A Factorization Theorem for Forest Algebras

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

Michaël Cadilhac
DePaul University, Chicago, IL, USA

Asaf Shoham
Department of Computer Science, Technion, Haifa, Israel

Abstract

Simon's factorization theorem is a celebrated tool in algebraic automata theory, providing bounded-depth decompositions of words with respect to morphisms into finite semigroups.

We develop an analogue of Simon's theorem for *forests* in the setting of forest algebras. In contrast with words, this presents a basic difficulty: recursively factoring a forest requires keeping track of where each subforest "fits". This difficulty ripples throughout the proof, and we overcome it by augmenting the free forest algebra and by developing a framework that supports recursive factorization of forests, along with its semantic implications.

Our main result identifies a new semantic restriction on morphisms (called $\mathcal{R}$-alignment) which intuitively ensures that different ways of cutting a forest remain compatible (in a certain sense) at the semigroup level. Under this condition, we prove that every morphism admits decompositions of bounded depth. We also prove that without this restriction, there are morphisms for which no bounded-depth decomposition exists (under our notion of decomposition).

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

Keywords and phrases Factorization Forest, Semigroup, Forest Algebra, Green's relations, Tree Languages

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2026.9

Related Version *Full Version*: https://arxiv.org/abs/2605.10368

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

1 Introduction

Regular languages can be studied through automata, logic, or algebra, and each of these viewpoints comes with its own structural tools. In the algebraic study of word languages, one of the most influential such tools is Simon's factorization forest theorem [8, 7]. If one fixes a morphism $\eta\colon A^+ \to S$ into a finite semigroup, then η does not merely assign to each word an element of S; it also constrains how words can be recursively factored. Informally, Simon's theorem says that every word admits a rooted decomposition tree of depth bounded solely in terms of the target semigroup. The leaves are letters, the internal nodes correspond to concatenations of consecutive factors, and the high-arity nodes arise when all children have the same idempotent image under η. In this way, algebraic recognition yields not only a classification of words by their images, but also a bounded-depth structural decomposition of every input.

The purpose of this paper is to obtain an analogous bounded-depth decomposition theorem for *forests*. At a broad level, the intended parallel is clear: one would like a morphism recognizing a forest language to impose on every forest a decomposition whose depth is controlled only by the finite target algebra. What makes this difficult is that the very notion of factorization changes in the branching setting. For words, one splits into

consecutive factors, and each factor is again a word, so the recursive process remains in the same class of objects. For forests, by contrast, removing a subtree leaves behind, in the standard terminology, a *context*: a forest with a distinguished hole indicating where the missing subtree is to be reinserted. Once one continues decomposing, several previously removed subforests may have to be remembered at once. This bookkeeping problem has no counterpart in the word case, and it is the first obstruction to a direct forest version of Simon's theorem.

The natural algebraic framework for this problem is that of *forest algebras*, introduced by Bojańczyk and Walukiewicz [4]. A forest algebra is a pair (H, V) of monoids, usually called the *horizontal* and *vertical* monoids. The horizontal monoid H captures forest types, the vertical monoid V captures context types, and V acts on H by substitution (see Section 3). In the *free* forest algebra over an alphabet, this picture becomes completely concrete: the elements of H are forests, the elements of V are contexts, and the action is exactly the operation of plugging a forest into the hole of a context. Forest algebras have proved to be a natural algebraic formalism for regular languages of unranked forests and trees, and they have already supported substantial interactions with logic and algorithms on trees [3, 2].

Our first contribution is a formalism that makes recursive decomposition possible in this setting. We introduce an augmented free forest algebra in which, besides the ordinary hole of a context, one may use *default holes* $\square_h$. Intuitively, $\square_h$ marks a specific subforest that has already been extracted and is to be treated as fixed while the rest of the decomposition continues. These markers let one continue decomposing the residue object without losing track of how previously removed pieces fit back into the whole forest, and they provide the bookkeeping device that is missing from the naive forest analogue of factorization forests.

On top of this augmented algebra, we first define *binary decompositions*. Their role is not simply to encode binary splitting, but to recover certain monotonicity properties that the classical proof for words gets "for free". Specifically, for a word w, if P_w denotes the set of images under η of the prefixes of w, then the implication "u prefix of $w \implies P_u \subseteq P_w$" is immediate from properties of concatenation and morphisms. In the forest setting, the relevant quantity is subtler, and only holds under certain conditions, which we therefore first ensure (see Section 4.3).

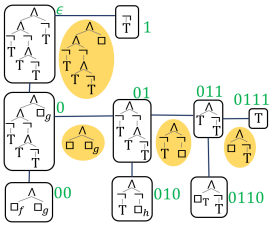
Binary decompositions are therefore fundamental in order to recover a notion of decomposition that, in the setting of words, comes for free by the definition of concatenation. Equipped with these, we then introduce *general decompositions*, the forest analog of Simon's decomposition trees. The nodes in a general decomposition may be leaves, binary nodes, or idempotent nodes. An idempotent ($\mathfrak{I}$) node compresses an entire region that already admits a binary decomposition all of whose relevant contexts map to a single idempotent of the target semigroup; this is the forest analogue of the high-arity idempotent node in Simon's theorem. For technical convenience, and to obtain better bounds, we also introduce a third type of inner nodes: centipede ($\mathfrak{C}$) nodes.

▶ Remark 1.1 (Visual representation of forests v.s. decomposition trees). In order to visually separate forests (elements in a forest algebra, including trees) from decomposition trees (the tree-structures we define), we depict forests with diagonal downward edges, and trees with right and down edges (as depicted in e.g., Figure 1).

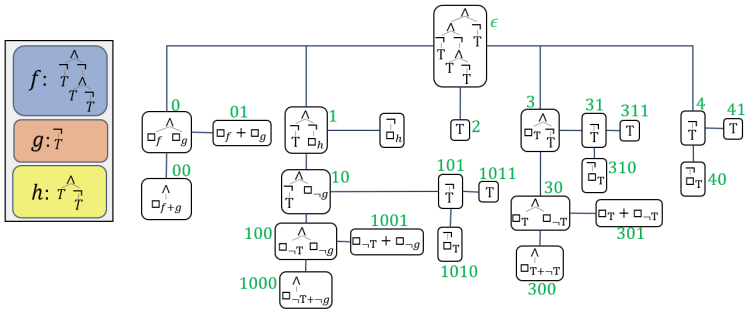
▶ **Example 1.2** (TBF)**.** We demonstrate the setting with a concrete example. We refer back to this example as the concepts are clarified throughout the paper. Consider the language of True Boolean Formulas (TBF), which consists of binary trees labeled by $A = \{T, \neg, \wedge\}$ that represent a formula evaluating to true.

This language can be formulated using forest algebras. For example, $\neg(\wedge(T + \neg(T))$ is a forest (in fact, a tree) in the free forest algebra $\langle H_A, V_A \rangle$ over A. It represents a formula that evaluates to true. We can write this tree as a composition of the *context* $\neg(\wedge(\Box))$ with the forest $T + \neg(T)$ (note that $+$ is the horizontal operator, and represents putting two forests "side by side").

For a more elaborate forest, see e.g., the node labeled ϵ in Figure 1a. There we also illustrate a binary decomposition of the forest: in each node we "pluck" a sub-forest to the right child, and put the "residue" in the bottom child. The contexts used in the decomposition are illustrated in the yellow ovals. In order to keep track of plucked sub-forests, we use default holes. For example, node 00 has default holes corresponding to both node 01 ($\Box_f$) and to node 1 ($\Box_g$). In Figure 1b we illustrate a general decomposition, starting with an $\mathfrak{I}$-node at the root. We revisit this in due course.



(a) A partial binary decomposition. For TBF, this is a decomposition corresponding to the $\mathfrak{I}$-node in Figure 1b.



(b) A general decomposition for TBF. The node at ϵ is an $\mathfrak{I}$-node.

Figure 1 Decompositions for TBF. f, g, h are given by the legend.

With this machinery in place, we identify a semantic condition on morphisms, called $\mathcal{R}$*-alignment*. Its definition appears in Section 6.1, but its role can be described informally as follows. When the same forest can be obtained by cutting in two different ways, one obtains two residue contexts. If their images lie in the same $\mathcal{J}$-class, $\mathcal{R}$-alignment requires these images to remain compatible in a stronger one-sided sense. In the word setting, the corresponding coherence is largely built into the linear order of factors; in the forest setting, it has to be identified explicitly as a property of the recognizing morphism.

Our main theorem shows that this condition is sufficient for bounded-depth decomposition.

▶ **Theorem** (informal). *Let φ be a forest morphism into a finite forest algebra (H, V), and assume that the V-component of φ is $\mathcal{R}$-aligned. Then every forest admits a general decomposition of depth at most $4|V| - 3$.*

We also show that, without $\mathcal{R}$-alignment, the notion of decomposition developed here does not admit any uniform depth bound in general. Thus the restriction is not merely a byproduct of the proof technique. The proof follows the same broad lines as semigroup-theoretic proofs of Simon's theorem, in that it is organized around the $\mathcal{J}$-order on the target (vertical) semigroup [8, 7]. More precisely, we establish bounded decomposition theorems for three building-block cases – groups, simple or $\mathbb{0}$-simple semigroups, and null semigroups – and then combine them by an induction that climbs up the hierarchy of $\mathcal{J}$-classes via a $\mathcal{J}$-minimal nonzero element, the ideal it generates, and the corresponding Rees quotient. Among these ingredients, the simple or $\mathbb{0}$-simple semigroup case is technically the most involved, and it is also the only point at which $\mathcal{R}$-alignment is used in an essential way; the group and null cases are treated separately and feed into the later induction.

We view this theorem as a significant foundational step toward a decomposition theory for forests, parallel to the role that factorization forests have long played for words. Such a theory is interesting already as a structural tool for inductive arguments over forest morphisms, but it also suggests algorithmic directions, since bounded-depth decompositions are natural candidates for hierarchical summaries of large tree-shaped inputs [2]. We do not pursue those applications here in full, but the framework developed in this paper is designed with that broader perspective in mind.

The extended abstract provides an intuitive overview of the main ideas, and is organized as follows. We begin by recalling forests, contexts, and forest algebras, and by introducing the augmented free forest algebra with default holes. We then define binary decompositions and show, using references and rotations, how to recover the monotonicity properties they need. After that we introduce general decompositions, which are the bounded-depth objects used in the theorem. In the second half of the paper, we define $\mathcal{R}$-alignment, establish the group and simple semigroup decomposition theorems, and combine them into the main result by induction on the target semigroup. We conclude with the reduction lemma and with a lower bound showing that without $\mathcal{R}$-alignment no bounded-depth theorem can hold in general. Due to space constraints, the technical details appear in the full version.

2 Preliminaries

Let $\mathbb{N}$ be the non-negative integers. For a set A let A^* and A^+ be the sets of finite words and finite non-empty words over A. Denote by ϵ the empty word. For $x, y \in A^*$ we write $x \leq y$ if x is a prefix of y. A *tree* is a prefix-closed subset of $\mathbb{N}^*$. A *forest* is a prefix-closed subset of $\mathbb{N}^+$. Note that trees are rooted at ϵ, whereas forests have roots in $\mathbb{N}$. For nodes $x \leq y$ in a forest f, we say that y is a *descendant* of x. If $y = x \cdot i$ for some $i \in \mathbb{N}$, then y is a *child* of x. The *depth* of a forest f is $\max\{d \geq 0 : \mathbb{N}^d \cap f \neq \emptyset\}$. A *labeled forest over alphabet* A is a forest f equipped with a labeling function $\mathsf{lab}\colon f \to A$. We overload notation and write $f[x]$ as $\mathsf{lab}(x)$ for $x \in f$. We say that a labeled forest g *extends* forest f if $f \subseteq g$ as forests, and $f[x] = g[x]$ for all $x \in f \cap g$.

Semigroups and Green's Relations. A *semigroup* is a structure $\langle S, \cdot \rangle$ such that S is a set and $\cdot\colon S \times S \to S$ is an associative operation. We often write uv instead of $u \cdot v$ for $u, v \in S$. An element $e \in S$ is *idempotent* if $e^2 = e$. An element z is a *zero*, denoted $\mathbb{0}$ if $zx = xz = z$ for all $x \in S$. An element u is a *unit*, denoted $\mathbb{1}$, if $ux = xu = x$ for all $x \in S$. We denote by S^1 the *monoid* obtained by adding a unit to S, if it does not have one.

Consider two elements $u, v \in S$. Green's relations are defined as follows.

- $u \leq_{\mathcal{L}} v$ if $u \in S^1 v$.
- $u \leq_{\mathcal{J}} v$ if $u \in S^1 v S^1$.
- $u \leq_{\mathcal{R}} v$ if $u \in v S^1$.
- $u \leq_{\mathcal{H}} v$ if $u \leq_{\mathcal{L}} v$ and $u \leq_{\mathcal{R}} v$.

Every Green relation $G \in \{\mathcal{J}, \mathcal{L}, \mathcal{R}, \mathcal{H}\}$ induces an equivalence relation by $uGv \iff u \leq_G v \wedge v \leq_G u$, and we denote by $[u]_G$ the equivalence class of u, dubbed the *G-class of u*.

The semigroup S is a *simple* (resp. *$\mathbb{0}$-simple*) if it has only one $\mathcal{J}$-class (resp., only $\{\mathbb{0}\}$ and $S \setminus \{\mathbb{0}\}$ are the $\mathcal{J}$-classes). In the full version we present several results about Green's relations and simple semigroups, used throughout the proof. These are omitted here as we do not reach this level of detail.

Forest Algebras. Glossing over some technical details, a forest algebra is a tuple $\langle H, V, +, \cdot \rangle$ where $\langle H, + \rangle$ and $\langle V, \cdot \rangle$ are monoids, along with an action of V on H (which we denote as $\cdot\colon V \times H \to H$ or omit) such that for every $u, v \in V$ and $h \in H$ we have $(u \cdot v) \cdot h = u \cdot (v \cdot h)$. A *forest morphism* from $\langle H_1, V_1 \rangle$ to $\langle H_2, V_2 \rangle$ is a pair $\langle \eta, \varphi \rangle$ where $\eta\colon H_1 \to H_2$ and $\varphi\colon V_1 \to V_2$, with $\eta(v \cdot h) = \varphi(v) \cdot \eta(h)$ for every $v \in V, h \in H$.

The *Free Forest Algebra* over alphabet A is a concrete forest algebra $\langle H_A, V_A\rangle$ where H_A is the set of A-labeled forests, and V_A is the set of $A \cup \{\Box\}$-labeled forests, where $\Box$ appears exactly in one leaf. The action of V_A on H_A is the natural composition where we replace $\Box$ with the given subforest. The full version is devoted to establishing a convenient formalism to capture and reason about free forest algebras. These formalisms enable our precise proofs, but encumber notations, hence our informality here.

▶ **Example 2.1** (TBF as a Forest Algebra)**.** We formulate TBF from Example 1.2 as a forest morphism $\langle H_A, V_A\rangle \to \langle H, V\rangle$ for $A = \{\wedge, \neg, T\}$. We start with describing $\langle H, V\rangle$. The forests are $H = \{\epsilon, \triangle, \blacktriangle, \blacktriangle\!\!\triangle, \triangle\!\!\triangle, \bot\}$ with the following intuition: ϵ captures the empty forest, and $\bot$ captures all the "hopelessly invalid" forests (e.g., forests that are malformed, ternary nodes, etc.). $\triangle$ and $\blacktriangle$ capture, respectively, trees whose root evaluates to true and false, respectively. $\triangle\!\!\triangle$ and $\blacktriangle\!\!\triangle$ capture forests of the form $t_1 + t_2$ where, respectively, both t_1, t_2 evaluate to true and where at least one evaluates to false. Note that such forests are then plugged into a $\wedge$ node, and we therefore need a way of tracking them, hence the latter two values.

To describe V, we identify it with a subset of H^H, so that each element describes its action on H. Since this set is elaborate, we only demonstrate a few important elements in V. Note that each context "expects" a certain structure of its input element, specifically whether it should have one or two roots. For example, the elements $f_\triangle, f_\blacktriangle, f_\wedge, f_\neg$ intuitively capture contexts such as $\Box + T$, $\Box + \neg(T)$, $\wedge(\Box)$, and $\neg(\Box)$ respectively. Thus, they are given by the following functions (undefined inputs map to $\bot$):

$$f_\triangle(h) = \begin{cases} \triangle & \epsilon \\ \blacktriangle\!\!\triangle & \blacktriangle \\ \triangle\!\!\triangle & \triangle \end{cases} \quad f_\blacktriangle(h) = \begin{cases} \blacktriangle & \epsilon \\ \blacktriangle\!\!\triangle & \triangle \text{ or } \blacktriangle \end{cases} \quad f_\wedge(h) = \begin{cases} \blacktriangle & \blacktriangle\!\!\triangle \\ \triangle & \triangle\!\!\triangle \end{cases} \quad f_\neg(h) = \begin{cases} \triangle & \blacktriangle \\ \blacktriangle & \triangle \end{cases}$$

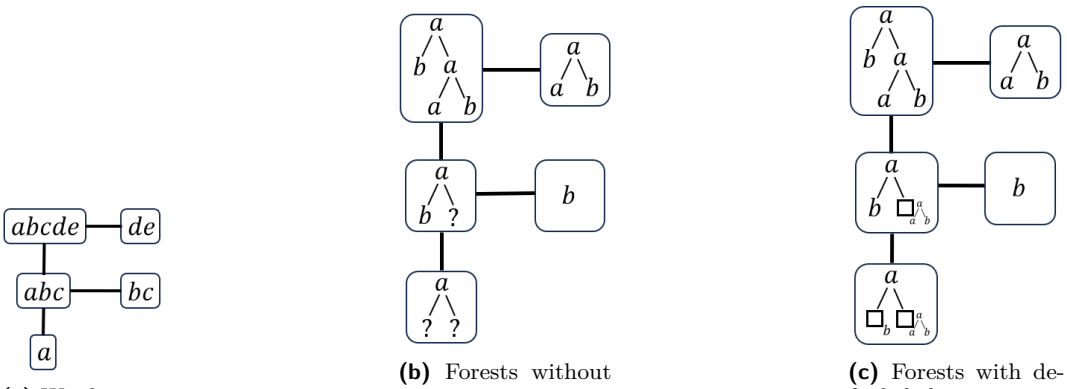
Compositions then give rise to other elements, e.g., $f_\wedge(f_\blacktriangle)$ corresponds to e.g., the context $\wedge(\Box + \neg(T))$, and is defined by $f_\wedge(f_\blacktriangle(h)) = \blacktriangle$ for $h \in \{\triangle, \blacktriangle\}$ and $\bot$ otherwise. Observe that this is an *idempotent* element in V, i.e., $f_\wedge(f_\blacktriangle(f_\wedge(f_\blacktriangle))) = f_\wedge(f_\blacktriangle)$.

The morphisms η, φ are then naturally obtained by assigning each context/forest to the element it represents. Observe that all the contexts (in yellow) in Figure 1a map to $f_\wedge(f_\blacktriangle)$.

3 Augmented Forest Algebras

In the setting of words there is a natural notion of decomposition: a word w can be split into $w = u \cdot v$, where both u and v are subwords (Figure 2a). Then, one can recursively continue to decompose. This decomposition supports the *local* nature of decompositions – in order to evaluate the morphism on an infix, one only needs to look at the corresponding nodes (indeed, this is the basis for the *fast infix evaluation* algorithm [1]).

For forests, a fundamental problem arises: when decomposing a forest h, we want to "pluck" a subtree rooted at a certain node x. This leaves us with a *context*, rather than a forest, so that we know where to restore the plucked subtree. However, if we now wish to continue decomposing, we may want to pluck another subtree from the remaining context. This results in two distinct "holes" to restore the subtrees (Figure 2b), and this would render the decomposition ill-defined, since it is not clear where to plug in each subforest – this is the main reason contexts have only one hole.



(a) Words.

(b) Forests without augmentation.

(c) Forests with default holes.

Figure 2 For words (Figure 2a), each node can be evaluated separately according to the morphism. In particular, the value of a node is uniquely determined by the values of its children.
For forests (Figure 2b), it is not even clear how to represent the residue forest after "plucking" a subforest. For example, the bottom leaf cannot be evaluated, and looking at it locally does not provide enough information to know how the two right leaves plug into it.
To overcome this, we use default holes (Figure 2c).

In order to overcome this problem, we introduce the *augmented free forest algebra*. Intuitively, we add a new symbol $\Box_h$ for every $h \in H_A$, called a "default hole": these are holes that can only be filled with the subforest h (Figure 2c). A priori, it may seem as though this achieves nothing – we just write $\Box_h$ instead of h. Nonetheless, this is the basic building block for our notion of decomposition: the elements $\Box_h$ are used as "markers" within a forest (or context) to denote a part which is *elementary* and cannot be factored further. We make this intuition concrete when defining decompositions in Section 4.

Technically, we augment the free forest algebra over alphabet A with *default holes*: let $\Box_{H_A} \coloneqq \{\Box_h : h \in H_A\}$, then we replace A with the augmented infinite alphabet $\widetilde{A} = A \cup \Box_{H_A}$, and consider the free forest algebra over $\widetilde{A}$, denoted $\langle \widetilde{H_A}, \widetilde{V_A} \rangle$, with one important restriction: the elements $\Box_h$ may appear *only in the leaves* of a forest or context (as intuitively they represent a suffix subforest, and do not have further children). In Example 1.2 above we demonstrate such default holes.

Default holes raise a problem: a morphism $\varphi \colon \langle H_A, V_A \rangle \to \langle H, V \rangle$ is not defined on default holes. However, a natural way to retain the semantics of a morphism is to *unravel* a default hole back to its original forest. Thus, for $f \in \widetilde{H_A} \cup \widetilde{V_A}$ we define $\texttt{unravel}(f) \in H_A \cup V_A$ to be the corresponding element where every default hole $\Box_h$ is replaced with h. Then, we define $\varphi(f) = \varphi(\texttt{unravel}(f))$. This allows us to refer to default holes recursively: recall that we do not allow nested default holes by the definition of H_A. However, we can now safely define for $f \in \widetilde{H_A}$ that $\Box_f = \Box_{\texttt{unravel}(f)}$. Unraveling naturally induces a notion of equivalence on contexts and forests, by which f and g are *unravel equivalent*, denoted $f \approx g$, if $\texttt{unravel}(f) = \texttt{unravel}(g)$.

In the following, we often consider specific subsets of $\widetilde{V_A}$ over which we factor a forest. However, for these purposes we do not wish to distinguish between equivalent contexts under `unravel`. Moreover, we want the set to be closed under composition. To this end, we call a set $\widetilde{V} \subseteq \widetilde{V_A}$ *stable* if it is a subsemigroup of $\widetilde{V_A}$, and for every $C_1 \in \widetilde{V}, C_2 \in \widetilde{V_A}$, if $C_1 \approx C_2$ then $C_2 \in \widetilde{V}$.

4 Binary Decompositions

Equipped with default holes, we can now define binary decompositions. For words, binary decompositions are trivial – one merely splits a word recursively by concatenation (Figure 2a). For forests, we note that splitting a forest does not yield two forests, but rather a forest and a context. Then, we intuitively need to keep decomposing both parts. We illustrate how this is done before proceeding to the precise details.

▶ **Example 4.1.** Consider the forest $h = b(a+a)+a(b+a(a+b))$. We wish to decompose it as $b(a+a)+a($ $b+a(a+b)$ $)$. To this end, we write $h = C \cdot h'$ where $C = b(a+a)+a(\square)$ and $h' = b+a(a+b)$. The corresponding binary decomposition (with some extension) is depicted in Figure 3. As per Remark 1.1, in the decomposition we refer to the right-child of a node as the *factor*, and to the down-child as the *residue*. The decomposition tree is constructed as follows: we start with h at the root of the decomposition, which is given address ϵ. We then "pluck" from h the sub-forest factor h' and place it as the right-child, with address 1. The node 1 is also assigned a *context*, which is the residue C (so that $C \cdot h' = h$). Then, in the down-child (addressed 0) we place the tree that remains to be factored, which is C only with $\square_{h'}$ instead of $\square$, to signify that this remaining hole is actually h', but h' itself is already being factored elsewhere (namely from node 1). Thus, the forest at node 0 is $C \cdot \square_{h'}$.

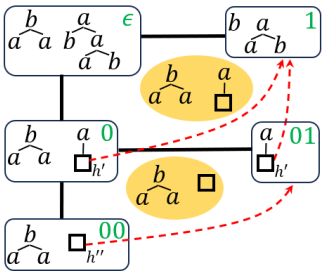


Figure 3 A (partial) binary decomposition with depicted references (dashed arrows). The first step corresponds to Example 4.1. The plucked factor h' appears in Node 1. The context C appears highlighted below the $\epsilon \to 1$ edge.

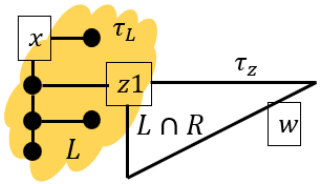


Figure 4 Decomposition structure of Lemma 6.6.

▶ **Definition 4.2** (Binary Decomposition – informal). *A* binary decomposition *of a forest* $h \in \widetilde{H_A}$ *is a binary tree* τ *where each node* x *is labeled with a forest* $\tau.\texttt{forest}(x) \in \widetilde{H_A}$ *and has a type that is either a* leaf *(*$\tau.\texttt{type}(x) = \mathfrak{L}$*) or a* binary node $\tau.\texttt{type}(x) = \mathfrak{B}$*. Binary nodes are associated with a context* $\tau.\texttt{ctx}(x) \in \widetilde{V_A}$*, and the following conditions hold:*

- *The root is labeled with* h*, i.e.,* $\tau.\texttt{forest}(\epsilon) = h$*.*
- *The forest of a binary node is decomposed correctly:* $\tau.\texttt{forest}(x) = \tau.\texttt{ctx}(x1) \cdot \tau.\texttt{forest}(x1)$*.*
- *The residue matches the context:* $\tau.\texttt{forest}(x0) = \tau.\texttt{ctx}(x1) \cdot \square_{\tau.\texttt{forest}(x1)}$*.*
- *Decompositions pluck out non-trivial forests:* $\tau.\texttt{forest}(x) \neq \tau.\texttt{forest}(xi)$ *for* $i \in \{0,1\}$*.*

See Figure 3 for an illustration; formal details are in the full version. A more elaborate example is in Figure 1a.

An important observation to be made about binary decompositions is that for a binary node x, the context $\texttt{ctx}(x1)$ actually determines both the factor and the residue. Thus, most of our reasoning focuses on the contexts, rather than the forests.

4.1 References and Stability

The proof of Simon's factorization for words crucially relies on a certain monotonicity with respect to the image of the morphism. Specifically, the following property is used: for a word w and a morphism φ, let $P_w = \{\varphi(u) : w = u \cdot v\}$ be the set of images of the prefixes of w, then if $w = w_1 w_2$, we trivially have that $P_{w_1} \subseteq P_w$. That is, the set of images of prefixes of w_1 is contained in that of w. This observation follows immediately by the definitions of a morphism and concatenation. The dual observation for P_{w_2} is slightly more involved: we have that $\varphi(w_1) \cdot P_{w_2} \subseteq P_w$, which is again obvious. Framed in the setting of binary decompositions, this relates the "available ways" of factoring a word w into subwords, with the available ways of factoring each of its children.

Recovering this property under our binary decompositions is, as it turns out, nontrivial, and leads to new definitions that complicate the remainder of the proof. At its core, the problem is that (unlike in the setting of words) each node in our binary decomposition may "refer" to other nodes by means of default holes. Moreover, this reference is meaningful as a possible decomposition, as we show in Sections 4.2 and 4.3. To illustrate this, consider the decomposition in Figure 2c. Intuitively, we first pluck out $a(a + b)$ and then pluck out b from the remaining forest, resulting in the two default holes at the bottom node. However, the default hole $\square_{a(a+b)}$ still "references" the node $a(a + b)$.

Consider a binary decomposition τ. To capture the notion of *reference*, for every node x and address u in $\tau.\texttt{forest}(x)$, we write $x@u \to y$ if $x[u] = \square_h$ such that $\tau.\texttt{forest}(y) = h$ and h was plucked to node y in some parent of x, creating this specific default hole. We write $x@* \to y$ when u is immaterial. In Figure 3, we depict references as dashed arrows. For example, $01@00 \to 1$ means that node 01, at address 00 (i.e., the label $\square_{h'}$) references node 1. We remark that capturing this definition formally involves a lot of accounting; see the full version. Of special importance are references from right-child nodes, called *inherited references* (i.e, $x1@u \to y$). These are caused when a subforest is plucked, and later on a parent of this subforest is plucked (e.g., the reference $01@00 \to 1$ mentioned above). Intuitively, these represent a "bad" way of factorization (in that it makes more "sense" to first pluck out the larger subforest). We make this precise in Lemma 4.4.

Next, for a reference $x@u \to y$ we define the *linking context from x to y* $C_{link}(\tau, x, y) = \tau.\texttt{forest}(x)[u \to \square]$, which formalizes the connection between the nodes. Continuing the example of Figure 3 we have $C_{link}(\tau, 01, 1) = a(\square)$. Intuitively, when considering a binary decomposition, the elements it represents are really the linking contexts, as these capture the "infixes" of the forest. Note that this has no analogue in words – there, simply looking at the leaves of the decomposition suffices to track the infixes.

In particular, when we wish to restrict a binary decomposition to use a specific set of contexts (e.g., when inductively decomposing a forest over an idempotent element), we apply this requirement to the linking contexts as well.

▶ **Definition 4.3** (Stable Decomposition). *For a stable context set $\widetilde{V} \subseteq \widetilde{V_A}$, we say that a decomposition τ is* stable over $\widetilde{V}$ *if $C_{link}(\tau, x, y) \in \widetilde{V}$ for every x, y such that $x@* \to y$.*

Observe that in a decomposition without inherited references, the linking contexts are compositions of contexts that appear as $\tau.\texttt{ctx}(\cdot)$. Since we assume a *stable* context sets, such compositions are already $\widetilde{V}$. We therefore have the following.

▶ **Lemma 4.4.** *Every decomposition τ with no inherited references is stable.*

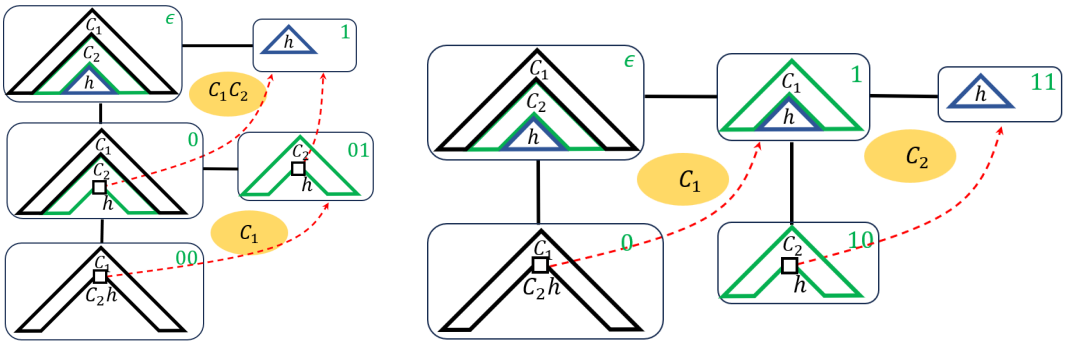
As an example, observe that the decomposition in Figure 1a has no inherited references, and is therefore stable. Moreover, as mentioned in Example 2.1, all the contexts there map to a single idempotent element.

On top of restricting the context set $\widetilde{V}$, in order to support inductive arguments we sometimes restrict the set of leaves that we allow to some set $B \subseteq \widetilde{H_A}$ called a *basis*. We then call a decomposition τ *full over* $\widetilde{V}, B$ if it is stable with respect to $\widetilde{V}$ and $\tau.\texttt{forest}(x) \in B$ for every x with $\tau.\texttt{type}(x) = \mathfrak{L}$. We call τ *partial over* $\widetilde{V}, B$ if it can be extended by further decomposing its leaves to a full decomposition over $\widetilde{V}, B$ (and in particular is stable). We denote by $\texttt{FullDec}(\widetilde{V}, B)$ and $\texttt{PartialDec}(\widetilde{V}, B)$ the sets of full and partial decompositions, respectively.

4.2 Tree Rotations and Monotonicity

The main tool we develop for working with binary decompositions, and for constructing them, is *rotations*. Intuitively, these provide a method for locally changing the shape of a tree while maintaining some desirable properties (in particular, no inherited references, hence stability by Lemma 4.4). Fundamentally, rotations can be thought of as changing the order in which we pluck out factors. In this section we illustrate the two types of rotations, and formulate their properties. We remark that despite their intuitive nature, proving these properties requires a careful analysis of how decompositions are built, and is the source of a lot of the technical challenges, presented in the full version.

Fix some context set $\widetilde{V}$. The first rotation, dubbed *sequential rotation*, is illustrated in Figure 5. For this rotation, we start with a forest $f = C_1C_2h$ where $C_1, C_2 \in \widetilde{V}$ and $h \in \widetilde{H_A}$, and we factor it first by plucking h (leaving the residue $C_1C_2\Box_h$), and then plucking $C_2\Box_h$ from the residue, leaving a further residue of $C_1\Box_{C_2h}$. We refer to this as a *Bottom-Top* factorization (BT). The rotation involves changing the order of plucking to *Top-Bottom* (TB), so that we first pluck C_2h, leaving residue $C_1\Box_{C_2h}$, and then from C_2h pluck h, leaving residue $C_2\Box_h$. It is easy to notice that both BT and TB have the same leaves. Further scrutiny



(a) BT factorization. **(b)** TB factorization.

Figure 5 Two factorizations (BT and TB) for C_1C_2h. The translation between them is a *sequential rotation*.

shows that in the BT decomposition τ, we have $\tau.01@* \to 1$, with $C_{link}(\tau, 01, 1) = C_1$. In addition, there are no inherited references in the TB decomposition (within the five nodes under consideration). We can actually apply this rotation starting at a node x that might not be the root.

To discuss such tree manipulations, we denote by $\tau \equiv_{\setminus x} \tau'$ the fact that two decompositions are identical except for the subtree under the node x. Then, the properties of BT to TB, and its dual TB to BT, are summarized as follows.

▶ **Lemma 4.5** (BT to TB – informal). *Consider a binary decomposition τ and nodes $x, x0, x1, x00, x01$ in a BT formation. Then there exists a binary decomposition $\tau' \equiv_{\setminus x} \tau$ such that:*

- $\tau'.\mathtt{ctx}(x1) = \tau.\mathtt{ctx}(x01)$ *and* $\tau'.\mathtt{ctx}(x11) = C_{link}(\tau, x01, x1)$
- $\tau'[x0] = \tau[x00]$, $\tau'[x10] = \tau[x01]$, *and* $\tau'[x11] = \tau[x1]$.

▶ **Lemma 4.6** (TB to BT – informal). *Consider a binary decomposition τ and nodes $x, x0, x1, x11, x10$ in a TB formation. Then there exists a binary decomposition $\tau' \equiv_{\setminus x} \tau$ such that:*

- $\tau'.\mathtt{ctx}(x1) = \tau.\mathtt{ctx}(x1) \cdot \tau.\mathtt{ctx}(x11)$ *and* $\tau'.\mathtt{ctx}(x01) = \tau.\mathtt{ctx}(x1)$
- $\tau'[x00] = \tau[x0]$, $\tau'[x01] = \tau[x10]$, *and* $\tau'[x1] = \tau[x11]$.

In the full version we consider an additional type of rotation called *parallel rotation* and abbreviated RL to LR, which deals with two subtrees that are not nested.

4.3 Next Contexts and Monotonicity

Equipped with the framework of rotations and stability, we can finally devise a notion of monotonicity that takes us a step closer to the proof. Here we present an abridged version; see the full version for the complete picture.

Consider a partial decomposition $\tau \in \mathtt{PartialDec}(\widetilde{V}, B)$ and a node x. We define the *Next Contexts* of x to be the set of contexts that can be used to factor x such that the extended decomposition can still be completed to a full decomposition over $\widetilde{V}$ and B.

$$\mathtt{NxtCtx}(\tau, \widetilde{V}, B, x) = \{\tau'.\mathtt{ctx}(x \cdot 1) : \tau \equiv_{\setminus x} \tau',\ \tau' \in \mathtt{PartialDec}(\widetilde{V}, B),\ \tau'.\mathtt{type}(x) = \mathfrak{B}\}$$

When clear from context, we write $\mathtt{NxtCtx}(\tau, x)$ for that set. With these notations, monotonicity is captured as follows.

▶ **Lemma 4.7.** *Consider a morphism $\varphi\colon \widetilde{V} \to V$ for some semigroup V.*

1. $\varphi[\mathtt{NxtCtx}(\tau, x0)] \subseteq \varphi[\mathtt{NxtCtx}(\tau, x)]$
2. $\varphi(\tau.\mathtt{ctx}(x1)) \cdot \varphi[\mathtt{NxtCtx}(\tau, x1)] \subseteq \varphi[\mathtt{NxtCtx}(\tau, x)]$

Proof Sketch. Consider a node x and some $C \in \mathtt{NxtCtx}(\tau, x0)$. Ideally, we would want to show a stronger claim than the statement, namely that $C \in \mathtt{NxtCtx}(\tau, x)$. We demonstrate such a case. By definition, we can extend τ to τ' so that $\tau'.\mathtt{ctx}(x01) = C$. We then split our reasoning according to which type of local structure (among BT, TB and LR) is below x. For example, if the structure is BT, then by rotating τ' to a TB structure τ'' as per Lemma 4.5 we have $\tau''.\mathtt{ctx}(x1) = \tau'.\mathtt{ctx}(x01) = C$ (see the context C_1 in Figure 5). Since $\tau'' \equiv_{\setminus x} \tau$, we have $C \in \varphi[\mathtt{NxtCtx}(\tau, x)]$, and in particular $\varphi(C) \in \varphi[\mathtt{NxtCtx}(\tau, x)]$.

For the rotation LR to RL, we cannot actually "move" the same context C from $x01$ to $x1$, but rather obtain at $x1$ a different context C' such that $C' \approx C$. But since φ is oblivious to unraveling, the result still follows. ◀

A particularly useful application of rotations is to obtain stable decompositions. To do so, we define a canonical way for decomposing a node, based on an order on contexts. For two contexts $C_1, C_2 \in \widetilde{V}$, write $C_1 \preceq_{\mathrm{pre}} C_2$ if $C_2 = C_1 \cdot C$ for some C, i.e., if C_1 is a "prefix" of C. Then, we say that a partial decomposition $\tau \in \mathtt{PartialDec}(\widetilde{V}, B)$ is *minimal below node* x if for every descendant y of x we have that $\tau.\mathtt{ctx}(y1)$ is $\preceq_{\mathrm{pre}}$-minimal among $\mathtt{NxtCtx}(\tau, y)$. In the full version we show that every partial decomposition can be extended below any node to a minimal decomposition.

Using rotations, we can show that a minimal decomposition has a simple structure, in the following sense.

▶ **Lemma 4.8** (Minimal Decompositions have No Inherited References). *Consider a decomposition τ that is minimal below x, then τ has no inherited references in the subtree of x. In particular, τ is stable below x (by Lemma 4.4).*

Proof Sketch. The complete proof is involved. However, as "supporting evidence", observe that the BT to TB rotation in Figure 5 eliminates the inherited reference $01@* \to 1$, and replaces the context C_1C_2 with the smaller C_1, progressing towards minimality. ◀

This completes our set of tools for binary decompositions. Note that for the setting of words, all the tools developed thus far are either irrelevant or follow trivially from the definition of concatenation and morphism (in particular, Lemma 4.7).

5 General Decompositions

In this section we introduce our notion of general decompositions. As in the setting of words, the main addition on top of binary decompositions are $\mathfrak{I}$dempotent nodes: nodes with unbounded arity that encapsulate a decomposition where all the contexts map to a single idempotent. However, for technical convenience we also introduce a new type of nodes called $\mathfrak{C}$entipedes. Intuitively, centipedes encapsulate a decomposition of several non-nested subforests. More precisely, a centipede is a binary decomposition whose nodes are in $0^*(\epsilon+1)$, and it does not have inherited references (see the full version for an illustration). We remark that centipede nodes can also be introduced in the setting of words. We discuss this in Section 7.

We start with the main definition.

▶ **Definition 5.1** (General Decomposition – informal). *A* general decomposition *with respect to a morphism $\varphi\colon \widetilde{V} \to V$ is a labeled tree τ where each node x has a type $\tau.\mathtt{type}(x) \in \{\mathfrak{L}, \mathfrak{B}, \mathfrak{I}, \mathfrak{C}\}$ and is labeled with a forest $\tau.\mathtt{forest}(x) \in \widetilde{H_A}$ and context $\tau.\mathtt{ctx}(x) \in \widetilde{V}$, and the following conditions hold.*

- *$\mathfrak{L}$ and $\mathfrak{B}$ nodes follow the conditions of binary decompositions.*
- *An $\mathfrak{I}$-node x has a set of children $S = \{x0, \ldots, xm\}$, such that there is an idempotent element $v \in V$ and a full decomposition $\tau' \in \mathtt{FullDec}(\varphi^{-1}[v], \tau.\mathtt{forest}[S])$ of $\tau.\mathtt{forest}(x)$ whose leaves bijectively match the children of x in τ.*
- *A $\mathfrak{C}$-node x has a set of children $S = \{x0, \ldots, xm\}$, such that there is a centipede decomposition τ' of $\tau.\mathtt{forest}(x)$ whose leaves bijectively match the children of x in τ.*

τ is full *with respect to a basis $B \subseteq \widetilde{H_A}$ if the forests at its leaves are all in B. We denote the set of full general decompositions of a forest f by $\mathtt{Dec}(\widetilde{V}, B, f, \varphi)$.*

Note that the requirement for $\mathfrak{I}$-nodes implies that τ' is a stable decomposition. Apart from facilitating proofs, this requirement makes intuitive sense: an $\mathfrak{I}$-node captures an idempotent "zone" in the decomposition. In particular, we expect that any relationships between any two related nodes in this decomposition also map to the idempotent v, not just immediate contexts. This is captured by stability (Definition 4.3), which requires that all the *linking* contexts also map to v.

▶ **Example 5.2** (TBF Full General Decomposition). In Figure 1b we present a full general decomposition of a forest in the TBF setting. The binary decomposition in Figure 1a is over an idempotent element, and thus we fold it to an $\mathfrak{I}$-node at the root. Notice that the leaves $0, 1, 2, 3, 4$ of Figure 1b bijectively match the leaves $00, 010, 0111, 0110, 1$ of Figure 1a.

The remainder of the decomposition uses binary nodes. Note that the leaves of the decomposition cannot be decomposed any further. Indeed, they are in the *standard basis*, defined shortly.

The basis B of the decomposition is used for inductive arguments. However, some special attention is given to the *standard basis* $B_A \subseteq \widetilde{H_A}$ given by $B_A = A \cup \{\sigma\Box_h : \sigma \in A, h \in H_A\} \cup \{\Box_{h_1} + \Box_{h_2} : h_1, h_2 \in H_A\}$. Note that the elements in B_A are indeed those that cannot be decomposed further.

We can now define decomposition bounds.[1] Let S be a semigroup and $\varphi\colon \widetilde{V} \to S$ a morphism. For a basis B we define $\|\varphi, B\| \coloneqq \sup_{f \in \widetilde{H_A}} \min_{\tau \in \mathtt{Dec}(\widetilde{V}, B, f, \varphi)} \mathtt{depth}(\tau)$ and by extension $\|\varphi\| = \sup_B \|\varphi, B\|$.

As mentioned above, $\mathfrak{C}$-nodes are a technical convenience. Indeed, the following result shows that we can eliminate them, at the cost of increasing the depth of a decomposition exponentially. Note that for our main result, this depth is constant, and therefore this exponential is still a constant.

▶ **Lemma 5.3.** *For every $\tau \in \mathtt{Dec}(\widetilde{V}, B_A, f, \varphi)$ there exists $\tau' \in \mathtt{Dec}(\widetilde{V}, B_A, f, \varphi)$ with no centipede nodes, and such that $\mathtt{depth}(\tau') \leq 2^{\mathtt{depth}(\tau)+1} - 2$.*

6 Bounded Decompositions

As mentioned in Section 1, our bounded-decomposition theorem holds under a semantic condition, which we now present.

6.1 $\mathcal{R}$-Alignment

▶ **Definition 6.1** ($\mathcal{R}$-aligned Forests and Morphisms). *Consider a semigroup S, a stable context set $\widetilde{V}$ and a morphism $\varphi\colon \widetilde{V} \to S$. A forest $f \in \widetilde{H_A}$ is* $\mathcal{R}$-aligned *over φ if for every $f_1, f_2 \in \widetilde{H_A}$ and $C_1 \neq C_2 \in \widetilde{V}$ such that $f = C_1(f_1) = C_2(f_2)$ and $\varphi(C_1)\mathcal{J}\varphi(C_2)$ there exists $r \in S$ such that $r\mathcal{J}\varphi(C_1)$ and $\varphi(C_2) = \varphi(C_1) \cdot r$.*

A morphism φ is said to be $\mathcal{R}$-aligned *if all forests $f \in \widetilde{H_A}$ are $\mathcal{R}$-aligned over φ.*

Note that since the requirement in the definition holds for every C_1, C_2, it is in fact symmetric: there is also $r'\mathcal{J}\varphi(C_1)$ such that $\varphi(C_1) = \varphi(C_2) \cdot r'$.

Intuitively, the definition states the following: suppose we decompose f in two different ways (by plucking out different sub-forests), such that the remaining contexts are $\mathcal{J}$-equivalent, then not only are these contexts also $\mathcal{R}$-equivalent, but they are $\mathcal{R}$-equivalent using multiplicands that are within the same $\mathcal{J}$-class (hence the term $\mathcal{R}$-aligned).

The requirement that $r\mathcal{J}\varphi(C_1)$ allows us to show that $\mathcal{R}$-alignment survives through taking certain quotients and subsemigroups, as follows.

▶ **Lemma 6.2.** *Consider an $\mathcal{R}$-aligned morphism $\varphi\colon \widetilde{V} \to S$ and an ideal $I \subseteq S$, then the following hold.*

- *Let $\mathtt{quo}_I\colon S \to S/I$ be the quotient morphism, then $\mathtt{quo}_I \circ \varphi\colon \widetilde{V} \to S/I$ is $\mathcal{R}$-aligned.*
- *If I is a $\mathbb{0}$-minimal ideal[2], let $\widetilde{V'} = \varphi^{-1}[I]$ then $\varphi|_{\widetilde{V'}}\colon \widetilde{V'} \to I$ is $\mathcal{R}$-aligned.*

We can finally state our main theorem.

▶ **Theorem 6.3.** *Consider a finite semigroup S and an $\mathcal{R}$-aligned morphism $\varphi\colon \widetilde{V} \to S$, then $\|\varphi\| \leq 4|S| - 3$.*

---

[1] The precise definition is more nuanced, see the full version.

[2] See the full version for the definition. Here it can be taken as "some condition on I".

The proof of Theorem 6.3 is by induction, with three base cases depending on the structure of S: a group, a simple or $\mathbb{0}$-simple semigroup, or a null semigroup. The latter is less involved and appears in the full version. We turn to sketch the main ideas of the interesting cases.

▶ Remark 6.4 (Comparison with the setting of words). Simon's theorem for words can actually be obtained (in full generality) as a particular case of Theorem 6.3. Indeed, words can be described as vertical forests, which trivially satisfy $\mathcal{R}$-alignment.

Moreover, we can obtain the best known upper bound for Simon's theorem on words, as we discuss in Remark 6.7.

6.2 Decomposition over a Group

▶ **Lemma 6.5.** *For a group V we have $\|V\| \leq 2|V| - 1$.*

Proof Sketch. The proof is by induction over $|\texttt{NxtCtx}(\tau, x)|$ at each node x of a decomposition. Let $f \in \widetilde{H_A}$, we start by decomposing f "as much as possible" using $\varphi^{-1}[\mathbb{1}]$ (i.e., over the unit of V, which is idempotent). We can then fold this decomposition to a single $\mathfrak{I}$-node. Using Lemma 4.7, we can show that the children of this $\mathfrak{I}$-node cannot be decomposed with $\mathbb{1}$, and therefore are decomposable over a smaller set of contexts, so continue by induction.

If f is not decomposable over $\varphi^{-1}[\mathbb{1}]$, we start by decomposing f using a maximal-depth centipede decomposition τ, such that τ is *minimal*. We then use the monotonicity of Lemma 4.7 together with the properties of groups and the centipede structure to conclude that each leaf y of the centipede either has a smaller $\texttt{NxtCtx}(\tau, y)$ set (so we can apply induction), or $\mathbb{1} \in \texttt{NxtCtx}(\tau, y)$, and we apply the case proved above. Then, we can fold the centipede to a single $\mathfrak{C}$-node, yielding a full general decomposition of the desired depth. ◀

6.3 Decomposition over Simple and $\mathbb{0}$-Simple Semigroup

This case is the most challenging part of our contribution. To frame it, we recall a central part of the proof of Simon's theorem [1]. In broad strokes, a word w is written as $w = u_0 a b u_1 a b u_2 a b \cdots a b u_k$ for some letters a, b such that the u_i do not contain the subsequence ab. Then, w is viewed as $w = u_0 a (b u_1 a)(b u_2 a) \cdots (b u_{k-1} a) b u_k$ and it is noticed that all the $b u_i a$ have the same $\mathcal{R}$- and $\mathcal{L}$-classes, and thus belong to the same $\mathcal{H}$-class, which is moreover idempotent (as an element in a quotient semigroup). The takeaway for us is that we wish to split a forest by consecutive elements that dictate a pair (L, R) of $\mathcal{L}$- and $\mathcal{R}$-classes.

Unfortunately, for forests this approach becomes much more complicated. In fact, it generally breaks down (as we show in Section 7), and is the reason for the introduction of $\mathcal{R}$-alignment. Importantly, we notice that for simple semigroups without $\mathbb{0}$, there is a single $\mathcal{J}$-class. Thus, for an $\mathcal{R}$-aligned morphism, for every $C, C' \in \widetilde{V_A}$ and $f, f' \in \widetilde{H_A}$ such that $C(f) = C'(f')$, we have $\varphi(C) \mathcal{R} \varphi(C')$.

We now give an overview of our approach. The first point to consider is that for forests, the notion of "two consecutive letters" makes only partial sense. Consider a node x in a binary decomposition τ. Similarly to $\texttt{NxtCtx}(\tau, x)$, we can consider the next *two* contexts, i.e., $\texttt{ctx}(x1)$ and $\texttt{ctx}(x11)$. Since we are interested in $\mathcal{L}$- and $\mathcal{R}$-classes, we define $\texttt{LR}(\tau, x)$ to be the set of (L, R) pairs such that $L = [\tau'.\texttt{ctx}(x1)]_{\mathcal{L}}$ and $R = [\tau'.\texttt{ctx}(x11)]_{\mathcal{R}}$ in some extension τ' of τ. That is, the possible "consecutive" (L, R) classes that can be used to decompose x.

Using Lemma 4.7 and results about Green's relations, we show in the full version a monotonicity property of $\texttt{LR}(\tau, x)$: if y is a descendant of x then $\texttt{LR}(\tau, y) \subseteq \texttt{LR}(\tau, x)$. We then proceed to show the bound on decomposition depth.

▶ **Lemma 6.6** (Decomposition over Simple Semigroup – abridged). *Let $\varphi\colon \widetilde{V} \to S$ with S a simple semigroup and φ $\mathcal{R}$-aligned. Then $\|\varphi\| \leq 2|S| + 1$.*

Proof Sketch. Let $f \in \widetilde{H_A}$. As mentioned above, we consider the case that S has no $\mathbb{0}$. If it does, we start by factoring over $\varphi^{-1}[\mathbb{0}]$, similarly to the case of Lemma 6.5.

The proof proceeds by induction over $|\mathtt{LR}(\tau, x)|$, assuming an existing decomposition τ and a node x, where we prove that we can extend τ below x with low-enough depth. Thus, consider a node x, and fix a pair $(L, R) \in \mathtt{LR}(\tau, x)$.

We now decompose f in two parts (see Figure 4). First, we decompose f with a maximal-depth centipede τ_L over the contexts $\varphi^{-1}[L]$, with an additional crucial requirement: that every right-child (i.e., "leg" of the centipede) $z1$ has $\tau_L.\mathtt{NxtCtx}(z1) \cap R \neq \emptyset$. That is, we can continue decomposing $z1$ with a context mapped to R. This captures the analogy of an (L, R) pair for words.

Then, for every such leaf $z1$ we decompose it using a maximal-depth binary decomposition τ_z with the following property: τ_z is minimal (as in Lemma 4.8) over $L \cap R$, and for every node y in τ_z, every way of factoring $\tau_z.\mathtt{forest}(y) = C'(f')$ satisfies $\varphi(C') \in R$. Using this latter condition we show two significant properties:

- $L \cap R$ is a group, and therefore we can use the group bound from Lemma 6.5 to replace τ_z with a low-depth decomposition with the same leaves.
- Every *leaf* w of τ_z has $(L, R) \notin \mathtt{LR}(\tau_z, w)$. To show this we rely on $\mathcal{R}$-alignment twice: once for showing that any way of factoring a leaf of w must start with a context mapped to R, and once for assuming that $(L, R) \in \mathtt{LR}(\tau_z, w)$ and reaching a contradiction. We remark that for leaves of the form $w'1$ the former is trivial, by the property above. The crucial part is that due to $\mathcal{R}$-alignment this also holds for $w'0$.

We can therefore decompose every leaf w of τ_z inductively, since it has a smaller $\mathtt{LR}$ set. Then, we fold τ_L to a single $\mathfrak{C}$-node, obtaining our desired decomposition. ◀

▶ Remark 6.7. The bound in Lemma 6.6 is given in terms of $|S|$. A finer approach relies on $N(S)$ – a quantity associated with the regular $\mathcal{J}$-classes [7, Chapter 18, Section 3], with $N(S) \leq |S|$. We can obtain a bound of $2N(S) - 1$ using a slightly more complicated proof: instead of choosing an arbitrary (L, R) pair in the induction, we choose it so that the R class coincides with the root, allowing us to merge τ_L and τ_z. This improved bound lifts to the general case, and apart from improving the bound we obtain, also yields the state-of-the-art upper bound in the particular case of words, viewed as vertical forests.

6.4 Decomposition over General Semigroups – Main Theorem

We can finally sketch the proof of Theorem 6.3. The proof is by induction over $|S|$, where the base cases are when S is a group (Section 6.2), simple or $\mathbb{0}$-simple semigroup (Section 6.3), or null semigroup.

Proof Sketch of Theorem 6.3. Consider a forest f. The inductive argument follows a similar scheme to the proof of Simon's theorem for words. We consider a nonzero element $a \in S$ that is $\mathcal{J}$-minimal and its corresponding ideal $M = S^1 a S^1$. We show that this is a minimal non-zero ideal of S with respect to containment, and that $|M| > 1$. Then, by Lemma 6.2 we have that $\mathtt{quo}_M \circ \varphi$ is $\mathcal{R}$-aligned. Moreover, $|S/M| < |S|$. We can therefore apply the induction hypothesis and obtain a decomposition τ of f over S/M. We further notice that S/M is fairly similar to S, with the only difference being that M is collapsed to a single element. However, this element is idempotent and is $\mathbb{0}_{S/M}$. By carefully analyzing the constructions of the base case, we can actually show that there is at most one $\mathfrak{I}$-node mapped to M in τ. Thus, all we need to do is replace this $\mathfrak{I}$-node by a decomposition over S, with bounded depth.

To this end, we show that M is either null (in which case we are done), or $|M| < |S|$, in which case we show that it is a $\mathbb{0}$-minimal ideal, so by the second part of Lemma 6.2 we can apply the induction hypothesis to conclude the theorem. ◀

7 Additional Results, Discussion and Future Research

In this overview we present the main ideas of our analogue of Simon's factorization theorem for forests. We briefly mention some contributions that appear only in the full version.

A Decomposition Reduction Scheme. In the full version we present the following general result (not relying on $\mathcal{R}$-alignment):

▶ **Lemma 7.1** (Reduction Lemma – abridged). *Consider two finite semigroups V_1, V_2 and a semigroup morphism $\varphi\colon V_1 \to V_2$, then for every context set $\widetilde{V}$ and a morphism $\beta\colon \widetilde{V} \to V_1$ we have $\|\beta\| \leq \|\varphi \circ \beta\| \cdot \max_{\substack{e \in V_2 \\ e^2=e}} \left\|\varphi^{-1}[e]\right\|$.*

This result allows us to obtain bounds on decomposition depth for one morphism (β) based on known bounds for e.g., quotient semigroups ($\varphi \circ \beta$), assuming knowledge of decompositions over the idempotent subsemigroups. We remark that a similar result is used in the proof of Simon's result for words.

Relaxing $\mathcal{R}$-Alignment. A natural question is whether $\mathcal{R}$-alignment is a necessary condition to obtain bounded-depth decompositions. The answer to this is no – it is not hard to check that the TBF example from Example 1.2 is not $\mathcal{R}$-aligned. However, it is possible to show that it has bounded decompositions by following the proof of Theorem 6.3 and noticing that after getting rid of prefixes that map to a "sink-element using a depth-2 decomposition, the remaining forests do satisfy $\mathcal{R}$-alignment. Thus, the failure of $\mathcal{R}$-alignment in TBF is an edge case, and is not fundamental.

The more important question is whether any condition is even necessary. In this section, we show that without any restrictions on φ, it might not admit bounded decompositions.

▶ **Theorem 7.2.** *There is a morphism $\varphi\colon \widetilde{V} \to V$ for some V that is not $\mathcal{R}$-aligned, such that $\|\varphi\| = \infty$.*

We bring here only the rough idea, without describing the concrete counterexample. Intuitively, we construct a morphism based on a tree-language of binary trees, with the following property: each idempotent element v is associated with a *side* – right or left, such that when decomposing a binary tree using a context $C \in \varphi^{-1}[v]$, only the right or left subtree of the tree (according to the associated side) is being decomposed.

This means that in a balanced tree, each $\mathfrak{I}$-node can decompose at most half the tree. Thus, a tree of depth n requires a decomposition of depth at least $\log n$, concluding the claim.

It thus remains to ask whether we can relax $\mathcal{R}$-alignment to obtain a necessary condition. A careful examination of the proof of Section 6.3 shows that $\mathcal{R}$-alignment can be relaxed to require that there is a bound on the number of "sequential violation" of the $\mathcal{J} \to \mathcal{R}$ condition of Section 6.1. In fact, this is exactly the idea upon which the lower bound in Theorem 7.2 is based on.

Discussion

Absent from this work are applications of bounded decompositions for forests. One immediate application is an analogue of fast infix query for words [1], where we want to compute the image of an "infix" of a forest under a morphism. Note that proper infixes of forests are contexts

rather than forests, and formulating precisely what an infix is is nontrivial. Unfortunately, this is not useful from an algorithmic approach, since *computing* a decomposition tree is not clearly efficient. Indeed, following our proof, it requires computing $\texttt{NxtCtx}(\tau, x)$ for certain nodes, and it is not clear how to do that. This, however, is not surprising, as the same difficulty arises in words.

Nonetheless, most applications of Simon's factorization are in proofs (e.g., boundedness of weighted automata [9]), where we hope our work would enable lifting results on words to forests. We remark that as a sanity check, a naive application is as a simple pumping argument: given a general decomposition with an $\mathfrak{I}$-node, we can modify the binary decomposition that corresponds to this $\mathfrak{I}$-node by repeating a binary node (e.g., replacing a forest $C(f)$ with $C(C(f))$). This readily yields a decomposition of a different, "larger" forest, with the same image under the morphism.

An existing algorithmic approach circumventing the difficulty of computing Simon's factorization is built on *forward Ramsey decomposition* [6, 7]. In fact, a similar approach using forward factorizations has been introduced for forests as well, yielding a solution for infix querying on forests [5, Appendix E]. Such decompositions can be computed for every forest (with bounded height), but provide weaker guarantees on their analogue of $\mathfrak{I}$-nodes. Our work can be viewed as a condition (namely $\mathcal{R}$-alignment) under which a forward decomposition can be turned into a "Simon" decomposition.

References

1 Mikołaj Bojańczyk. Factorization forests. In Volker Diekert and Dirk Nowotka, editors, *Developments in Language Theory, 13th International Conference, DLT 2009, Stuttgart, Germany, June 30 - July 3, 2009. Proceedings*, volume 5583 of *Lecture Notes in Computer Science*, pages 1–17, Berlin, Heidelberg, 2009. Springer. doi:10.1007/978-3-642-02737-6_1.

2 Mikołaj Bojańczyk. Algorithms for regular languages that use algebra. *SIGMOD Record*, 41(2):5–14, 2012. doi:10.1145/2350036.2350038.

3 Mikołaj Bojańczyk, Howard Straubing, and Igor Walukiewicz. Wreath products of forest algebras, with applications to tree logics. *Logical Methods in Computer Science*, 8(3:19), 2012. doi:10.2168/LMCS-8(3:19)2012.

4 Mikołaj Bojańczyk and Igor Walukiewicz. Forest algebras. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 107–132. Amsterdam University Press, 2007.

5 Mikołaj Bojańczyk and Paweł Parys. Efficient evaluation of nondeterministic automata using factorization forests. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, volume 6198 of *Lecture Notes in Computer Science*, pages 515–526. Springer, 2010. doi:10.1007/978-3-642-14165-2_44.

6 Thomas Colcombet. On factorisation forests. *CoRR*, abs/cs/0701113, 2007. doi:10.48550/arXiv.cs/0701113.

7 Thomas Colcombet. The factorisation forest theorem. In Jean-Éric Pin, editor, *Handbook of Automata Theory, Volume I*, pages 653–693. EMS Press, 2021. doi:10.4171/AUTOMATA-1/18.

8 Imre Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990. doi:10.1016/0304-3975(90)90047-L.

9 Imre Simon. On semigroups of matrices over the tropical semiring. *RAIRO-Theoretical Informatics and Applications*, 28(3-4):277–294, 1994. doi:10.1051/ita/1994283-402771.