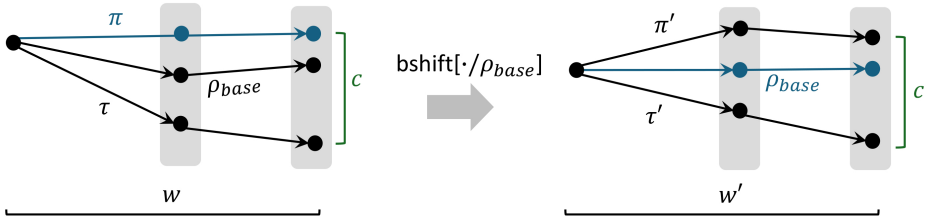

Figure 4 Shifting the baseline: ρ_{base} becomes the baseline run instead of π; the word w changes to $w' = \mathsf{bshift}[w/\rho_{\text{base}}]$, and the runs π and τ change accordingly to $\pi' = \mathsf{bshift}[\pi/\rho_{\text{base}}]$ and $\tau' = \mathsf{bshift}[\tau/\rho_{\text{base}}]$, respectively. The shifted runs keep their relative distance from one another.

3.4 Cactus Unfolding

Cactus letters are useful to capture runs on $w^{2\mathbf{m}k}$ for any k. However, we sometimes want to extract these concrete runs back. Intuitively, we can "unfold" a cactus letter $\alpha_{S',w}$ into $w^{2\mathbf{m}k}$ for any k. Due to the characterisation of grounded pairs, we know what this unfolding does: the grounded pairs maintain their weight, whereas ungrounded pairs increase with k.

We borrow this tool with only cosmetic changes from [1], and therefore bring only the essentials here. We have two operators: the first is $\mathsf{unfold}(u\alpha_{S',w}v \wr F) = uw^{2\mathbf{m}M_0}v$, which unfolds a single cactus letter in such a way that all ungrounded pairs yield runs with weight greater than F after the $w^{2\mathbf{m}M_0}$. The second is $\mathsf{flatten}(w \wr F)$ which recursively applies unfold to all cactus letters. The resulting word is in Γ^*. We remark that here we sweep under the rug many important details.

Due to unfolding a new entity is introduced, called *ghost states*: an ungrounded pair (s,t) of $\alpha_{S',w}$ does not yield runs over $\alpha_{S',w}$, but does yield them on $w^{2\mathbf{m}k}$ for every k. We think of such states as "ghosts" floating high above the other runs. When unfolding, we can increase their weight arbitrarily. The states that are reachable as ghosts upon reading x are denoted $\delta_{\text{👻}}(s_0, x)$. These play a significant role later on.

3.5 Baseline Shifts

As mentioned in Section 2.2, $\widehat{\mathcal{A}}$ offers a convenient way to shift perspective between runs, called *baseline shifts*, by changing the word fed to $\widehat{\mathcal{A}}$ to represent a different run of $\mathcal{A}$. Once cactus letters are introduced, however, it is no longer clear what this means, since cactus letters are not transitions of $\mathcal{A}$. Fortunately, in [1] some tools are developed to support these shifts also in the presence of cactus letters. Technically, this involves the introduction of a new set of letters called *rebase letters*, and a new hierarchy on top of the cactus hierarchy. Our contribution uses this framework, but does not develop it further, and therefore we sweep it under the rug in this abstract, leaving the details to the full version.

We thus assume there is an operator $\mathsf{bshift}[\cdot/\cdot]$ with the following usage and properties: given a word $w \in \Upsilon^*$ and some run $\rho_{\text{base}} : s_0 \xrightarrow{w} S$ of $\widehat{\mathcal{A}}$, there is a word $w' = \mathsf{bshift}[w/\rho_{\text{base}}]$ whose baseline run is $\mathsf{bshift}[\rho_{\text{base}}/\rho_{\text{base}}]$, and for every two runs $\pi, \tau : s_0 \xrightarrow{w} S$ of $\widehat{\mathcal{A}}$, we have corresponding runs $\pi' = \mathsf{bshift}[\pi/\rho_{\text{base}}]$ and $\tau' = \mathsf{bshift}[\tau/\rho_{\text{base}}]$ such that $\mathsf{wt}(\pi) - \mathsf{wt}(\tau) = \mathsf{wt}(\pi') - \mathsf{wt}(\tau')$. Moreover, this holds for every prefix of the runs and the words. Intuitively, the entire run structure is maintained, but the baseline changes. This is illustrated in Figure 4.

A minor contribution of this work is an extension of baseline shifts beyond runs and words to e.g., sets and cycles, so we can write $\mathsf{bshift}[(S', w)/\rho_{\text{base}}]$ to mean a shift of a reflexive cycle. In addition, we establish new results about the invariance of certain structures to baseline shifts. We believe that this framework is extremely convenient, and may prove useful in other cases of reasoning about WFAs and Counter Automata.

3.6 From Reflexive Cycles to Stable Cycles

Finding stable cycles is a difficult task, due to their strict requirement on minimal cycles. This is a source of complication in [1], and leads to a lot of branching in the proof structure. Moreover, there a stable cycle is found by considering infinite families of words, which is a non-constructive technique we cannot afford.

Our first contribution is that given a reflexive cycle, we can shift it to find a stable cycle, at the cost of possibly elongating the word by at most $|S|$ repetitions. We sketch the idea as a gentle introduction to baseline shifts.

▶ **Lemma 3.1.** *Consider a reflexive cycle (S', w), then there exists $k \leq |S|$ and a cycle $\rho : s \xrightarrow{w^k} s$ with $s \in S'$ such that $\mathsf{bshift}[(S', w)/\rho]$ is a stable cycle.*

Proof Sketch. Assume by way of contradiction that (S', w) is not a stable cycle. Thus, there is some run $\rho : s \xrightarrow{w^k} s$ with $s \in S'$ and $\mathsf{wt}(\rho) < 0$. We look for a run whose *slope* is minimal, i.e., it has the steepest descent. We show that this minimal slope is attained by $k \leq |S|$.

We then perform a baseline shift on this ρ, using the word w^k. We show that $\mathsf{bshift}[(S', w^k)/\rho]$ is a reflexive cycle, which has no negative-slope cycles, and is therefore a stable cycle. ◀

The implication of this result is that it is now fairly easy to find stable cycles: in long-enough words, we must reach a reflexive cycle by simple pigeonhole principle. Then, we can elongate the word somewhat and perform a baseline shift to obtain a stable cycle.

4 (Effective) Dominance, Charge and Potential

Recall that our goal is to reason about gap witnesses (Theorem 2.2). There, we compare a minimal run on a prefix x with a higher run that can potentially become minimal by reading y. A key insight from [1] (that is not made explicit there) is that the two attributes – being minimal versus being *potentially* minimal, are very different. As it turns out, separating them opens up new avenues of reasoning that ultimately pave the way to decidability of determinisation. Technically, they are separated by incorporating the baseline run as a separator between them. To this end, the concepts of *charge* (ψ) and *potential* (ϕ) are introduced. The charge simply measures the weight of the minimal run. Technically, it is negated to keep it positive (as the minimal run is always below the baseline, which is 0). The potential is somewhat more involved, and measures how high a run can get, so that there is a suffix from which it reaches a finite weight, but all runs below it jump to ∞.

In this work, the definition is adapted to account for the finite alphabet. Specifically, we no longer allow arbitrary suffixes in the definition of the potential, but rather very specific ones. We give an intuitive version of the definitions. Consider a configuration $\boldsymbol{c}$. We define the *charge* $\psi(\boldsymbol{c}) = \min\{\boldsymbol{c}(q) \mid q \in S\}$ to be the minimal weight. Next, we fix a language L_{dom} of "legal suffixes" (detailed in the full version). We say that a state $q \in S$ is *dominant* in $\boldsymbol{c}$ if there is a suffix $z \in L_{\text{dom}}$ such that $\mathsf{mwt}(q \xrightarrow{z} S) < \infty$ but $\mathsf{mwt}(q' \xrightarrow{z} S) = \infty$ for every q'

such that $\boldsymbol{c}(q') < \boldsymbol{c}(q)$. If q has the highest weight in the configuration with this property, it is *maximal dominant*, and we define $\phi(\boldsymbol{c}) = \boldsymbol{c}(q)$. For a word w, we define $\phi(w)$ as the potential on the configuration reached after reading w.

The new notions come with some important tools. First, as part of our development of baseline shifts, we show that dominance is maintained through baseline shift. That is, if q is dominant in $\boldsymbol{c}$, then $\mathsf{bshift}[q/\rho]$ is dominant in $\mathsf{bshift}[\boldsymbol{c}/\rho]$. This may seem like a trivial result by shifting the suffix "accordingly". Unfortunately, there is no clear way to define this. The technicalities of this proof is what gives rise to the exact definition of L_{dom}; see the full version for the details. Note that the potential and charge may change under baseline shift, due to the change in baseline.

The potential satisfies a very nice property called *bounded growth*, namely that $|\phi(w \cdot \sigma) - \phi(w)|$ is bounded by a constant depending on the alphabet Υ. This renders the potential well-behaved, and is used extensively later on. We remark that this is a useful simplification from [1], as there this property fails in general, due to the infinite alphabet.

Unfortunately, this property does not hold for charge: the minimal run can increase abruptly by not being able to continue, and making a higher run suddenly minimal. This is where ψ and ϕ significantly differ, and why separating them is so useful. In order to overcome this, we slightly sharpen a result from [1], stating the following:

▶ **Lemma 4.1.** *For every $P \in \mathbb{N}$ and $u\sigma \in \Upsilon^*$ with $\mathsf{dep}(\sigma) = d$, if $\psi(u) - \psi(u\sigma) > P + \mathtt{GMaxWt}(d)$, then there is a word w such that $\phi(w) > P$.*

$\mathtt{GMaxWt}(d)$ is a function that bounds the maximal weight over any transition on a letter from Υ of depth d. In order to define it, we first need to establish the length bound $L(d)$ which we are yet to do. This important lemma gives a precise and effective connection between the potential and the charge, and is used to split our main proof into two conceptual parts: one dealing with the case where the charge has bounded growth, and one dealing with the case where the potential can be very high.

Another small observation is that ψ plays nice with unfolding, in the sense that if $x = \mathsf{unfold}(u\alpha_{S',w}v \wr F)$ then $\psi(x) \geq \psi(u\alpha_{S',w}v)$. We remark that a dual property for ϕ does not hold, and this is of central importance in the following.

5 Witnesses

The surprising element in [1] is that the characterisation of determinisability is not via unbounded gaps, but rather through a new notion called *witness*. In this work, we restrict the notion of witnesses in a way that retains this characterisation, but works over our finite alphabet. In a nutshell, the changes in this definition are a direct result of the changes in the definition of dominance. We bring a semi-precise definition.

▶ **Definition 5.1.** *A* witness *consists of $w_1, \alpha_{S_1,w_2}, w_3$ over Υ such that $S_1 = \delta_{\text{Ⓐ}}(s_0, w_1)$, and the following holds:*

- $\mathsf{mwt}(s_0 \xrightarrow{w_1\alpha_{S_1,w_2}w_3} S) = \infty$, *but*
- $\mathsf{mwt}(s_0 \xrightarrow{w_1 w_2^{2\mathbf{m}} w_3} S) < \infty$.

Intuitively, since $w_1\alpha_{S_1,w_2}w_3$ gets weight ∞, but $w_1 w_2^{2\mathbf{m}} w_3$ get finite weight, it means that the minimal runs on the latter go through ghost states. More precisely, it means that the maximal dominant state after reading $w_1 w_2^{2\mathbf{m}}$ is a ghost state in $\delta_{\text{Ⓐ}}(s_0, w_1\alpha_{S_1,w_2})$.

Since ghost states can be raised arbitrarily high using flattening, this serves to give the intuition that a witness implies unbounded gaps. Indeed, the main result of [1] is that a witness exists if and only if $\widehat{\mathcal{A}}$ is undeterminisable, which readily leads to decidability via two semi algorithms (with the "if" direction being the bulk of the work).

We bring a crucial result, heavily abridged here, relating to witnesses, whose proof is similar (up to alphabet-based changes) to the analogue in [1]. The full version contains the formal statement and proof.

▶ **Lemma 5.2.** *Either there is a witness, or the following holds. Consider a word* $u\alpha_{B,x}v \in \Upsilon^*$ *and let* $ux^{2\mathfrak{m}k}v = \mathsf{unfold}(u, \alpha_{B,x}, v \wr F)$ *for large-enough* F*. For every prefix* v' *of* v *we have* $\phi(ux^{2\mathfrak{m}k}v') = \phi(u\alpha_{B,x}v')$.

The way this should be read is: "either the potential unfolds nicely, or we are done". Indeed, once we establish the existence of a witness, we know $\widehat{\mathcal{A}}$ is undeterminisable, which is our goal. This lemma plays a central role in the main proof.

5.1 On ϕ, ψ and two length functions

We can now make the following vague observation: from Lemma 4.1 we know that in some cases, properties of ψ can induce high potential. Similarly, from Lemma 5.2 we know that in some cases, properties of the potential induce a witness. In Section 8 our proof takes us from high ψ to high ϕ, and from there to witnesses. However, these two steps have several technical differences. Henceforth, we often separate definitions and results to ψ and ϕ.

We start by separating the length functions mentioned in Section 3.3: we now have two length functions, *simple* (L_{simp}) and *general* (L_{gen}), which give rise to two effective cactus alphabets Υ_{simp} and Υ_{gen}. The constraints on each function become clearer in Sections 6 and 8.

6 Separated Repeating Infixes

We can finally start describing our contribution in earnest. The first part of our technical contribution is the introduction of *Separated Repeating Infixes (SRI)*. Intuitively, these are words whose run structure exhibits certain organised behaviours, that can be (carefully) used for pumping and for shortening words. The idea is based on the notion of separated increasing infixes from [1], but the technical details differ in many significant ways, and the tools we develop here are new. See the full version for a detailed comparison.

We start with the basic definition. An SRI is a word $uxyv$ with the following structure (see Figure 5):

- Reading u from s_0 reaches a configuration that is partitioned to "sub-configurations" that have a huge *gap* G between them, we call the state sets in these sub configurations $V_1, \ldots, V_\ell$.
- Next, reading x maintains the exact structure of these V_j sets, and shifts the weight of the states in each set V_j by a constant $k_{j,x}$. In particular, the entire configuration maintains its support after x.

  An additional property of x is that it is quite short – enough so that if needed, we can turn x^k into a stable cycle as per Lemma 3.1 without exceeding the length bound $L(\mathsf{dep}(x))$.
- Reading y has very similar behaviour to x, in the sense that the V_j are maintained, and each is increased by a constant $k_{j,y}$, and the sign of $k_{j,y}$ is the same as that of $k_{j,x}$, so each V_j is decreasing/increasing on both x and y.
- Finally, v is just a harmless suffix.

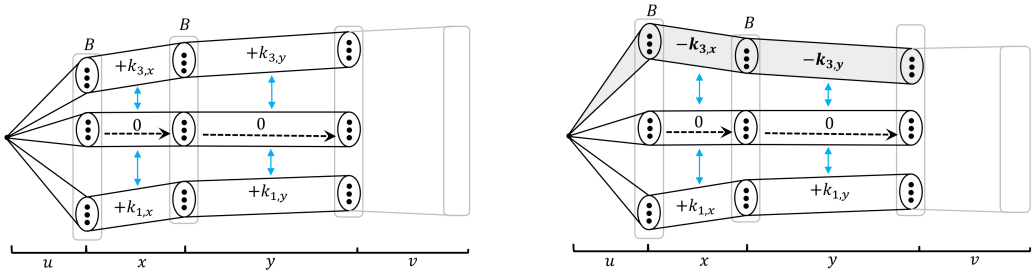

Figure 5 An SRI. After reading u, the configuration is separated to sub-configurations corresponding to the V_i. Reading x maintains this partition and the exact distances within each set. The same holds for y, but with different shifts. The gaps between the sub-configurations are much larger than the effect of x and y. On the left is a positive SRI: all sub-configurations are trending upwards. On the right is a negative SRI: There exists a sub-configuration (the upper one in the right figure) with a downwards slope.

An underlying theme of this work is that increasing and decreasing runs are inherently different. This manifests also in SRI, where sub-configurations with $k_{j,x} < 0$ are different to those with $k_{j,x} \geq 0$. We formalise and extend this by defining different *flavours* of SRI. An SRI is:

- *Negative* if $k_{j,x} < 0$ for some j.
- *Positive* if $k_{j,x} \geq 0$ for all j.
- *Stable* if $(\delta_{\text{🔔}}(s_0, u), x)$ is a stable cycle. In this case it is *Degenerate* if $(\delta_{\text{🔔}}(s_0, u), x)$ is degenerate, and *Non-degenerate* otherwise.

We also distinguish between SRIs that pertain to ϕ and to ψ, and specialise the definition for each one. The definitions here (and also in other places) make use of the *maximal effect* of a word w, denoted $\mathsf{max\_eff}(w)$, which is an upper bound on the change in weight upon reading w. Also, recall that G is the gap size of the SRI, and that $L_{\mathsf{simp}}, L_{\mathsf{gen}}$ are two length functions giving rise to two alphabets $\Upsilon_{\mathsf{simp}}, \Upsilon_{\mathsf{gen}}$ which we do not elaborate on here.

- A *simple* SRI (SSRI) is over Υ_{simp}, has $G = 4\mathfrak{m}\mathsf{max\_eff}(xy)$, and requires $|x| \leq \frac{1}{\mathfrak{m}} L_{\mathsf{simp}}(\mathsf{dep}(x) + 1)$ and $\phi(u) \leq \phi(ux) \leq \phi(uxy)$.
- A *general* SRI (GSRI) is over Υ_{gen}, has $G = 4\mathfrak{m}\mathsf{max\_eff}(xy) + \mathbf{H}$, and requires $|x| \leq \frac{1}{\mathfrak{m}} L_{\mathsf{gen}}(\mathsf{dep}(x) + 1)$ and $\psi(u) \geq \psi(ux) \geq \psi(uxy)$, where $\mathbf{H}$ is a constant explained later.

The bounds involved in the definition (including $\mathbf{H}$) arise from the proof as we go along. In the following, we use SRI as a general term for either an SSRI or GSRI.

SRIs serve as a bridge between two parts of the proof: on the one hand, if a word can be decomposed as an SRI, we show that we can use its structure to obtain certain properties (mostly – shortening the word while maintaining ϕ or ψ). Showing such properties is technically challenging, and makes extensive use of baseline shifts and dominance. On the other hand, showing that SRIs can be found at all is challenging, and requires us to develop a *quantitative zooming technique*.

Didactically, the correct way to proceed with the proof is to ask how large a B-gap witness we need to guarantee the existence of an SRI, and then show how an SRI helps us once we find it. It is more concise, however, to develop our tools "bottom-up" and combine everything at the end. We therefore start by showing what we can do given an SRI.

6.1 SRI – A Toolbox

Our first result identifies what kinds of runs are allowed when reading x and y in an SRI. Intuitively, it shows that there are no runs from a "low" V_j to a "higher" $V_{j'}$.

▶ **Proposition 6.1.** *Consider an SRI $uxyv$ and $\rho : p \xrightarrow{x} q$ with $p \in V_j$, then $q \in V_{j'}$ with $j' \leq j$. Moreover, there exists some $p' \in V_{j'}$ such that $p' \xrightarrow{x} q$. We also have $|k_{j,x}| \leq \mathsf{max\_eff}(x)$.*

The same holds for y.

Proof Sketch. The idea is simple, and appears already in [1]: if there is a run that goes from e.g., V_2 to V_3, then the gap between V_2 and V_3 (after reading x or y) would be at most e.g., $2\mathsf{max\_eff}(x)$. Indeed, two runs on x stemming from the same state cannot diverge by more than $2\mathsf{max\_eff}(x)$.

Note that in order to make the proof work, we must ensure that the gap between the sets is much higher than $2\mathsf{max\_eff}(x)$, which is part of the reason behind the choice of G above. ◀

We now consider Stable SRIs, which are the strictest flavour. In the full version, we show that every Stable SRI is in particular Positive. This is not surprising, since stable cycles do not allow negative cycles. For Degenerate Stable SRIs, we have that x is a degenerate stable cycle. As the name suggests, this induces a degenerate behaviour. Concretely, we show that we can get rid of x entirely.

▶ **Proposition 6.2.** *For a degenerate SRI $uxyv$ we have $\psi(uxyv) = \psi(uyv)$ and $\phi(uxyv) = \phi(uyv)$.*

For non-degenerate SRIs, things are more involved. Fortunately, techniques for this fragment are developed in [1], and require only small adaptations. Intuitively, we show that we can replace x with $\alpha_{S',x}$ and drop y entirely, while keeping ϕ and ψ well behaved. We remark that the following result is what gives rise to $G \geq 4\mathfrak{m}\mathsf{max\_eff}(xy)$ for both SSRI and GSRI.

▶ **Lemma 6.3.** *Consider a Stable Non-degenerate SRI $w = uxyv$ and let $w' = u\alpha_{S',x}v$, then*

- *$\psi(w) \geq \psi(w')$, and*
- *either there is a witness, or $\phi(w) \leq \phi(w')$.*

Naturally, the "either there is a witness" clause stems from Lemma 5.2.

Our first major roadblock is Negative SRIs. Indeed, suppose V_3 has $k_{3,x} < 0$ and contains a maximal dominant state, but $k_{2,x} = 0$ and V_2 contains the baseline run. Then, ϕ may *decrease* with pumping. Dually, if a negative V_j is below the baseline, this may cause ψ to *increase*. Such behaviours are very problematic and we would like to get rid of them. The problem goes even further: even in a Positive SRI, there may be negative *ghost runs* on x (c.f., Section 3.4). These also cause problems if we want to turn x to a stable cycle.

The following result handles these scenarios using careful baseline shifts, and identifies what happens to the resulting SRI. Intuitively, the lemma considers a setting where there is a negative cycle on some x^k from a reachable or ghost-reachable state s. Then, by Lemma 3.1 we can shift x^k to obtain a reflexive cycle (S'', x'). We then ask what this stable cycle does to the states in various V_j's. The lemma says that if $V_1, \ldots, V_{\ell'}$ (i.e., the lowest ℓ' sets, for some $\ell' \leq \ell$) all have $k_{i,x} \geq 0$, then upon reading $\alpha_{S'',x'}$ they all jump to ∞.

▶ **Lemma 6.4.** *Consider an SRI $uxyv$ where $k_{j,x} \geq 0$ for all $0 \leq j \leq \ell'$ for some $\ell' \leq \ell$ (i.e., the first ℓ' $k_{j,x}$ are positive). Assume there is a minimal-slope run $\rho : s \xrightarrow{x^k} s$ with $s \in \delta_{\mathbb{N}}(s_0, u)$, $k \leq |S|$ and $\mathsf{wt}(\rho) < 0$. Then $(S'', x') = \mathsf{bshift}[(\delta_{\mathbb{N}}(s_0, u), x^k)/\rho]$ is a stable cycle, and for every state $s' \in V_j$ with $j \leq \ell'$ we have $\mathsf{mwt}(\mathsf{bshift}[s'/\rho] \xrightarrow{\alpha_{S'',x'}} S) = \infty$.*

Proof Sketch. Intuitively, we proceed in four steps as depicted in Figure 6. First, we baseline shift x^k on the negative run ρ, resulting in a stable cycle (S'', x') as per Lemma 3.1.

We want to use the cactus letter $\alpha_{S'',x'}$. However, we need it to be effective, so as not to go beyond our effective alphabet. This is where the requirement on the length of x in an SRI comes into play: since $k < |S| \ll \mathfrak{m}$ and $|x| \leq \frac{1}{\mathfrak{m}} L_{\mathsf{simp}}(\mathsf{dep}(x)+1)$ (or similarly for L_{gen}), then $|x^k| \leq L_{\mathsf{simp}}(\mathsf{dep}(x)+1)$, so we can consider the corresponding cactus letter as per Section 3.3.

We now consider a state s_1 in some V_j with $j \leq \ell'$, i.e., one of the "low and non-negative" V_j's. The shifted version of s_1 is $t_1 = \mathsf{bshift}[s_1/\rho]$, and we examine what happens when it reads $\alpha_{S'',x'}$. If, by way of contradiction, there is a (finite-weight) run $t_1 \xrightarrow{\alpha_{S'',x'}} t_2$, then (t_1, t_2) is a grounded pair for (S'', x') (as per Section 3.1). This means that there exists a minimal reflexive state $t_3 \in S''$ and a run $\tau : t_3 \xrightarrow{x'^{\mathfrak{m}}} t_3$ with $\mathsf{wt}(\tau) = 0$ such that $t_1 \xrightarrow{x'} t_3$.

We now "pull back" t_3 from the baseline ρ to our original baseline of $uxyv$ using another baseline shift, denoting the corresponding state t'_3. Thus, there is a run $s_1 \xrightarrow{x^k} t'_3$. By Proposition 6.1 we have that $t'_3 \in V_{j'}$ for some $j' \leq j$ (recall that runs cannot go to "higher" sub-configuration). In particular, by the premise of the lemma we have $k_{j',x} \geq 0$, so there are no negative cycles on x generated from t'_3.

However, we now consider the run τ, and similarly pull it back to a run $\tau' : t'_3 \xrightarrow{x^k} t'_3$. Since baseline shifts preserve the run structure and since $\mathsf{wt}(\tau) = 0$, then $\mathsf{wt}(\tau')$ must be "0 with respect to $\mathsf{wt}(\rho)$". More precisely, we have

$$\mathsf{wt}(\rho) - \mathsf{wt}(\tau') = \mathsf{wt}(\mathsf{bshift}[\rho/\rho]) - \mathsf{wt}(\tau) = 0$$

Thus, $\mathsf{wt}(\tau') = \mathsf{wt}(\rho) < 0$. This means that there is a negative cycle from t'_3 on x^k. It is now not difficult to reach a contradiction with the fact that $k_{j',x} \geq 0$ (see the full version for details).

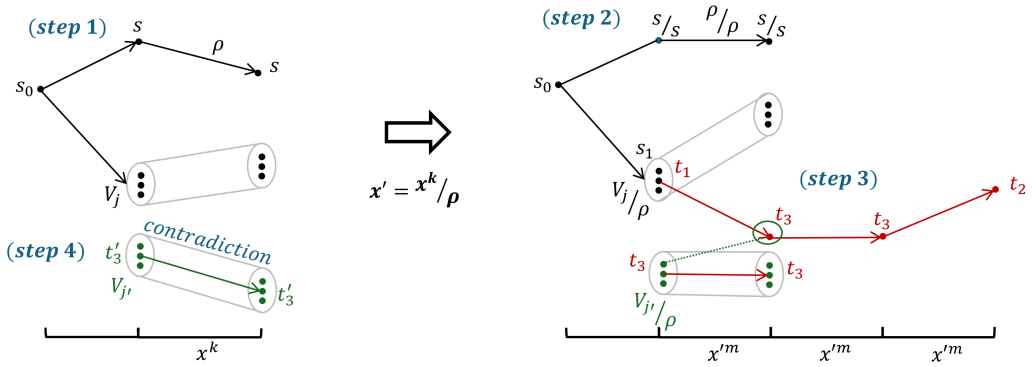

Figure 6 Proof of Lemma 6.4. In step 1 we consider the decreasing run ρ over x^k. We baseline shift to it to obtain the figure on the right. In step 2 we consider some increasing V_j, and assume there is a finite-weight run from it (the red run). We then identify a state t_3 with a 0-weight run. In step 3 we pull t_3 back to the left drawing (t'_3) so it becomes a negative run. In step 4 we reach a contradiction, since this negative run is lower than V_j. ◀

Equipped with general tools for SRI, we proceed to develop tools specific for SSRI and GSRI.

6.2 SSRI – A Toolbox

Recall that SSRI pertain to ϕ. In the main proof (Section 8) when we reach an SSRI our goal is either to find a witness, or to shorten the SSRI while not decreasing ϕ. We present the relevant tools to achieve this. Equipped with Lemma 6.4, we start by tackling Negative SSRI.

▶ **Lemma 6.5.** *If there exists a Negative SSRI, then there exists a witness.*

Proof Sketch. Consider a Negative SSRI $w = uxyv$. The proof proceeds in four steps.

Recall that a dominant state has a suffix it can read with finite weight, and that all states below it cannot. In Step 1 we show that the maximal dominant states at $\boldsymbol{c_u}$ and $\boldsymbol{c_{ux}}$ are the same. This is simple, since the ordering of the states induced by $\boldsymbol{c_u}$ and $\boldsymbol{c_{ux}}$ is maintained, due to the large gaps between sub-configurations, and the identical sub-configurations before and after reading x.

In Step 2 we start our path towards a witness by flattening the prefix u (as per Section 3.4). We then ask what this does to ϕ. Recall that flattening is repeated unfolding. We therefore call upon Lemma 5.2, and get that either there is a witness (in which case we are done), or that flattening does not affect the potential and the dominant states. Note that after flattening, the ghost states are reachable.

In Step 3 we consider the minimal-slope run on the x infix (rather, on any x^k), and baseline shift on this run, thus inducing a stable cycle x' as per Lemma 3.1. Note that since the SSRI is negative, this run has negative slope indeed. We also shift the (flattened) prefix u so that it "glues" properly to the shifted cycle. We then keep track again of the maximal dominant state, and show (using our result that dominance is shift-invariant, described in Section 4) that it corresponds to the shift of the dominant state at $\boldsymbol{c_u}$. We also designate a suffix z that shows it is dominant.

Finally, in Step 4 we show that the resulting tuple of (prefix,stable cycle,suffix) forms a witness. Specifically, we consider $(u, \alpha_{S'',x'}, z)$. By Definition 5.1 we want to show that $\mathsf{mwt}(s_0 \xrightarrow{u\alpha_{S'',x'}z}) = \infty$, but that $\mathsf{mwt}(s_0 \xrightarrow{ux'^{2\mathbf{m}}z}) < \infty$. First, since in an SSRI x does not change the reachable states, this also holds for x'. Thus, the dominant state t in $\boldsymbol{c_u}$ is reachable also by ux'. By definition, z can be read from t (as it is the separating suffix showing that t is dominant). We therefore have

$$\mathsf{mwt}(s_0 \xrightarrow{ux'^{2\mathbf{m}}z}) \leq \mathsf{mwt}(s_0 \xrightarrow{ux'} t \xrightarrow{z} S) < \infty$$

Assume by way of contradiction that there is a finite weight run $\tau : s_0 \xrightarrow{u} s'_1 \xrightarrow{\alpha_{S'',x'}} s'_2 \xrightarrow{z} s'_3$. This means that (s'_1, s'_2) is a grounded pairs. Using similar baseline shifts as in the proof of Lemma 6.4, we show that this puts s'_2 in a set V_j with negative $k_{j,x}$.

We then compare the weight of s'_2 to the weight of the maximal dominant t. Intuitively, we show the following:

- If s'_2 is below t, then it cannot read z by the definition of dominance (which is a contradiction to the assumption).
- If s'_2 is in the same set V_j as t, then since $k_{j,x}$ is negative, this would mean that the ϕ is *decreasing* along x, in contradiction to the definition of SSRI.
- If s'_2 is way above t, then we can show it is maximal dominant, contradicting the maximal dominance of t. ◀

Having handled Negative SSRI, we now turn our attention to Positive SSRI.

▶ **Lemma 6.6.** *Either there is a witness, or every Positive SSRI is a Stable SSRI.*

Proof Sketch. Consider a Positive SSRI $uxyv$. If $(\delta_{\text{🔔}}(s_0, u), x)$ is a stable cycle, then $uxyv$ is a Stable SSRI and we are done. Assume $(\delta_{\text{🔔}}(s_0, u), x)$ is not a stable cycle, then there is some negative-slope run $\rho : s \xrightarrow{x^k} s$ for some k and $s \in \delta_{\text{🔔}}(s_0, u)$. However, since x is part of a Positive SSRI, then no reachable state has such a cycle, i.e., $s \in \delta_{\text{🔔}}(s_0, u) \setminus \delta_{\mathbb{B}}(s_0, u)$ is a proper ghost state.

We are now within the conditions of Lemma 6.4: $k_{j,x} \geq 0$ for all j (since we are in a Positive SSRI), and there is a negative run. We apply it and get that $(S'', x') = \mathsf{bshift}[(\delta_{\text{🕮}}(s_0, u), x^k)/\rho]$ is a stable cycle, and that $\mathsf{mwt}(\delta_{\mathbb{B}}(s_0, u) \xrightarrow{\alpha_{S'',x'}} S) = \infty$ (since now all reachable states jump to ∞ upon reading $\alpha_{S'',x'}$)[2]. On the other hand, $\mathsf{mwt}(\mathsf{bshift}[s/\rho] \xrightarrow{x'} S) < \infty$, since we shift on ρ. This is exactly the structure of a witness (Definition 5.1), so we conclude the claim.

We remark that the precise definition of witness is somewhat more elaborate. Thus, the full version has some additional steps that we omit here. ◀

We are now equipped to handle SSRI on all their flavours (where Stable SSRI are handled as SRI).

6.3 GSRI – A Toolbox

We proceed to develop a single tool for GSRI. Recall that now our main focus is ψ, i.e., the distance of the minimal run from the baseline ($= 0$). Also recall that in a GSRI we have $\psi(u) \geq \psi(ux) \geq \psi(uxy)$, implying that the minimal run increases. It is not hard to show that this implies $k_{1,x} \geq 0$. Further recall that the gaps in GSRI have an additional $+\mathbf{H}$ on top of their SSRI counterparts, where $\mathbf{H}$ is yet to be defined, but should be thought of as a very large constant.

The next lemma is the central property of GSRI, and is the counterpart of Lemma 6.6. Conceptually, it is what links the charge to the potential.

▶ **Lemma 6.7** (From GSRI to Stable GSRI or High Potential). *Consider a GSRI $w = uxyv$, then it is a Stable GSRI, or there is a word w' such that $\phi(w') \geq \boldsymbol{H}$.*

Proof Sketch. If $(\delta_{\text{🕮}}(s_0, u), x)$ is a stable cycle then $uxyv$ is a Stable GSRI and we are done.

If $(\delta_{\text{🕮}}(s_0, u), x)$ is not a stable cycle, then there is some minimal negative-slope run ρ on some x^k. Since $k_{1,x} \geq 0$, this cycle cannot stem from a V_1 state, so it stems from a higher (possibly ghost) state s. Using V_1 as the "low" sub-configurations, we can again apply Lemma 6.4 to obtain a cactus letter $\alpha_{S'',x'}$ that has no finite-weight transition from V_1. By the gap property of GSRI, s is more than $\mathbf{H}$ higher than V_1.

Intuitively, this can be used to show that the maximal dominant state s' after reading u is not in V_1, and is therefore at least $\mathbf{H}$ above it (the precise details are not so straightforward, but we omit them here).

We are now almost where we want: a very high dominant state. In order to make ϕ high, we need to make sure this dominant state is also high above the baseline. Fortunately, this can be done by baseline-shifting u to a word u' so that V_1 is on the baseline, rendering $\phi(u') > \mathbf{H}$. The full version contains a detailed depiction of the proof, including the elements that are glossed over in this sketch. ◀

7 Quantitative Zooming

We now come to the second challenge of showing that under certain conditions, an SSRI or GSRI exists. The technical contribution here is the *quantitative zooming technique*. We start with a rough intuition. Consider a long word w, and suppose (unreasonably) there is some infix w_2 of w where every state in S is reachable, and the gap between each two states is

[2] The diligent reader may notice that we cannot actually read $\alpha_{S'',x'}$ after u, due to a mismatch in baselines. This is handled carefully in the full version.

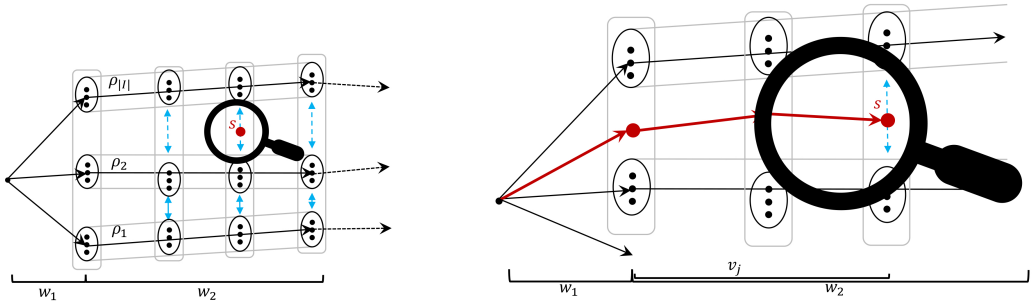

Figure 7 The state s is not near any of the independent runs. We can then zoom in on s, and construct a new independent run that reaches it.

very large. We refer to this setting as $|S|$ *independent runs*. Moreover, suppose w_2 is very long with this property. By the pigeonhole principle, at some point (i.e., after $2|S|!+1$ steps) the reachable states repeat three times in the same order. The infixes x and y between these three repetitions can almost serve as an SRI, except we also need to track for every run whether it is increasing, decreasing, or 0, in order to have $\mathsf{sign}(k_{j,x}) = \mathsf{sign}(k_{j,y})$. Adding this information, we can find an SRI in this simplest of cases. Note that this does not give any guarantees about ϕ or ψ. Also note that in this case, each V_j is just a single state.

We now ask what happens when there are only $|S|-1$ states that are far away from each other, and another reachable state s. In this case there are two options: if s is always close up to a constant to a single independent run, we say that s is *covered*. We can then track the exact distance from s to its covering run, and using the pigeonhole principle we can again find an SRI, this time with one V_j capturing s and its covering run.

The second option is that s gets far away from every independent run (depicted in Figure 7). In this case, we observe that in order to get *really* far away, s must be *somewhat* far away for a long while. We can then "zoom in" on this part of the run, and we see that now s essentially generates its own independent run. This brings us back to the $|S|$ independent runs case. Thus, we have an inductive scheme. Note that this still does not track the ϕ and ψ, which we need to add somehow. However, this is a small issue compared to the following.

A Major Quantitative Problem. We are now facing a significant problem: when zooming in we need to modify the definition of "far away", to a smaller gap. This needs to be done in a way such that at the end of the induction we indeed have sufficient gaps for an SRI. It is tempting to now say that we can define the gap arbitrarily large, and just take long enough words. However, to do so we need to allow long words in our cactus alphabet (inductively). This, in turn, increases the *weight* of nested cactus letters (longer inner word $\implies$ higher weight). But this defeats the purpose: if the weight of letters is higher, we need the gap to be even larger, creating a cyclic dependence.

One way to overcome this is to lose the effectiveness. Indeed, the approach taken in [1] assumes infinitely many words of increasing lengths and gaps. Then all these problems go away. However, this renders the proof non-constructive.

Here we carefully tailor the length function together with: the gap sizes, cover sizes, maximal weight, and the numbers required to apply the pigeonhole principle in such a way that all the quantitative arguments fall into place. Crucially, all of these functions are now parametrised not just by the depth of the word, but also by the number of *independent runs*, to facilitate the induction.

7.1 Recursive Functions

In order to reason about the various properties required for zooming, we introduce two families of recursive functions. We bring here their intuitive description and dependencies, without the explicit definitions. The parameters d and i represent the *depth* of a word and the number of *independent runs*, respectively. The inductions are always decreasing with d (smaller depth is simpler) and increasing with i (more independent runs is simpler).

- $\texttt{Len}(d,i)$: the maximal length we allow for a word w in a cactus letter $\alpha_{S',w}$ of depth d with i independent runs. Depends on: $\texttt{Len}(d,i+1)$, $\texttt{Typ}(d,i)$, and $\texttt{Amp}(d,i)$
- $\texttt{Cov}(d,i)$: the maximal distance allowed between a run and its closest independent run (i.e., the *cover*). Depends on: $\texttt{MaxWt}(d-1)$, $\texttt{Len}(d,i+1)$ and $\texttt{Cov}(d,i+1)$.
- $\texttt{MaxWt}(d)$: the maximal weight of a letter of depth d (does not depend on i). Depends on: $\texttt{MaxWt}(d-1)$ and $\texttt{Len}(d,1)$ (note that 1 is the "worst" i).
- $\texttt{Amp}(d,i)$: A lower bound on what is considered "high values" of ϕ, dubbed *amplitude*. Depends on: $\texttt{MaxWt}(d-1)$, $\texttt{Typ}(d,i)$, and $\texttt{Len}(d,i+1)$.
- $\texttt{Typ}(d,i)$: An upper bound on the number of configurations representable with i runs, such that all other runs are within $\texttt{Cov}(d,i)$ from an independent run. These are the "pigeonholes". Depends on $\texttt{Cov}(d,i)$.

We can now define some well-awaited terms: the function $L_{\mathsf{simp}}(d) = \texttt{Len}(d,1)$ for all d, and the constant $\mathbf{H} = \texttt{Amp}(|S|,0)$.

The functions above pertain to ϕ. A similar set of functions (prefixed by $\texttt{G}$ for "general") pertains to ψ. Their definitions, however, depend also on $\mathbf{H}$. In particular, we can define $L_{\mathsf{gen}}(d) = \texttt{GLen}(d,1)$.

7.2 From Increasing ϕ to SSRI

Consider a word $w = w_1 w_2$ with $\phi(w_1) \leq \phi(w_2)$, and such that along w_2 we identify a set I of runs that maintain some gaps between them. Denote $i = |I|$ and $d = \texttt{dep}(w_2)$. Assume that w_2 is fairly long, specifically, $|w_2| = \beta \cdot \texttt{Len}(d+1,i)$ for some small fixed $0 < \beta < 1$. We now distinguish between two cases: the case where $\phi(w_1 w_2) > \phi(w_1) + \texttt{Amp}(d+1,i)$ (*high-amplitude*), and when $\phi(w_1 z)$ remains within $\pm\texttt{Amp}(d+1,i)$ for every prefix z of w_2 (*bounded amplitude*). These do not cover all possible cases, but they suffice.

We then proceed to define two types of *decompositions*. In the high-amplitude case, we divide the word to many infixes, such that ϕ strictly increases between each two infixes, and never exceeds ϕ of the next segment. In addition, we require that these segments are not too short (at least $\texttt{Len}(d,1)$), and that there are many of them – so many that $\texttt{Typ}(d,i)$ can serve as pigeonholes for a certain argument (see Figure 8).

In the bounded-amplitude case we also show a decomposition, still requiring long segments, and that the potential is equal after each segment. We remark that showing these decompositions is technically challenging; the full version contains the precise statements and proofs, which account for much in the recursive definitions.

Next, we call a decomposition a *cover*, if after each segment, all states are within $\texttt{Cov}(d+1,i)$ from some independent run. Crucially, the number of segments in the decomposition is chosen so that if it is also a cover, then we can apply the pigeonhole principle and find 3 repetitions of a configurations, using which we can obtain an SSRI.

With these definitions in place, we can apply the zooming technique described above: start with a single run in I, and decompose the word. If its decomposition is a cover, we find an SSRI and we are done. Otherwise, there is some state that goes outside the cover. Zooming in on it and the run that leads to it, we can induce a new independent run. An important point is that the carefully-chosen lengths, gaps and amplitudes actually yield all the correct assumptions for $i+1$, and we can proceed.

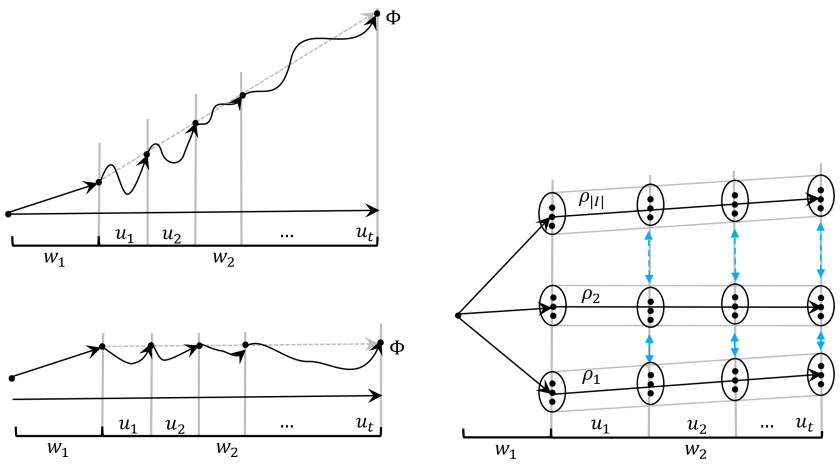

Figure 8 In a ϕ-increasing decomposition (top left), the potential gradually increases, and within each segment, it never surpasses the potential at its end. In a ϕ-bounded decomposition (bottom left), the potential keeps its value at the end of the segments, and certain points within the segments never surpass it. In a cover decomposition (right), the states within the sub-configurations at the end of each segment are close to one of the independent runs.

A caveat to the above is that while $\mathsf{dep}(w_2) = d$, it may be the case that the infix u we zoom on has $\mathsf{dep}(u) < d$. This is a problem since u is now too *long* to serve as an effective cactus letter. To this end, we tailor the functions even further so that in such cases we can use induction with $d-1$ rather than $i+1$.

Proceeding with the induction up to depth $|S|$, we establish the following.

▶ **Lemma 7.1.** *Consider a word w_1w_2 with $\phi(w_1) \leq \phi(w_1w_2)$ and $|w_2| = \beta \cdot \mathtt{Len}(|S|, 1)$, then $w_1w_2 = uxyv$ where $uxyv$ is an SSRI.*

7.3 From Decreasing ψ to GSRI

Using similar techniques to Section 7.2, we define decompositions and cover of ψ-decreasing words. Here, however, we cannot assume that ψ has bounded growth, so Lemma 4.1 looms over us, whose consequences (if it applies) is a high-potential word.

We condition on ψ having bounded growth, and choose a constant depending on $\mathbf{H}$ for this bound. This allows us on the one hand to obtain a GSRI, by relying on the definition of the General recursive functions, which account for this $\mathbf{H}$ factor, and on the other hand if the conditioning fails, we obtain a word with high enough ϕ amplitude. We show the following.

▶ **Lemma 7.2.** *Either there is a word $w \in \Upsilon^*_{\mathsf{simp}}$ with $\phi(w) > \boldsymbol{H}$, or for every word $w_1w_2 \in \Upsilon^*_{\mathsf{gen}}$ with $\psi(w_1) \geq \psi(w_1w_2)$ and $|w_2| = \beta \cdot \mathtt{GLen}(|S|, 1)$, we have $w_1w_2 = uxyv$ where $uxyv$ is a GSRI.*

8 Outline of the Proof

We are finally ready to put all the components together. The proof outline is depicted in Figure 9.

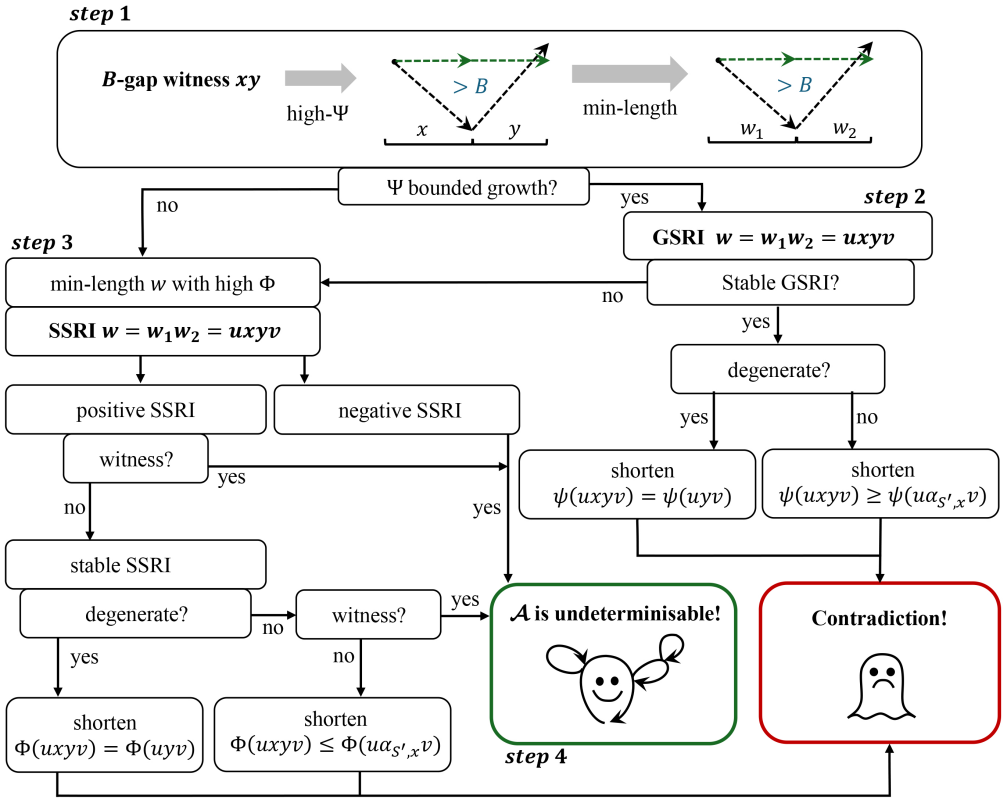

Figure 9 A summary of our proof flow.

The precise main result is the following.

▶ **Theorem 8.1.** *Consider a WFA $\mathcal{A}$, then $\mathcal{A}$ is undeterminisable if and only if there is a B-gap witness with $B = \texttt{GAmp}(|S|, 0)$ where $|S|$ is the size of $\widehat{\mathcal{A}}$.*

Note that the direction $\mathcal{A}$ is undeterminisable $\implies$ B-gap witness is trivial from Theorem 2.2. We handle the converse in several steps.

Proof.

Step 1: From a Gap witness to a ψ-Decreasing Word. Consider a B-gap witness xy for $B = \texttt{GAmp}(|S|, 0)$ as per Theorem 2.2. By choosing the minimal run on xy as baseline, we have another run that goes far below it on x, yielding $\psi(x) > B$ but $\psi(xy) = 0$. That is, a significant decrease in ψ upon reading y.

Step 2: From ψ-Decreasing to ϕ-Increasing. Thus far, xy do not have cactus letters. Let w_1w_2 be a *minimal length* word over $\Upsilon^*_{\mathsf{gen}}$ such that $\psi(w_1) - \psi(w_1w_2) > \texttt{GAmp}(|S|, 0)$. Note that we now allow cactus letters, and also require length minimality. There exists such a word because xy satisfies this condition. By the construction of Υ_{gen}, we have $\mathsf{dep}(w_1w_2) \leq |S| - 1$.

We now ask whether there is a word $w' \in \Upsilon^*_{\mathsf{simp}}$ such that $\phi(w') > \mathbf{H}$. If there is, we proceed to Step 3. Otherwise, we assume by way of contradiction that there is no such word.

By Lemma 4.1, this assumption tells us that the ψ has bounded decrease over Υ_{gen}. Since w_1w_2 has a huge charge drop, then w_2 must be very long. We use this with Lemma 7.2 to show that w_1w_2 is a GSRI. Then, by Lemma 6.7 and the assumption that there is no word with $\phi(w') > \mathbf{H}$, we get that this GSRI is stable. In particular, it is a Stable SRI $w_1w_2 = uxyv$. Then, we have the following:

- If it is a Degenerate SRI then by Proposition 6.2 we can shorten w_1w_2 while maintaining ψ, contradicting the minimality.
- If it is Nondegenerate, then by Lemma 6.3 we can shorten it to $u\alpha_{S',x}v$ (which is indeed shorter, since xy is replaced with a single letter), while decreasing ψ on the suffix. This is again a contradiction to the minimality of w_1w_2.

Thus, either way we reach a contradiction. So there must exist a word $w' \in \Upsilon^*_{\mathsf{simp}}$ with $\phi(w') > \mathbf{H}$.

Step 3: Increasing Potential to Witness. Intuitively, we now repeat Step 2 with ϕ and SSRI instead of ψ and GSRI, and this results in a witness, as follows.

By the existence of w', let $w \in \Upsilon^*_{\mathsf{simp}}$ be the shortest word for which $\phi(w) > \mathbf{H}$. Note that we again require minimality, now over Υ_{simp}. By the bounded growth of ϕ (which is unconditional, see Section 4) we can show that w is long enough to apply Section 7.2. We can therefore write $w_1w_2 = uxyv$ where $uxyv$ is an SSRI.

The case split now is somewhat more intricate than in GSRI. In each case, if we reach a witness then we proceed to Step 4.

- If $uxyv$ is a Negative SSRI, then by Lemma 6.5 there is a witness.
- If $uxyv$ is not a Negative SSRI, then it is a Positive SRI. By Lemma 6.6, either there is a witness or $uxyv$ is a Stable SSRI, and we proceed to the next item.
- If $uxyv$ is a Stable SSRI, we split to cases again.
  - If $uxyv$ is a Degenerate SSRI, then by Proposition 6.2 we have $\phi(uxyv) = \phi(uyv)$, contradicting the minimality of w.
  - If $uxyv$ is a Non-degenerate SSRI, then by Lemma 6.3, either there is a witness or $\phi(uxyv) \leq \phi(u\alpha_{S',x}v)$, again contradicting the minimality of w.

We conclude that in all cases we either reach a contradiction, or we have a witness.

Step 4: Witness to Undeterminisability. Recall that if there is a witness, then $\mathcal{A}$ is undeterminisable. This concludes the proof of Theorem 8.1. ◀

8.1 Algorithm and Complexity

The proof of the gap characterisation of Theorem 2.2 is constructive [9, 1], in the following sense:

▶ **Theorem 8.2.** *Consider a WFA $\mathcal{A}$ and $B \in \mathbb{N}$. We can construct a deterministic WFA $\mathcal{A}|_B$ such that $\mathcal{A}$ is equivalent to $\mathcal{A}|_B$ if and only if $\mathcal{A}$ has B-bounded gaps. Moreover, $\mathcal{A}|_B$ can be computed in time $B^{O(|\mathcal{A}|)}$.*

A determinisability algorithm is now simple: given a WFA $\mathcal{A}$, compute $B = \texttt{GAmp}(|S|, 0)$, construct $\mathcal{A}|_B$, and check whether $\mathcal{A}$ is equivalent to $\mathcal{A}|_B$ (which is decidable by [2]). If it is, it is determinisable, and $\mathcal{A}|_B$ is a deterministic equivalent. Otherwise, it is not determinisable by Theorem 8.1.

The complexity of this algorithm is dominated by the computation of B. In the full version, we analyse the recursive structure of our functions, placing it in the 6th level of the fast-growing hierarchy [10, 22, 24], i.e., non-elementary (elementary is the 3rd level of this hierarchy), but primitive recursive (well below the Ackermann function). We do not bring the ideas of the analysis here. In broad terms, the bound is obtained by formulating upper bounds for our functions in terms that are easy to manage using standard operations of the fast-growing functions, namely substitutions and bounded primitive recursion.

9 Discussion and Future Research

The decidability of determinisation of WFAs has been out of reach for over 30 years until recently, when it was shown to be decidable. This raises the hope that this problem might not be as intractable as its open status may have suggested.

In this paper, we make the first step towards establishing the complexity of this problem, by showing that it is primitive recursive. This is particularly good news, as it is already far below some problems in related counter-based models, specifically reachability in Vector Addition Systems and Universality of One Counter Nets, which are both Ackermann-complete [5, 17, 13].

A natural question is what machinery is needed to improve the complexity. To this end, our work restores the faith in the original approach: improve the bound on the maximal gap B, and get a better algorithm. On the negative side, our approach seems limited for further significant improvements. Indeed, the presence of cactus letters quickly leads to recursive Ramsey-style arguments, which lead to a tower of exponents. A possible direction would be to use only constant-depth cactus letters. This, however, is already dangerously close to working with the original alphabet, and approach that was unsuccessful for decades.

Going beyond determinisability, our work introduces simpler and more accessible tools for analysing the run-tree of a WFA (namely Zooming and SRI). We believe these can be reused and adapted for other counter-based methods, in particular certain open problems for One-Counter Nets and One-Counter Automata.

References

1 Shaull Almagor, Guy Arbel, and Sarai Sheinvald. Determinization of min-plus weighted automata is decidable. In Kasper Green Larsen and Barna Saha, editors, *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, Vancouver, BC, Canada, January 11-14, 2026*, pages 247–257. SIAM, SIAM, 2026. doi:10.1137/1.9781611978971.11.

2 Shaull Almagor, Udi Boker, and Orna Kupferman. What's decidable about weighted automata? *Inf. Comput.*, 282:104651, 2022. Special issue on 9th International Workshop Weighted Automata: Theory and Applications (WATA 2018). doi:10.1016/j.ic.2020.104651.

3 Krishnendu Chatterjee, Laurent Doyen, and Thomas A Henzinger. Quantitative languages. *ACM Transactions on Computational Logic (TOCL)*, 11(4):1–38, 2010. doi:10.1145/1805950.1805953.

4 Agnishom Chattopadhyay, Filip Mazowiecki, Anca Muscholl, and Cristian Riveros. Pumping lemmas for weighted automata. *Logical Methods in Computer Science*, 17, 2021. doi:10.46298/lmcs-17(3:7)2021.

5 Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is ackermann-complete. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1229–1240. IEEE, IEEE, 2021. doi:10.1109/FOCS52979.2021.00120.

6 Laure Daviaud. Register complexity and determinisation of max-plus automata. *ACM SIGLOG News*, 7(2):4–14, 2020. doi:10.1145/3397619.3397621.

7 Laure Daviaud and David Purser. The big-o problem for max-plus automata is decidable (PSPACE-complete). In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2023. doi:10.1109/LICS56636.2023.10175798.

8 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.

9 Emmanuel Filiot, Ismaël Jecker, Nathan Lhote, Guillermo A Pérez, and Jean-François Raskin. On delay and regret determinization of max-plus automata. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE, 2017. doi:10.1109/LICS.2017.8005096.

10 Andrzej Grzegorczyk. Some classes of recursive functions, 1953.

11 K. Hashiguchi. Limitedness theorem on finite automata with distance functions. *Journal of computer and system sciences*, 24(2):233–244, 1982. doi:10.1016/0022-0000(82)90051-4.

12 K. Hashiguchi. New upper bounds to the limitedness of distance automata. *Theoretical Computer Science*, 233(1-2):19–32, 2000. doi:10.1016/S0304-3975(97)00260-0.

13 Piotr Hofman and Patrick Totzke. Trace inclusion for one-counter nets revisited. *Theoretical Computer Science*, 735:50–63, 2018. doi:10.1016/j.tcs.2017.05.009.

14 Daniel Kirsten. Distance desert automata and the star height problem. *RAIRO-Theoretical Informatics and Applications*, 39(3):455–509, 2005. doi:10.1051/ita:2005027.

15 Daniel Kirsten and Sylvain Lombardy. Deciding unambiguity and sequentiality of polynomially ambiguous min-plus automata. In *26th International Symposium on Theoretical Aspects of Computer Science STACS 2009*, pages 589–600. IBFI Schloss Dagstuhl, 2009. doi:10.4230/LIPIcs.STACS.2009.1850.

16 Ines Klimann, Sylvain Lombardy, Jean Mairesse, and Christophe Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theoretical Computer Science*, 327(3):349–373, 2004. doi:10.1016/j.tcs.2004.02.049.

17 Jérôme Leroux. The reachability problem for petri nets is not primitive recursive. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1241–1252. IEEE, IEEE, 2021. doi:10.1109/FOCS52979.2021.00121.

18 H. Leung and V. Podolskiy. The limitedness problem on distance automata: Hashiguchi's method revisited. *Theoretical Computer Science*, 310(1-3):147–158, 2004. doi:10.1016/S0304-3975(03)00377-3.

19 Sylvain Lombardy and Jacques Sakarovitch. Sequential? *Theoretical Computer Science*, 356(1-2):224–244, 2006. doi:10.1016/j.tcs.2006.01.028.

20 Mehryar Mohri. Compact representations by finite-state transducers. In *32nd Annual Meeting of the Association for Computational Linguistics*, pages 204–209, 1994. doi:10.3115/981732.981760.

21 Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311, 1997.

22 Sylvain Schmitz. Complexity hierarchies beyond elementary. *ACM Transactions on Computation Theory (TOCT)*, 8(1):1–36, 2016. doi:10.1145/2858784.

23 Marcel Paul Schützenberger. On the definition of a family of automata. *Inf. Control.*, 4(2-3):245–270, 1961. doi:10.1016/S0019-9958(61)80020-X.

24 George Tourlakis. *Computability*. Springer, 2022. doi:10.1007/978-3-030-83202-5.