Regular D-length: A tool for improved prefix-stable forward Ramsey factorisations

Théodore Lopez, Benjamin Monmege, Jean-Marc Talbot

Regular $\mathcal{D}$-Length: a Tool for Improved Prefix-Stable Forward Ramsey Factorisations

Théodore Lopez[a], Benjamin Monmege[a] and Jean-Marc Talbot[a]

[a]Aix-Marseille Univ, CNRS, LIS, Marseille, France

ARTICLE INFO

Keywords:
Ramsey factorisation
Green's relations
Fast infix query evaluation

Abstract

Recently, Jecker has introduced and studied the regular $\mathcal{D}$-length of a monoid, as the length of its longest chain of regular $\mathcal{D}$-classes. We use this parameter in order to improve the construction, originally proposed by Colcombet, of a deterministic automaton that allows to map a word to one of its forward Ramsey splits: these are a relaxation of factorisation forests that enjoy prefix stability, thus allowing a compositional construction. For certain monoids that have a small regular $\mathcal{D}$-length, our construction produces an exponentially more succinct deterministic automaton. Finally, we apply it to obtain better complexity result for the problem of fast infix evaluation.

1. Introduction

Computing Ramsey-type information of finite sequences of elements in a finite semigroup or monoid, e.g. the idempotent factors the sequence contains, can be performed by using forest factorisation, introduced by Simon [1]: he proved that every morphism from words to a finite semigroup admits a Ramsey factorisation forest (where all internal nodes that are not binary coincide with the presence of consecutive factors which all evaluate to the same idempotent element) of linear height with respect to the cardinality of the semigroup. Another way to describe such a factorisation forest is to use the notion of Ramsey split, introduced by Colcombet [2], that maps each position of the word to a natural number, called height. As such, a factorisation forest is tailored for the study of each input word independently. Moreover, the construction of factorisation forests a priori lacks compositionality: given a factorisation forest for some word u and a letter a, it may not be easy to directly build a factorisation forest for the word $u \cdot a$, fulfilling the requirement on the *height*.

To cope with this problem, Gastin and Krishna [3] use an unambiguous automaton that associates with each word a particular Ramsey split. At each step of the computation of the automaton, a bounded number of candidate splits are kept in memory. Alternatively, Colcombet proposed in [4, 2] the weaker notion of forward Ramsey split to trade unambiguity of the previously cited automaton for determinism: this permits to keep in memory a single split along the computation. From this split, interesting combinatorial properties like idempotent factors can still be deduced, though we do no longer have the factorisation forest, that might be fully needed for some other purpose.

Our goal is to reduce the size of the deterministic automaton computing a forward Ramsey split. To do so, we use a new complexity parameter, the *regular $\mathcal{D}$-length* $L(M)$ of a finite monoid (or semigroup) M, recently highlighted by Jecker [5]: it is the length of the longest chain of regular $\mathcal{D}$-classes ($\mathcal{D}$-classes containing an idempotent element). Jecker uses this new measure to bound more tightly the Ramsey function associated to M, that gives the minimal size of a word having sufficiently many consecutive factors with the same idempotent element (like non-binary nodes of factorisation forests). We revisit the construction of the deterministic automaton to obtain a smaller one, having only $|M|^{\min(2L(M)-1,|M|)}$ states (instead of $|M|^{|M|}$ for Colcombet), $|M|$ being the cardinality of M. Whereas $L(M)$ is always at most $|M|$, Jecker gives examples where it is much smaller. As a simple example, the regular $\mathcal{D}$-length of the transformation monoid over k elements (containing all mappings from $\{1, 2, \dots, k\}$ to itself) is $k+1$ [5, 6] whereas its cardinality is k^k.

We define in Section 2 a coarser version of forward Ramsey splits, that we call forward Ramsey *labelled* splits. It will reduce the number of possible heights (the natural numbers mapped by the split) from $|M|$ to $2L(M) - 1$, at the price of keeping an additionnal *label*. Hence, our contribution is twofold: we reduce the size of the automaton computing the splits, and we reduce the number of heights. Yet, we explain in Remark 7 why our automaton can also compute a forward Ramsey split by using the height function defined by Colcombet.

Computing deterministic Ramsey factorisations has several applications. For instance, it has been used for MSO query enumeration [7, 8]: given an MSO query φ, and a finite word u, the task is to precompute a data structure in time linear in the length of u that allows one to enumerate all factors satisfying φ with constant delay. Another example, that we study in more details in Section 3, is the infix evaluation problem: given a regular language L and a finite word u, an infix query is of the form "does the factor of u between positions i and j belong to L?". Once the word u is known, the problem is to quickly answer such infix queries. The basic approach would be to precompute the answer for all factors of u. However it requires quadratic time with respect to the length of the word, which is not doable in

theodore.lopez@univ-amu.fr (T. Lopez);
benjamin.monmege@univ-amu.fr (B. Monmege);
jean-marc.talbot@univ-amu.fr (J. Talbot)
ORCID(s): 0000-0002-1901-1274 (T. Lopez); 0000-0002-4717-9955 (B. Monmege)

practice when the word represents large inputs. The *fast infix evaluation* problem asks to pre-compute a data structure in time *linear* in u that allows one to answer infix queries in constant time (relative to u). The constants in both the precomputation and the query evaluation depend on the representation of L. Ramsey factorisations, with respect to the syntactic monoid of the language L, give a framework for quick computation. In [9, 10], factorisation forests are used for the data structure, while in [4], Colcombet uses forward Ramsey splits. We study how forward Ramsey labelled splits can be used in this context, with an improved query evaluation.

2. An improved forward Ramsey factorisation

We start by recalling useful algebraic definitions. A *semigroup* $(M, \cdot)$ is given by a set and an associative product operation. When M contains additionally a neutral element $\mathbf{1}$ for the product (for all m in M, $\mathbf{1} \cdot m = m \cdot \mathbf{1} = m$), then $(M, \cdot, \mathbf{1})$ is called a *monoid*. An element m of M is said to be *idempotent* if $m \cdot m = m$. We let $|M|$ be the cardinality of a finite monoid M.

For a *finite* monoid $(M, \cdot, \mathbf{1})$ (in case of a semigroup, a neutral element is added if necessary), we define the preorder[1] $\leq_{\mathcal{D}}$ by letting, for all $(m, m') \in M^2$, $m \leq_{\mathcal{D}} m'$ if there exists $(p, p') \in M^2$ such that $m = p \cdot m' \cdot p'$. This induces an equivalence relation $\sim_{\mathcal{D}}$ on M defined as $m \sim_{\mathcal{D}} m'$ if $m \leq_{\mathcal{D}} m'$ and $m' \leq_{\mathcal{D}} m$. The *$\mathcal{D}$-classes* (or shortly *classes*, since we are only preoccupied with $\mathcal{D}$-classes in this article) of M are the equivalence classes defined by $\sim_{\mathcal{D}}$. We say that a class is *regular* if it contains an idempotent element. The preorder $\leq_{\mathcal{D}}$ can also be extended to compare classes: for two classes D_1 and D_2, we have $D_1 \leq_{\mathcal{D}} D_2$ if $m \leq_{\mathcal{D}} m'$ for some elements $m \in D_1$ and $m' \in D_2$. We denote by $<_{\mathcal{D}}$ the strict preorder relation associated with $\leq_{\mathcal{D}}$ (on elements of M or their classes). Following Jecker [5], we note $L(M)$ the *regular $\mathcal{D}$-length* of a finite monoid M, that is the number of classes in one of its longest chains (i.e. increasing sequences) of regular classes.

Running examples Consider the language $K = (ab)^* \cup c\Sigma^* \cup \Sigma^* d$ over the alphabet $\Sigma = \{a, b, c, d\}$. Its syntactic monoid M_K is the set of words Σ^* quotiented by the following equations:

$$aba = a \qquad bab = b \qquad c = ca = cb = cc$$
$$aa = bb = ac = bc = da = db = dc = aaa = aab = baa$$
$$d = ad = bd = dd$$

In Figure 1, we group elements by classes. The regular $\mathcal{D}$-length of M_K is $L(M_K) = 3$.

For some integer $p \geq 2$, consider the language $L_p = b^* a^p a^* b^*$ of words over the alphabet $\{a, b\}$ with exactly one block of a's, which is of size at least p. Its syntactic monoid M_{L_p} is the set of words Σ^* quotiented by the following equations:

$$b = bb \qquad a^p = a^p a$$
$$aba = abaa = aaba = abab = baba$$
$$aba = ba^i b \quad \text{for all } 1 \leq i < p$$

The diagram of classes is represented in Figure 2 in the case $p = 3$. Remark that all classes contain exactly one element. The regular $\mathcal{D}$-length of M_{L_p} is $L(M_{L_p}) = 3$, and there are two distinct maximal regular $\mathcal{D}$-chains: (aba, a^p, ε) and (aba, b, ε).

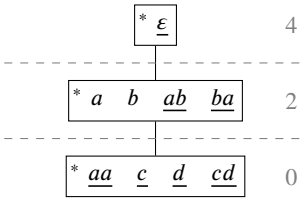

Figure 1: Classes of M_K, where stars denote classes containing idempotent elements, and where idempotent elements have been underlined. Grey numbers denote the height function we define afterwards.

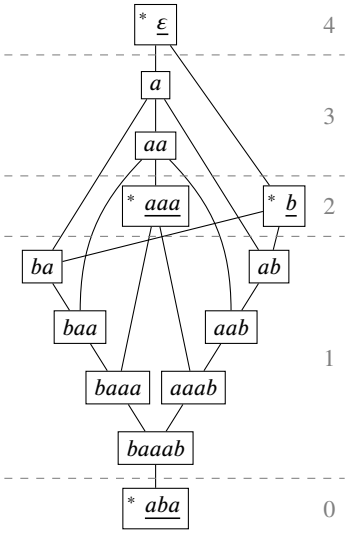

Figure 2: Classes of M_{L_3}

Forward Ramsey splits. Let Σ be a finite alphabet and Σ^* the set of words over Σ. We denote by $\mathbf{N}$ the set of all natural numbers and by $[k, m]$, with $k \leq m$, the set of natural numbers $\{k, k+1, \ldots, m\}$. Then, we let $[0, n]$ to be the set of *positions* of the word, where position 0 denotes the one before the beginning of the word, position 1 denotes the position in-between letters a_1 and a_2, while position n denotes the one after the end of the word. We let $u_{i,j} = a_{i+1} a_{i+2} \cdots a_j$ be the *factor* between positions i and j.

Let M be a finite monoid, and $\varphi \colon \Sigma^* \to M$ be a morphism of monoids. Ramsey theory aims at associating with a word $u = a_1 \cdots a_n \in \Sigma^*$ enough information so

[1] This relation could be more properly denoted as $\leq_{\mathcal{J}}$ since it should be associated with the $\mathcal{J}$-Green's relation. However, $\mathcal{J}$ and $\mathcal{D}$ coincide when M is finite, as recalled in [11, Lemma 5].

as to be able to track idempotent factors of the sequence $\varphi(a_1), \dots, \varphi(a_n)$. To achieve this, Simon [1] showed the existence of factorisation forests of linear height with respect to the cardinality $|M|$ of M for any word u. An alternative presentation was proposed by Colcombet [2, 4] as the notion of *Ramsey split*. A split is a mapping $s : [0, n] \to \mathbf{N}$ associating a height with each position in a word $u = a_1 \cdots a_n$. We have depicted one such split in Figure 5. A split defines an equivalence relation $\sim_s$ such that for all positions $x \leq y$:[2]

$$x \sim_s y \quad \text{iff } s(x) = s(y) \text{ and } \forall x \leq z \leq y \quad s(x) \leq s(z)$$

Then, a split is said to be Ramsey if for all $\sim_s$-equivalence classes, there exists an idempotent element $e \in M$ such that all positions $x < y$ in the class delimitate the element $\varphi(u_{x,y}) = e$. The problem of computing, for a given word, a factorisation forest of linear height translates into the search for a Ramsey split with a maximal height bounded linearly with respect to M.

Unfortunately, those approaches do not provide methods to compute such Ramsey splits in a compositional way, i.e. being able to compute the split of a word ua solely with respect to the split of u with a time complexity independent on the length of u. Such a compositional computation is not known, and a priori difficult because there are many Ramsey splits for a given word, some being easily extendable when reading a new letter a (while maintaining the linear bound on the maximal height), some not.

To address this issue, Colcombet relaxed the Ramsey condition on splits, to introduce *forward Ramsey* splits. A split is said to be *forward Ramsey* if all positions $x < y$ and $x' < y'$ in the same $\sim_s$-equivalence class are such that $\varphi(u_{x,y}) \cdot \varphi(u_{x',y'}) = \varphi(u_{x,y})$. Notice that for all positions $x < y < z$ of the same $\sim_s$-equivalence class, this implies that $\varphi(u_{x,z}) = \varphi(u_{x,y}) \cdot \varphi(u_{y,z}) = \varphi(u_{x,y})$. Forward Ramsey splits contain less information than factorisation forests, but are at least sufficient to detect some idempotent factors. Indeed, taking $x = x'$ and $y = y'$ in the forward Ramsey property, $\varphi(u_{x,y})$ is an idempotent element of M, for all $x < y$ of the same $\sim_s$-equivalence class.

The main result of Colcombet [2, 4] is to compute such a forward Ramsey split by means of a deterministic automaton: the automaton labels each position of the word with some state (from a set Q) which is a finite sequence $\langle m_1, \dots, m_k \rangle$ of elements of M, each state being associated with a height. Writing $\lambda : [0, n] \to Q$ the labelling, and $h : Q \to \mathbf{N}$ the height of states, we call *labelled split* the pair (λ, h). The underlying split s is then obtained as the composition $h \circ \lambda$.

The height is obtained by associating with each element m of M a height $h_0(m)$, and then by stating that $h(\langle m_1, \dots, m_k \rangle) = h_0(m_k)$. In the work of Colcombet, it is required that the height h_0 is an injective mapping such that

$$\forall (m, m') \in M^2 \qquad m \leq_D m' \implies h_0(m) \leq h_0(m') \quad (1)$$

[2] With respect to the original definition ($x \sim_s y$ if $s(x) = s(y)$ and $\forall x \leq z \leq y \quad s(x) \geq s(z)$), we inverted the order of heights to be coherent with the choice we do afterwards.

A first improvement that we propose is to remove the injective requirement, in order to diminish the range of heights.

This is the basis for the reduction of the size of Q we then obtain. Indeed, the state space Q of Colcombet consists of *valid* sequences $\langle m_1, \dots, m_k \rangle$ of elements of M, where validity means that

- the classes of elements form a *C*hain:

$$m_1 <_D m_2 <_D \cdots <_D m_k \quad (C)$$

- the *P*roduct of consecutive elements of the sequence is in the same class as the first element:

$$\forall 1 \leq i < j \leq k \qquad m_i \sim_D m_i \cdot m_{i+1} \cdots m_j \quad (P)$$

The length k of each valid sequence is thus bounded by $|M|$, which also bounds the number of states of Q by $|M|^{|M|}$.

Assuming that we design a height function h_0 in a way that diminishes the maximal height, a natural alternative definition of validity consists in only requiring that sequences $\langle m_1, \dots, m_k \rangle$ are such that

$$h_0(m_1) < h_0(m_2) < \cdots < h_0(m_k) \quad (H)$$

To distinguish the cases, we will speak of CP-validity for the one of Colcombet, and H-validity for ours.

The loss of injectivity for the height mapping h_0 will force us to strengthen the forward Ramsey condition on the labelled split (in order to keep the desired combinatorial properties, like idempotency). First, we refine the equivalence relation $\sim_s$, with s being the split $h \circ \lambda$; to do so, we take into account the labels and enforce that two equivalent positions must have the same label and not only the same height: $\sim_{\lambda,h}$ is the smallest equivalence relation such that for all $x \leq y$:

$$x \sim_{\lambda,h} y \quad \text{if } \lambda(x) = \lambda(y) \text{ and } x \sim_{h\circ\lambda} y$$

The labelled split (λ, h) is then called *forward Ramsey* for the word u if the split $h \circ \lambda$ is forward Ramsey with respect to the equivalence class $\sim_{\lambda,h}$, i.e. if all positions $x < y$ and $x' < y'$ in the same $\sim_{\lambda,h}$-equivalence class are such that $\varphi(u_{x,y}) \cdot \varphi(u_{x',y'}) = \varphi(u_{x,y})$. Clearly, if $h \circ \lambda$ is a forward Ramsey split, then (λ, h) is a forward Ramsey labelled split. The reciprocal implication does not always hold (see Remark 4).

New height mapping We start by defining the height mapping $h_0 : M \to \mathbf{N}$, associating with each element of the monoid a height, satisfying the monotonous condition (1). Intuitively, it measures the longest chain of regular classes below an element, differentiating elements that belong to a regular class (giving them an even height) and others (giving them an odd height).

Definition 1. *For a monoid M, the function $h_0 : M \to \mathbf{N}$ is defined as follows: for an element $m \in M$, letting k be the maximal length of a chain $(D_1, \dots, D_k)$ of regular classes such that $e \leq_D m$ for some $e \in D_k$, we let*

$$h_0(m) = \begin{cases} 2k - 2 & \text{if } m \text{ belongs to a regular class} \\ 2k - 1 & \text{otherwise} \end{cases}$$

If an element $m \in M$ belongs to a regular class, the length k is at least 1, so that $h_0(m) \geq 0$. Moreover, the minimal classes, with respect to the $\leq_D$ preorder, are regular,[3] which explains why the height of all elements is indeed non-negative. Moreover, the maximal class, with respect to the $\leq_D$ preorder, is the class of the neutral element of the monoid, and is thus regular. Its associated length k is $L(M)$ by definition of the regular D-length, which implies that the maximal height is $2L(M) - 2$. The height of all elements thus belongs to $[0, 2L(M) - 2]$. Heights of elements (that only depend on the decomposition into classes) are depicted in the examples of Figures 1 and 2.

Lemma 2. *Let $(m_1, m_2) \in M^2$. Then, the following properties hold:*

1. *If $m_1 \leq_D m_2$, then $h_0(m_1) \leq h_0(m_2)$.*
2. *If $h_0(m_1) = h_0(m_2)$ and $m_1 \cdot m_2 = m_1$, then m_1 and m_2 are in the same class, which is moreover regular.*

PROOF. 1. Consider a maximal chain $(D_1, \dots, D_k)$ of regular classes such that $e \leq_D m_1$ for any $e \in D_k$, so that $h_0(m_1) \in \{2k-2, 2k-1\}$. Since $e \leq_D m_1 \leq_D m_2$, this sequence is also below m_2, and thus $h_0(m_2) \geq 2k - 2$. The only contradiction to the conclusion $h_0(m_1) \leq h_0(m_2)$ would be if $h_0(m_1) = 2k - 1$ and $h_0(m_2) = 2k - 2$, i.e. if m_1 is not in a regular class while m_2 is. But then, the sequence $(D_1, \dots, D_k, D)$ with D the class of m_2 would be a chain of regular classes below m_2 of length $k + 1$, which would imply that $h_0(m_2) \geq 2(k + 1) - 2$.

2. Let n be such that $e = (m_2)^n$ is idempotent. From $m_1 \cdot m_2 = m_1$, we deduce that $m_1 \cdot e = m_1$. Thus, we have $m_1 \leq_D e \leq_D m_2$. Hence, if m_1 and m_2 are in the same class, then it is regular. It remains to prove that $m_1 \sim_D m_2$.

   Towards a contradiction, let us assume that m_1 and m_2 are not in the same class and thus (as $m_1 \leq_D m_2$) that $m_1 <_D m_2$. We distinguish two cases, depending on the regularity of the class containing m_1.

   If m_1 is in a regular class, let $(D_1, \dots, D_k)$ be a maximal chain of regular classes below m_1, with k such that $h_0(m_1) = 2k - 2$: in particular, D_k is the class of m_1. Let us consider two cases. If the class D of m_2 is regular, then $(D_1, \dots, D_k, D)$ is a chain of regular classes below m_2, showing that $h_0(m_2) \geq 2(k+1)-2 > h_0(m_1)$, which is a contradiction. If m_2 is not in a regular class, then $h_0(m_2) \geq 2k - 1 > h_0(m_1)$ which is again a contradiction.

   Thus, m_1 cannot be in a regular class. Then, we have $m_1 <_D e$ since the idempotent e cannot be in the same class as m_1. As above, from a maximal chain of regular classes below m_1, of length k, we can add the regular class of e and thus get that $h_0(e) \geq 2(k + 1) - 2 > 2k - 1 = h_0(m_1)$. Since $e \leq_D m_2$, from (1), we deduce that $h_0(e) \leq h_0(m_2)$. In the overall, we have that $h_0(m_1) < h_0(m_2)$ which contradicts the hypothesis. □

A new deterministic automaton The labelling $\lambda : [0, n] \to Q$ of a word $u = a_1 \cdots a_n$ is obtained as the run of a deterministic automaton $\mathcal{A} = (Q, q_0, \delta)$. As already recalled, the finite set Q of states consists in H-valid sequences $\langle m_1, \dots, m_k \rangle$ of elements of M. By (H), the height of elements of M lying in $[0, 2L(M) - 2]$, the length of an H-valid sequence is bounded by $2L(M) - 1$. It is also bounded by $|M|$ since all elements of the sequence are different. Hence, there are at most $|M|^{\min(2L(M)-1,|M|)}$ states. This must be compared with the bound $|M|^{|M|}$ originally obtained by Colcombet. For the example of M being the transformation monoid mentioned in the introduction, we thus obtain an exponentially smaller machine.[4] The initial state q_0 can be chosen arbitrarily, a natural choice being the sequence only containing the neutral element of M, having maximal height. The (deterministic) transition function $\delta : Q \times \Sigma \to Q$ is defined as follows: for a valid sequence $\langle m_1, \dots, m_k \rangle$ and a letter $a \in \Sigma$, the unique successor of the sequence by reading a for the transition function is the valid sequence

$$\delta(\langle m_1, \dots, m_k \rangle, a) = \left\langle m_1, \dots, m_i, (m_{i+1} \cdot \cdots \cdot m_k \cdot \varphi(a)) \right\rangle$$

with i chosen maximal. Notice that all singleton sequences are valid, in particular this is the case for $\langle m_1 \cdot \cdots \cdot m_k \cdot \varphi(a) \rangle$, hence the successor is always well-defined.

A *run* of $\mathcal{A}$ on a word $u = a_1 \cdots a_n$ consists in a labelling $\lambda : [0, n] \to Q$ of positions of the word with states such that $\lambda(0) = q_0$ and for all $i \in [1, n]$, $\lambda(i) = \delta(\lambda(i - 1), a_i)$.

Example 3. *For the monoid M_K, the height of all elements was already depicted in the class diagram of Figure 1. The associated automaton is depicted in Figure 3. We remark that with another choice of initial state (like $\langle c \rangle$ or $\langle aa \rangle$), the number of reachable states would drop to 10 states instead of 24.*

The forward Ramsey labelled split obtained on a given input word is depicted in Figure 4: recall that the height is computed solely by considering the last element of the sequence contained in the current state of the run, which is exactly the label we depict in the height drawing. All positions labelled $\langle aa \rangle$ are $\sim_{\lambda,h}$-equivalent. This ensures that $\varphi(ababaa) = aa$ (the word read between the first and second occurrence of label aa) and $\varphi(c) = c$ (the word read between the second and third occurrence of label aa), e.g., are idempotent elements. Notice though that these idempotent elements may be different, as it is the case for this example, but they belong to the same class, as expected. The three last positions of label b are $\sim_{\lambda,h}$-equivalent, but the one before is not equivalent to them, as there is a position

[3] Indeed, if m is an element of a minimal class, since we have $m \geq_D m^2 \geq_D m^3 \geq_D \cdots$ letting $k = |M|!$, we have that m^k is an idempotent element in the same class as m.

[4] Notice that for the example of $M = \{1, 2, \dots, n\}$ with the maximum operation, $L(M) = n$ and thus our automaton will have $|M|^{|M|}$ states, which is thus a worst case.

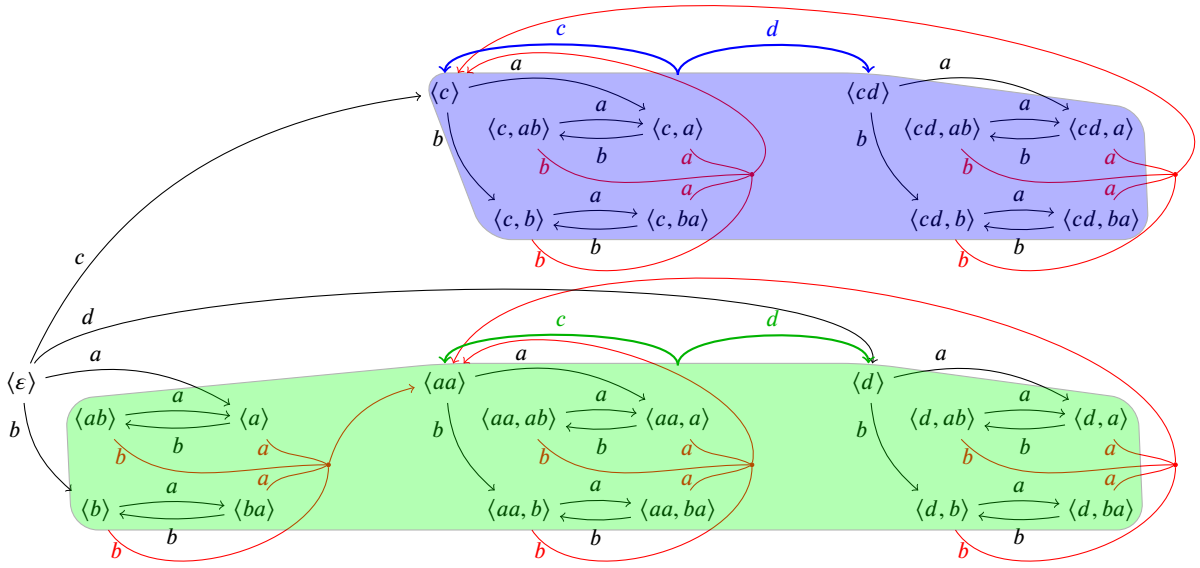

Figure 3: The deterministic automaton associated with M_K. Green transitions, labelled c and d are a shortcut to similar transitions outgoing from all states of the green component. Similarly for the blue transitions. Red transitions are quadruplets of transitions going to the same state.

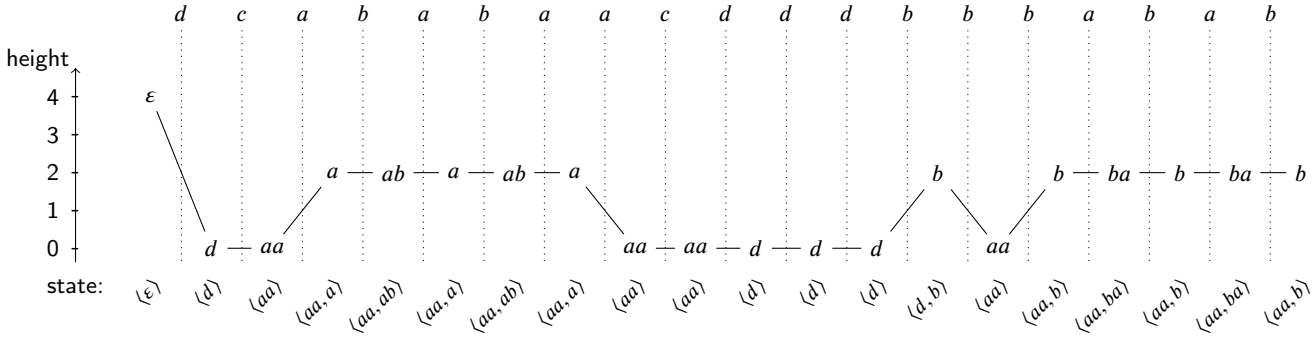

Figure 4: Forward Ramsey labelled split obtained by running the automaton associated with M_K on a word

in-between of lower height. There are equivalent pairs (x, y) and (z, t) of positions that overlap (i.e. $x < z < y < t$), for instance positions labelled with aa and d. In this case, the idempotent elements they delimitate are also in the same class: for instance, $\varphi(cababaacd) = cd$ (in-between the first two occurrences of label d) is equivalent to $\varphi(dddbb) = d$ (in-between the two last occurrences of label aa).

On Figure 5, we depict a forward Ramsey labelled split for an input word with the monoid M_{L_3}, computing the transitions of the automaton on-the-fly.

Remark 4. *Even if (λ, h) is a forward Ramsey labelled split, it does not imply that $h \circ \lambda$ is a forward Ramsey split. For instance, as shown by the previous example in Figure 4, positions 8, 9 and 10 are $\sim_{h\circ\lambda}$-equivalent, but $\varphi(c) \neq \varphi(c) \cdot \varphi(d)$. These three positions are not in the same $\sim_{\lambda,h}$-equivalence class, since $\lambda(9) \neq \lambda(10)$.*

Let us turn to the correctness of our construction: given two states p and q of $\mathcal{A}$, and a word u, we write $p \xrightarrow{u:k} q$ to denote the fact that the automaton $\mathcal{A}$ goes from state p to state q when reading u, and k is the minimal height of a state appearing along the run (not counting the height of p). Formally, when q is the successor of p when reading a in $\mathcal{A}$, then $p \xrightarrow{a:h(q)} q$, and when $p \xrightarrow{u:k} q$ and $q \xrightarrow{v:\ell} r$, then $p \xrightarrow{uv:\min(k,\ell)} r$. The notation $p \xrightarrow{u:k} q$ is extended to $u = \varepsilon$ by taking $q = p$ and $k = +\infty$. The crucial argument allowing

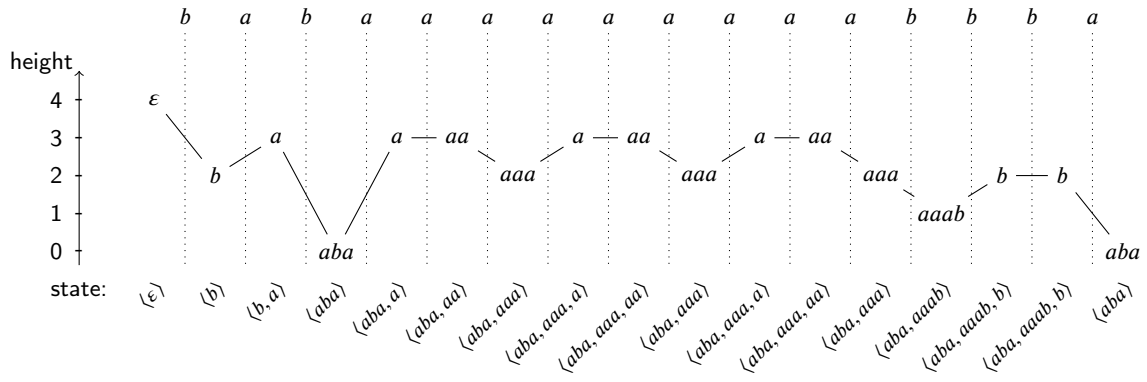

Figure 5: Forward Ramsey labelled split obtained by running the automaton associated with M_{L_3} on a word

Colcombet to show the correctness of his construction is Lemma 5.4 of [4], that we adapt in our case:

Lemma 5. *Let v and w be two valid sequences, m be an element of M, and $u \in \Sigma^+$ be a non-empty word. If $\langle v, m\rangle \xrightarrow{u:h_0(m)} \langle w, m\rangle$, then $m \cdot \varphi(u) = m$ and $m \sim_{\mathcal{D}} \varphi(u)$. Moreover, $v = w$.*

Proof. We prove by induction on the length of u the stronger property that whenever $\langle v, m\rangle \xrightarrow{u:f} \langle w\rangle$ with $f \geq h_0(m)$, then

(1) $\langle w\rangle = \langle v, m_1, \dots, m_k\rangle$ for $k \geq 1$ and $m_1, \dots, m_k \in M$;

(2) $h_0(m_1) = h_0(m)$ and $m_1 \leq_{\mathcal{D}} m$;

(3) $m_1 \cdots m_k = m \cdot \varphi(u)$;

(4) if $k > 1$, there exists a decomposition $u = u_1u_2$ and $f' \leq f$ such that $\langle v, m\rangle \xrightarrow{u_1:f'} \langle v, m_1\rangle$ and $\varphi(u_2) = m_2 \cdots m_k$;

(5) if $u \neq \varepsilon$, $k = 1$ and $m_1 = m$, then $m \sim_{\mathcal{D}} \varphi(u)$.

The result of the lemma holds by (1), (2) that implies that $k = 1$, and then (3) and (5).

The statement trivially holds for $u = \varepsilon$. Assume now (1), …, (5) holds for some non-empty u such that

$$\langle v, m\rangle \xrightarrow{u:f'} \left\langle v, m'_1, \dots, m'_{k'}\right\rangle$$

with $f' \geq h_0(m)$. Let a be a letter such that

$$\left\langle v, m'_1, \dots, m'_{k'}\right\rangle \xrightarrow{a:f} \langle w\rangle$$

for some $f = h(w) \geq h_0(m)$. We aim at establishing the claim for the word ua.

(1) Assume that $v = \langle d_1, \dots, d_i\rangle$ is not a strict prefix of w. Then, by definition of the transition function, there is some $\ell \leq i$ such that $w = \langle d_1, \dots, d_{\ell-1}, d\rangle$ with $d = d_\ell \cdots d_i \cdot m'_1 \cdots m'_{k'} \cdot \varphi(a)$. In particular, this means that $d \leq_{\mathcal{D}} d_i$, so $h_0(d) \leq h_0(d_i)$ by Lemma 2(1). Furthermore, $h_0(d_i) < h_0(m'_1)$ since $\left\langle d_1, \dots, d_i, m'_1, \dots, m'_{k'}\right\rangle$ is a valid sequence. Using (2) of the induction hypothesis, $h_0(m'_1) = h_0(m)$. So $h_0(d) < h_0(m)$ which contradicts the hypothesis that $f = h(w) \geq h_0(m)$. Hence, v is a strict prefix of w, and thus (1) holds: let $m_1, \dots, m_k$ be such that

$$\left\langle v, m'_1, \dots, m'_{k'}\right\rangle \xrightarrow{a:f} \langle w\rangle = \langle v, m_1, \dots, m_k\rangle$$

(2) According to the definition of $\xrightarrow{a:f}$, two cases can happen. Either $k > 1$ and by definition of the transition function $m_1 = m'_1$. We directly obtain $h_0(m_1) = h_0(m'_1) = h_0(m)$, by the induction hypothesis. Otherwise, $k = 1$ and we have $m_1 = m'_1 \cdots m'_{k'} \cdot \varphi(a)$ and $f = h_0(m_1)$. This implies $m_1 \leq_{\mathcal{D}} m'_1$, hence, by Lemma 2(1), $f = h_0(m_1) \leq h_0(m'_1) = h_0(m)$. Moreover, by hypothesis, $f \geq h_0(m)$ so that $h_0(m_1) = h_0(m)$. By induction, we also have $m'_1 \leq_{\mathcal{D}} m$. Overall, (2) holds.

(3) By (3) of the induction hypothesis, $m'_1 \cdots m'_{k'} = m \cdot \varphi(u)$. By definition of the transition function of $\mathcal{A}$, we obtain $m_1 \cdots m_k = m'_1 \cdots m'_{k'} \cdot \varphi(a) = m \cdot \varphi(u) \cdot \varphi(a) = m \cdot \varphi(ua)$. Hence (3) holds.

(4) Assume that $k > 1$. Two cases can happen. If $k' = 1$, then $k = 2$, $m_1 = m'_1$, and $m_2 = \varphi(a)$. It follows that $ua = u_1u_2$ with $u_1 = u$ and $u_2 = a$ is a witness for (4). Otherwise $k' > 1$ (and thus $m_1 = m'_1$). Let $u = u_1u_2$ be the witness decomposition of (4) obtained by the induction hypothesis. Using the definition of the transition function we obtain $\varphi(u_2a) = \varphi(u_2) \cdot \varphi(a) = m'_2 \cdots m'_{k'} \cdot \varphi(a) = m_2 \cdots m_k$. Therefore, the decomposition $ua = u'_1u'_2$ with $u'_1 = u_1$ and $u'_2 = u_2a$ is a witness for (4).

(5) Finally, assume $k = 1$ and $m_1 = m$. Then, letting $d = m'_2 \cdots m'_{k'} \cdot \varphi(a)$, we have $m'_1 \cdot d = m_1 = m$.

By definition of the transition function, $\langle v, m_1', d\rangle$ is not a valid sequence while $\langle v, m_1' \cdot d\rangle$ is valid. The reason why $\langle v, m_1', d\rangle$ is not valid is that $h_0(d) \leq h_0(m_1') = h_0(m)$. As $m_1' \cdot d = m$, we have $d \geq_{\mathcal{D}} m$, thus $h_0(d) \geq h_0(m)$ by Lemma 2(1). Therefore $h_0(d) = h_0(m)$. Using (4) of the induction hypothesis for u, we get:

$$\langle v, m\rangle \xrightarrow{u_1 : f_1} \langle v, m_1'\rangle \xrightarrow{u_2 a : f_2} \langle v, m\rangle$$

We have

$$\begin{aligned} h_0(\varphi(ua)) &= h_0(\varphi(u_1)\cdot\varphi(u_2)\cdot\varphi(a)) \\ &= h_0(\varphi(u_1)\cdot m_2'\cdots m_{k'}'\cdot\varphi(a)) \\ &= h_0(\varphi(u_1)\cdot d) \leq h_0(d) \\ &\qquad\qquad \text{(since } \varphi(u_1)\cdot d \leq_{\mathcal{D}} d) \\ h_0(\varphi(ua)) &\leq h_0(m) \end{aligned}$$

By (3), $m_1 = m = m \cdot \varphi(ua)$. Thus, by Lemma 2(1), $h_0(\varphi(ua)) \geq h_0(m)$, and we obtain that $h_0(\varphi(ua)) = h_0(m)$. Using Lemma 2(2) with $m_1 = m$ and $m_2 = \varphi(ua)$, we obtain (5). □

We are finally able to show that $\mathcal{A}$ and h define a forward Ramsey labelled split for any word u.

Proposition 6. *For all finite monoids M, let h and $\mathcal{A}$ be respectively the height function and the deterministic automaton previously defined. For all words $u \in \Sigma^+$, let λ be the unique run of $\mathcal{A}$ on u. Then, the labelled split (λ, h) is forward Ramsey.*

Proof. Denoting n the length of u, let $x < y$ and $x' < y'$ be positions in $[0, n]$ belonging to the same $\sim_{\lambda,h}$-equivalence class. Since they are $\sim_{\lambda,h}$-equivalent, $\lambda(x) = \lambda(y) = \lambda(x') = \lambda(y')$ is a sequence that ends with the same element m of M, and the height of labels along the run in-between positions $\min(x, x')$ and $\max(y, y')$ is at least $h_0(m)$. By Lemma 5, we deduce that $m \cdot \varphi(u_{x,y}) = m$, $m \sim_{\mathcal{D}} \varphi(u_{x,y})$, $m \cdot \varphi(u_{x',y'}) = m$ (and $m \sim_{\mathcal{D}} \varphi(u_{x',y'})$, which we do not use). Lemma 5.3 of [4] states that for all $(m, m', m'') \in M^3$ such that $m \cdot m' = m \cdot m'' = m$ and $m \sim_{\mathcal{D}} m'$, it holds that $m' \cdot m'' = m'$. Applying it with $m' = \varphi(u_{x,y})$ and $m'' = \varphi(u_{x',y'})$, we obtain $\varphi(u_{x,y}) \cdot \varphi(u_{x',y'}) = m' \cdot m'' = m' = \varphi(u_{x,y})$ as expected. □

Remark 7. *We notice that if we use the height function of Colcombet defined at (1), mapping a different height to all elements of the monoid, but keep our smaller automaton, we can obtain a forward Ramsey split (and not labelled split). Indeed, this more refined height would still allow us to obtain Lemma 5, and thus Proposition 6 with a forward Ramsey split.*

Alternative validity conditions We enforced the finiteness of the automaton $\mathcal{A}$ by limiting ourselves to H-valid sequences of elements in M. As already mentioned, in [4], Colcombet uses CP-validity.

We now consider the other alternative definitions of validity, and the automata $\mathcal{A}$ they induce, when combining the three properties H, C, and P. We consider combinations $\{H, C\}$, and $\{H, C, P\}$. The combination $\{H, P\}$ is identical to the latter one, since (H) and (P) together imply (C). Depending on the case, the obtained automaton could be smaller for some validity conditions or another. We anyway show that the alternative definitions would also give rise to a forward Ramsey labelled split.

The only place where the specific definition of validity matters is in the proof of (5) in the induction proof of Lemma 5, when the sequence $\langle v, m_1, d\rangle$ is not valid but $\langle v, m_1 \cdot d\rangle$ is. We thus reprove (5) of Lemma 5 for the two alternative definitions of valid sequences, using the very same notations as in the original proof.

Proof (with $\{H, C\}$). Assume $n = 1$ and $m_1' = m$. Then, letting $d = m_2 \cdots m_k \cdot \varphi(a)$, we have $m_1 \cdot d = m_1' = m$. By definition of the transition function, $\langle v, m_1, d\rangle$ is not a valid sequence while $\langle v, m_1 \cdot d\rangle$ is valid. We only prove that $h_0(d) = h_0(m)$, the rest of the proof being the same. Suppose that $\langle v, m_1, d\rangle$ does not satisfy (C), then $m_1 \not\sim_{\mathcal{D}} d$. Meanwhile, by (2), $m_1 \sim_{\mathcal{D}} m = m_1 \cdot d$, hence $m_1 \leq_{\mathcal{D}} d$. So, $m_1 \sim_{\mathcal{D}} d \sim_{\mathcal{D}} m$ thus $h_0(m_1) = h_0(d) = h_0(m)$. Otherwise, suppose that $\langle v, m_1, d\rangle$ does not satisfy (H), then as in the original proof, $h_0(d) \leq h_0(m_1) = h_0(m)$. As $m_1 \cdot d = m$, we have $d \geq_{\mathcal{D}} m$ thus $h_0(d) \geq h_0(m)$, so that $h_0(d) = h_0(m)$. □

Proof (with $\{H, C, P\}$). Assume $n = 1$ and $m_1' = m$. Then, letting $d = m_2 \cdots m_k \cdot \varphi(a)$, we have $m_1 \cdot d = m_1' = m$. By definition of the transition function, $\langle v, m_1, d\rangle$ is not a valid sequence while $\langle v, m_1 \cdot d\rangle$ is valid. Also, $m_1 \cdot d = m$ thus $m_1 \geq_{\mathcal{D}} m$. By (2), $m_1 \leq_{\mathcal{D}} m$, thus $m \sim_{\mathcal{D}} m_1$. Hence $\langle v, m_1, d\rangle$ satisfies (P). Hence, $\langle v, m_1, d\rangle$ does not satisfy either (C) or (H). The rest of the proof is the same as for $\{H, C\}$. □

Computing the labelled split of a word We are now interested in understanding what is the time complexity for computing the pair (λ, h), with respect to M and u. For that, we suppose that products in M are performed in time given by some parameter T_M (that we may consider polylogarithmic in the size of M). We suppose also that the lattice of $\mathcal{D}$-classes of $\varphi(\Sigma^*)$ has been precomputed (in time linear in $|M|$ by using the bi-Cayley graph of $\varphi(\Sigma^*)$, the submonoid of M generated by $\varphi(\Sigma)$). The height $h_0(m)$ of each element $m \in M$ can be obtained from the bi-Cayley graph by computing its strongly connected components as well as the direct acyclic graph relating them, in linear time in $|M|$: we thus now suppose that $h_0(m)$ has been precomputed and thus can be questionned in constant time. We only present the computation when considering the H-validity: it would be entirely similar with the two other validity conditions.

First, notice that when computing the labelled split of a word u, the automaton does not need to be entirely computed a priori, but may rather be computed on-the-fly with a possible storage of already computed values. Given a state $\langle m_1, \ldots, m_k\rangle$, and a letter a, the successor state can be

computed by iteratively checking validity of sequences of the form $\langle m_1, \dots, m_i, (m_{i+1} \cdots m_k \cdot \varphi(a)) \rangle$ for decreasing i: for each step i, this costs T_M to compute the product (since the comparison of heights is then supposed to be performed in constant time). Considering amortised complexity, each element that is added to the sequence is later aggregated at most once during a validity check. Overall, the complexity is thus of the form $n\,T_M$, where n is the size of the input word.

3. Fast infix evaluation

In this last section, we consider an application of the previous construction, proposed in [4, 9]: given a regular language L and some fixed word u, an L-infix query in u is a query of the form "given positions $i \leq j$ in u, does $u_{i,j}$ belong to L?". After a precomputation that must be linear in the length n of u, the goal is to be able to answer any infix query in constant time (with respect to n).

In [9], Bojańczyk answers positively to this goal, in the case where L is defined by a morphism $\varphi\colon \Sigma^* \to M$ of monoids: L is then the preimage $\varphi^{-1}(A)$ of a subset A of the finite monoid M. To do so, he precomputes a factorisation forest for the word u, in time linear in $|M|$ and n, then allowing queries to be answered in time linear in $|M|$. By computing forward Ramsey splits as in [4], one would also obtain an algorithm answering queries in time linear in $|M|$. The accelerating pointers techniques developed for non-deterministic automata in [10] could allow one to obtain a $\mathcal{O}(T_M \log_2 |M|)$ complexity. The precomputation time would be the same as the one described in the end of the previous section, i.e. $n\,T_M$.

We present a new algorithm for solving the L-infix query problem. The precomputation consists in building a forward Ramsey labelled split for u as well as some additional information. The query answering can then be performed in time logarithmic in $L(M)$, leading to an improvement of one or two exponentials with respect to previous results, depending on M.

A new precomputation Consider a monoid M and a word $u = a_1 \cdots a_n$. First, the forward Ramsey labelled split (λ, h) for the word u is computed as explained in Section 2. Before moving to the query, we need to compute additional information, inspired by the accelerating pointers of [10], as follows.

- A suffix operator

$$\mathsf{suff}\colon [0,n] \times M \to ([0,n] \times M) \cup \{\bot\}$$

that associates with each pair (i, m) of position and monoid element, the pair (j, m') composed of the smallest position $j > i$ labelled with a sequence ending with m, and the value $m' = \varphi(u_{i,j})$. When there is no such position j, we let $\mathsf{suff}(i, c)$ be the dummy element $\bot$. We can compute suff efficiently, in a single pass of the word from right to left. For all $m \in M$, we let $\mathsf{suff}(n, m) = \bot$, and for all $0 \leq i \leq n-1$, $\mathsf{suff}(i, m)$ being equal to

$$\begin{cases} (i+1, \varphi(a_i)) & \text{if } \lambda(i+1) \text{ ends with } m \\ \bot & \text{otherwise, if } \mathsf{suff}(i{+}1, m) = \bot \\ (j, \varphi(a_i) \cdot m') & \text{otherwise, if } \mathsf{suff}(i{+}1, m) = (j, m') \end{cases}$$

- A prefix operator

$$\mathsf{pre}\colon [0,n] \times \{0, \dots, 2\,L(M){-}2\} \to ([0,n] \times M) \cup \{\bot\}$$

that associates with each pair (i, ℓ) of position and height, the pair (j, m') composed of the greatest position $j < i$ with a height at most ℓ, and the value $m' = \varphi(u_{j,i})$. When there is no such position j, we let $\mathsf{pre}(i, c)$ be the dummy element $\bot$. As for suff, pre can be efficiently computed, in a single pass from left to right. For all $\ell \in \{0, \dots, 2L(M) - 2\}$, we let $\mathsf{pre}(0, \ell) = \bot$, and for all $1 \leq i \leq n$, $\mathsf{pre}(i, \ell)$ being equal to

$$\begin{cases} (i-1, \varphi(a_i)) & \text{if } h{\circ}\lambda(i-1) \leq \ell \\ \bot & \text{otherwise, if } \mathsf{pre}(i{-}1, \ell) = \bot \\ (j, m' \cdot \varphi(a_i)) & \text{otherwise, if } \mathsf{pre}(i{-}1, \ell) = (j, m') \end{cases}$$

Computing the forward Ramsey labelled split (λ, h), as explained in Section 2, takes a time complexity $n\,T_M$. Moreover, one step in the computation of suff and pre requires a test of pairs in $[0, n] \times M$ and a product in M, thus a time of $\log_2(n|M|) + T_M$. Thus, computing the mappings suff or pre requires total time

$$\mathcal{O}\left(n\,|M|\,(\log_2(n|M|) + T_M)\right)$$

that is the overall complexity of the precomputation. Notice that this time is sufficient to also precompute the lattice of $\mathcal{D}$-classes of M, required at the beginning of the precomputation.

A new algorithm to answer L-infix queries Using the forward Ramsey labelled split (λ, h), as well as operators suff and pre, L-infix queries can be answered efficiently as follows. Consider a pair (i, j) of positions of u, the evaluation of $\varphi(u_{i,j})$ is performed by the three following steps, graphically summarised in Figure 6.

1. Among the positions of $\{i, \dots, j-1\}$ with minimal height ℓ (which we can compute by using a binary search, as we explain afterwards), find k_1 the maximal position. Notice that k_1 can be computed as $\mathsf{pre}(j, \ell) = (k_1, m_1)$, and that $k_1 \in \{i, \dots, j-1\}$ by definition of ℓ.
2. If $k_1 = i$, the query can be directly answered: $\varphi(u_{i,j}) = m_1$. Otherwise ($k_1 > i$), we let m be the last element of $\lambda(k_1)$, and we compute $(k_2, m_2) = \mathsf{suff}(i, m)$. By construction $h{\circ}\lambda(k_2) = h{\circ}\lambda(k_1) = h_0(m)$ and all states between k_2 and k_1 have a greater or equal height; thus by Lemma 5, we obtain the much stronger

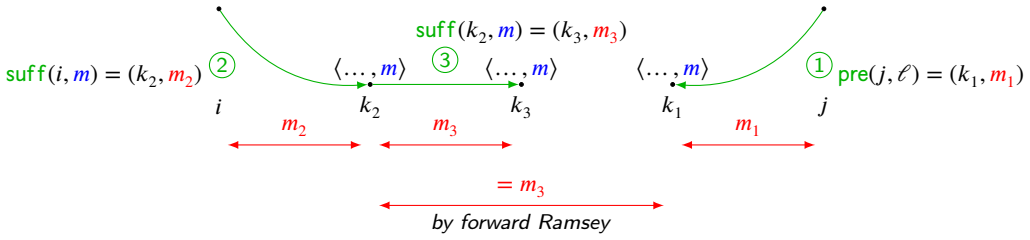

Figure 6: Infix evaluation

property that $\lambda(k_2) = \lambda(k_1)$: not only the last elements of the two sequences are equal, the whole sequences are equal.

3. If $k_1 = k_2$, then, the query can be answered since $\varphi(u_{i,j}) = \varphi(u_{i,k_2}) \cdot \varphi(u_{k_1,j}) = m_2 \cdot m_1$. Otherwise ($k_2 < k_1$), we let $(k_3, m_3) = \mathsf{suff}(k_2, m)$. The same remark as before leads to $\lambda(k_1) = \lambda(k_2) = \lambda(k_3)$. By construction, k_1, k_2, k_3 are thus $\sim_{\lambda,h}$-equivalent. Using the forward Ramsey property, $\varphi(u_{k_2,k_3}) = \varphi(u_{k_2,k_3}) \cdot \varphi(u_{k_3,k_1}) = m_2$, thus, the query can be answered: $\varphi(u_{i,j}) = \varphi(u_{i,k_2}) \cdot \varphi(u_{k_2,k_3}) \cdot \varphi(u_{k_3,k_1}) \cdot \varphi(u_{k_1,j}) = m_2 \cdot m_3 \cdot m_1$.

The only operation in these steps requiring non-constant complexity is the search of the height ℓ, in order to find k_1. An immediate solution consists in inspecting

$$\mathsf{pre}(j, 0), \mathsf{pre}(j, 1), \ldots, \mathsf{pre}(j, 2L(M) - 2)$$

sequentially, stopping the first time we get a pair with a first component greater than or equal to i, the searched height ℓ being the corresponding index. This would require a complexity linear in $L(M)$ in the worst case. A more clever solution consists in searching for ℓ with a binary search. Indeed, letting i_z the first component of $\mathsf{pre}(j, z)$ for all $z \in \{0, 1, \ldots, 2L(M) - 2\}$ if this is a pair, and -1 otherwise, we notice that $i_0 \leq \ldots \leq i_{2L(M)-2}$ since every position with a height at most z has also a height at most $z + 1$. We can therefore look for the leftmost index of a position greater than or equal to i in this ordered list, in time $\log_2(2L(M) - 1)$. In summary, we have obtained a query evaluated in complexity $\mathcal{O}\left(T_M \log_2(L(M))\right)$.

4. Conclusion

We introduced forward Ramsey labelled splits, a weaker notion than forward Ramsey splits. We have explained how they can be computed by deterministic automata, taking profit of the regular $\mathcal{D}$-length to obtain smaller automata than for forward Ramsey splits. As future works, it would be interesting to find a variant of our construction with a parameter tractable complexity, i.e. with a number of states of the form $f(L(M))\,\mathrm{poly}(|M|)$ (compared with $|M|^{2L(M)-1}$ in our current construction). We also plan to study some other tasks on words that can be answered by studying forward Ramsey labelled splits, instead of forward Ramsey splits or factorisation forests.

Acknowledgements

This work was supported by the ANR projects DeLTA (ANR-16-CE40-0007) and Ticktac (ANR-18-CE40-0015).

References

[1] I. Simon, Factorization forests of finite height, Theor. Comput. Sci. 72 (1990) 65–94. doi:10.1016/0304-3975(90)90047-L.

[2] T. Colcombet, A combinatorial theorem for trees, in: Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Vol. 4596 of Lecture Notes in Computer Science, 2007, pp. 901–912. doi:10.1007/978-3-540-73420-8_77.

[3] P. Gastin, S. N. Krishna, Unambiguous forest factorizations, Research Report 1810.07285, arXiv (2018).

[4] T. Colcombet, The factorisation forest theorem, in: J. Pin (Ed.), Handbook of Automata Theory, European Mathematical Society Publishing House, Zürich, Switzerland, 2021, pp. 653–693. doi:10.4171/Automata-1/18.

[5] I. Jecker, A Ramsey theorem for finite monoids, in: 38th International Symposium on Theoretical Aspects of Computer Science (STACS), Vol. 187, 2021, pp. 44:1–44:13. doi:10.4230/LIPIcs.STACS.2021.44.

[6] O. Ganyushkin, V. Mazorchuk, Classical finite transformation semigroups, an introduction, Springer-Verlag London, 2009. doi:10.1007/978-1-84800-281-4.

[7] W. Kazana, L. Segoufin, Enumeration of monadic second-order queries on trees, ACM Trans. Comput. Log. 14 (4) (2013) 25:1–25:12. doi:10.1145/2528928.

[8] M. Niewerth, L. Segoufin, Enumeration of MSO queries on strings with constant delay and logarithmic updates, in: Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, 2018, pp. 179–191. doi:10.1145/3196959.3196961.

[9] M. Bojańczyk, Factorization forests, in: Developments in Language Theory, 13th International Conference, DLT 2009, 2009, pp. 1–17. doi:10.1007/978-3-642-02737-6_1.

[10] M. Bojańczyk, P. Parys, Efficient evaluation of nondeterministic automata using factorization forests, in: Automata, Languages and Programming, Vol. 6198 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 515–526. doi:10.1007/978-3-642-14165-2_44.

[11] T. Colcombet, Green's relations and their use in automata theory, in: A.-H. Dediu, S. Inenaga, C. Martín-Vide (Eds.), Language and Automata Theory and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 1–21.