International Journal of Algebra and Computation
(2016)
© World Scientific Publishing Company
DOI: 10.1142/S0218196716500363

World Scientific
www.worldscientific.com

Solution sets for equations over free groups are EDT0L languages

Laura Ciobanu

Institut de mathématiques
Université de Neuchâtel
Rue Emile-Argand 11, CH-2000 Neuchâtel, Switzerland
laura.ciobanu@unine.ch

Volker Diekert

Institut für Formale Methoden der Informatik
Universität Stuttgart
Universitätsstr. 38, D-70569 Stuttgart, Germany
diekert@fmi.uni-stuttgart.de

Murray Elder

School of Mathematical & Physical Sciences
The University of Newcastle
Callaghan, New South Wales 2308, Australia
murray.elder@newcastle.edu.au

Received 8 August 2015
Accepted 17 May 2016
Published

Communicated by O. Kharlampovich

We show that, given an equation over a finitely generated free group, the set of all solutions in reduced words forms an effectively constructible EDT0L language. In particular, the set of all solutions in reduced words is an indexed language in the sense of Aho. The language characterization we give, as well as further questions about the existence or finiteness of solutions, follow from our explicit construction of a finite directed graph which encodes all the solutions. Our result incorporates the recently invented recompression technique of Jeż, and a new way to integrate solutions of linear Diophantine equations into the process. As a byproduct of our techniques, we improve the complexity from quadratic nondeterministic space in previous works to $\mathsf{NSPACE}(n \log n)$ here.

Keywords: Equation in a free group; EDT0L language; indexed language; compression; free monoid with involution.

Mathematics Subject Classification: 03D05, 20F65, 20F70, 68Q25, 68Q45

0. Introduction

In this paper, we prove that the set of all solutions, as reduced words, to an equation in a finitely generated free group or free monoid with involution, has a description as an EDT0L language. Furthermore, we show that this description can be computed in $\mathsf{NSPACE}(n \log n)$, where n is the length of the equation plus the number of generators of the group or monoid.

We construct a finite graph, of singly exponential size $2^{\mathcal{O}(n \log n)}$, with nodes labeled by equations of bounded size plus some additional data, and directed edges corresponding to transformations applied to the equations. More precisely, the edges are labeled by endomorphisms of a free monoid C^*, where C is a finite alphabet which includes the group or monoid generators. The graph, viewed as a nondeterministic finite automaton, produces a rational language of endomorphisms of C^*. We show that the set of all such endomorphisms applied to a particular "seed" word gives the full set of solutions to the input equation as reduced words. Thus, by the definition of Asveld [2], we obtain that the solution set is an EDT0L language, and therefore an indexed language. Moreover, one can decide if there are zero, infinitely or finitely many solutions simply by checking if the graph is empty, has directed cycles or not. Our complexity results concerning these decision problems are the best known so far; and with respect to space complexity they might be optimal.

The first algorithmic description of all solutions to a given equation over a free group is due to Razborov [20, 21]. His description became known as a *Makanin–Razborov diagram*, and this concept plays a major role in the positive solution of Tarski's conjectures about the elementary theory in free groups [14, 24]. While Makanin–Razborov diagrams are also graphs whose edges are labeled by morphisms, these morphisms are group homomorphisms, and it is unfeasible to use this approach to directly obtain solutions in freely reduced words, as the cancellation within group elements after applying a homomorphism cannot be controlled. Also, it is extremely complicated to explicitly produce a Makanin–Razborov diagram for a given equation, and this has been done only in very few cases ([25]).

A description of solution sets as EDT0L languages was known before only for quadratic word equations over a free monoid by [10]; the recent paper [6] did not aim at giving such a structural result. The present paper builds on the techniques in [6], in particular we make use of Jeż's *recompression* method [12]. There is also a description of all solutions for a word equation over free monoids by Plandowski in [19]. His description is given by some graph which can be computed in singly exponential time, but without the aim to give any formal language characterization.

In this paper, we restrict ourselves to equations in free groups or free monoids with involution, and their solution sets in reduced words. It is possible to generalize our construction in several directions. First, we can replace the free group by any finitely generated free product $\mathsf{P} = \star_{1 \le i \le s} F_i$, where each F_i is either a free or finite group, or a free monoid with arbitrary involutions. Second, we can allow arbitrary rational constraints for free products. We consider Boolean formulae Φ, where each atomic formula is either an equation or a *rational constraint*, written as $X \in L$, where $L \subseteq \mathsf{P}$ is a rational subset. More concretely, let P be a free product as above, Φ a Boolean formula over equations and rational constraints, and $\{X_1, \dots, X_k\}$ any subset of variables. Then the techniques developed in this paper allow us to prove that $\mathrm{Sol}(\Phi) = \{\sigma(X_1)\# \cdots \#\sigma(X_k) \mid \sigma \text{ solves } \Phi \text{ in reduced words}\}$ is EDT0L. Moreover, there is an algorithm which takes Φ as input and produces an NFA $\mathcal{A}$ such that $\mathrm{Sol}(\Phi) = \{\varphi(\#) \mid \varphi \in L(\mathcal{A})\}$. The algorithm is nondeterministic and uses quasi-linear space in the input size of Φ. However, these more technical results are not the scope of the present paper. They follow from standard results in the literature and they have been announced in the conference version of this paper which was presented at ICALP 2015, Kyoto (Japan), 4–10 July 2015 [3]. Full proofs are in the corresponding paper on arXiv [4].

0.1. *Article organization*

In Sec. 1, we give preliminary definitions and notations. In Sec. 2, we state the main result, Theorem 2.1, that solutions in reduced words to equations in either a free group or a free monoid with involution are described by a finite graph or nondeterministic finite automaton (NFA) which can be constructed in nondeterministic quasi-linear space. The main work of the paper is in Sec. 3 which treats the monoid case. We define the NFA in Sec. 3.6, and present the proofs that the NFA encodes only correct solutions (soundness), and all solutions (completeness), in Secs. 3.9 and 3.10, respectively. The most complicated part is the completeness proof, which involves producing a path for a given solution from initial to final node by alternatively expanding and compressing the equation, ensuring that at all times the size of the equation is bounded so that we stay within the graph.

Once the monoid case is proved, in Sec. 4 we follow relatively standard methods to reduce the problem of finding solutions in reduced words in a free group to the monoid case. In the final section, we give an explicit example of the alternating expansion-compression procedure.

We stress that the complicated part of the paper is to prove that the NFA we construct encodes exactly all solutions; the specification and construction of the NFA, and hence the EDT0L language description, is extremely simple by contrast.

1. Preliminaries

1.1. *Monoids with involution*

An *alphabet* is a finite set whose elements are called *letters*. By Γ^* we denote the free monoid over the finite set Γ. The elements of a free monoid are called *words*, and the empty word is denoted by 1. The length of a word w is denoted by $|w|$, and $|w|_x$ counts how often a symbol x appears in w. Let M be any monoid and $u, v \in M$. We write $u \leq v$ if u is a *factor* of v, which means we can write $v = xuy$ for some $x, y \in M$. We denote the neutral element in M by 1, and use the notation id_{C^*} for the neutral element in the monoid of endomorphisms over a free monoid C^*.

An *involution* on a set Γ is a mapping $x \mapsto \overline{x}$ such that $\overline{\overline{x}} = x$ for all $x \in \Gamma$. For example, the identity map is an involution. An *involution on a monoid* must also satisfy $\overline{xy} = \overline{y}\,\overline{x}$. Any involution on a set Γ extends to Γ^*: for a word $w = a_1 \cdots a_m$ we let $\overline{w} = \overline{a_m} \cdots \overline{a_1}$; then Γ^* endowed with the involution is called a *free monoid with involution*. If $\overline{a} = a$ for all $a \in \Gamma$ then $\overline{w}$ is simply the word w read from right-to-left.

A *morphism* between sets with involution is a mapping respecting the involution, and a morphism between monoids with involution is a homomorphism $\varphi : M \to N$ such that $\varphi(\overline{x}) = \overline{\varphi(x)}$. A morphism is a Δ-*morphism* if $\varphi(x) = x$ for all $x \in \Delta$ where $\Delta \subseteq M$. In this paper, whenever the term "morphism" is used, it refers to a mapping which respects the underlying structure, including the involution. All groups are monoids with involution given by $\overline{x} = x^{-1}$; and all group homomorphisms are morphisms.

1.2. *Free partially commutative monoids*

Let Δ be a finite set with involution. An *independence relation* is an irreflexive relation $\theta \subseteq \Delta \times \Delta$ such that $(x, y) \in \theta \Leftrightarrow (\overline{x}, \overline{y}) \in \theta$. Every independence relation defines a *free partially commutative monoid with involution* $M(\Delta, \theta)$ by

$$M(\Delta, \theta) = \Delta^* / \{xy = yx \,|\, (x, y) \in \theta\}.$$

These monoids are well-studied in computer science as they form the basic algebraic model for concurrency, see [7, 13, 15]. In mathematics free partially commutative groups are commonly referred to as right-angled Artin groups (RAAGs). Their study has a long history with strong connections to topology and geometric group theory, see for example [26].

In this paper, we will need algorithms for equality and factor testing in free partially commutative monoids. This can be done very efficiently: for example, there is a linear time algorithm ([16]) to decide on input $u, w \in \Delta^*$ whether $u \leq w$ in $M(\Delta, \theta)$. Here we need the uniform version, as follows: the input is a tuple (Δ, θ, u, w) with $u, w \in \Delta^*$, and the question is whether u is a factor of w in $M(\Delta, \theta)$. This problem can easily be solved in nondeterministic linear space (which suffices for our purposes) by the following argument: first find words $p, q \in \Delta^*$ by scanning w from left to right and for each position guessing (nondeterministically) whether each corresponding letter belongs to p, u or q, requiring that $|puq| = |w|$ (we do this by marking each letter of the input, which requires linear space). Second, check that the choice of positions assigned to u produces a word that is indeed equal to u. Third, check whether puq is equal to w in $M(\Delta, \theta)$. For both the second and third steps we use the "projection lemma" of [13, 5]: for example, in the third step we check that $|puq|_a = |w|_a$ for all $a \in \Delta$, then we check that the projections of puq and w to $\{a, b\}^*$ yield identical words for all $a, b \in \Delta$ such that $ab \neq ba$ in $M(\Delta, \theta)$. The projections are obtained by ignoring all letters in puq and w which are not in $\{a, b\}$.

Another fact about partially commutative monoids that we use later is that for $u \in M(\Delta, \theta)$ the values $|u|$ and $|u|_a$ are well-defined since $|xy|_a = |yx|_a$ for all $x, y \in \Delta^*, a \in \Delta$.

We will define free partially commutative monoids through "types" in Sec. 3.3, which for simplicity of notation are also denoted by θ.

1.3. *Languages*

Languages refer traditionally to subsets of finitely generated free monoids; the class of *regular languages* can be defined via rational expressions, nondeterministic finite automata, or recognizability via homomorphisms to finite monoids, to mention just a few of the possible definitions [18]. These notions generalize to arbitrary monoids, but lead to different classes, in general.

We define a *rational subset* in any monoid M by means of *nondeterministic finite automaton* (NFA). An NFA is a directed finite graph $\mathcal{A}$ with initial and final *states*, where the transitions between states are labeled by elements of the monoid M. We say that $m \in M$ is *accepted* by the automaton $\mathcal{A}$ if there exists a path from some initial to some final state such that multiplying the edge labels together in M yields m. This defines the accepted language $L(\mathcal{A}) = \{m \in M \,|\, m \text{ is accepted by } \mathcal{A}\}$. Then $L \subseteq M$ is *rational* if and only if L is accepted by some NFA over M (see [9]). An NFA is called *trim* if every state is on some path from an initial to a final state. For a trim NFA $\mathcal{A}$ we have $L(\mathcal{A}) \neq \emptyset$ if and only if $\mathcal{A} \neq \emptyset$.

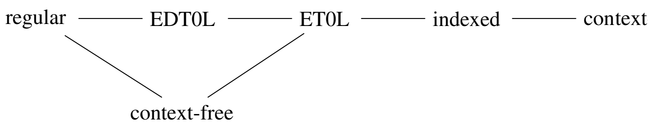
We say that $L \subseteq M$ is *recognizable* if there is a homomorphism $\nu : M \to N$ to a finite monoid N such that $L = \nu^{-1}(\nu(L))$. The family of recognizable subsets is closed under finite union and complementation (and therefore also under finite intersection), and therefore forms a Boolean algebra. For finitely generated free monoids Kleene's Theorem asserts that a subset is recognizable if and only if it is *rational*; and in this context a rational subset is also called *regular*.

In this paper, we are mainly interested in rational subsets of free groups $F(A_+)$, free monoids A^*, and monoids $\mathrm{End}(C^*)$ of endomorphisms over a free monoid C^*. If $|C| \geq 2$, then $\mathrm{End}(C^*)$ is neither free nor finitely generated and it contains nontrivial finite subgroups.

Suppose we have an NFA where each transition label is an endomorphism in $\mathrm{End}(C^*)$ which is applied in the opposite direction of the transition. If a path is labeled by the sequence $h_1, \ldots, h_t$, then we can apply the endomorphism $h = h_1 \cdots h_t$ to an element $u \in C^*$ and the result is a word $h(u) = h_1 \cdots h_t(u) \in C^*$. Thus, $\{h(u) \,|\, h \in L(\mathcal{A})\}$ defines a language in C^*. This leads to the notion of EDT0L, defined next.

1.3.1. *EDT0L Languages*

The acronym EDT0L refers to ***E****xtended,* ***D****eterministic,* ***T****able,* ***0*** *interaction, and* ***L****indenmayer*. There is a vast literature on Lindenmayer systems, see [22], with various acronyms such as D0L, DT0L, ET0L, HDT0L and so forth. For more



AQ: Please provide the citation for Figs. 1 and 3.

Fig. 1. Containments of formal language classes. Each edge from left to right represents strict containment.

background on Lindenmayer systems we refer to [23]. The subclass EDT0L is equal to HDT0L (see for example [23, Theorem 2.6]), and has received particular attention. It is a subclass of indexed languages in the sense of Aho [1], see for example [8]. Indexed languages are context-sensitive, and they strictly contain all context-free languages. The classes of EDT0L and context-free languages are incomparable [8] and therefore the inclusion of EDT0L into indexed languages is proper.

We define EDT0L languages in A^* through a characterization (using rational control) due to Asveld [2], which is the analogue of Ginsburg and Rozenberg's result for ET0L languages ([11, Lemma 4.1]). We start with some alphabet C such that $A \subseteq C$, and a rational set of endomorphisms $\mathcal{R} \subseteq \mathrm{End}(C^*)$. Note that if $\mathcal{R} \subseteq \mathrm{End}(C^*)$ is any subset of endomorphisms, then we can apply $\mathcal{R}$ to any word $u \in C^*$ and we obtain a subset $\{h(u) \mid h \in \mathcal{R}\} \subseteq C^*$.

Definition 1.1. Let A be an alphabet and $L \subseteq A^*$. We say that L is an EDT0L language if there is an alphabet C with $A \subseteq C$, a rational set of endomorphisms $\mathcal{R} \subseteq \mathrm{End}(C^*)$, and a letter $c \in C$ such that $L = \{h(c) \mid h \in \mathcal{R}\}$.

The set $\mathcal{R}$ is called the *rational control*, and C the *extended alphabet*.

Note that for an arbitrary set $\mathcal{R}$ of endomorphisms of C^* we have $\{h(c) \mid h \in \mathcal{R}\} \subseteq C^*$, but the definition implies that $\mathcal{R}$ must guarantee $h(c) \in A^*$ for all $h \in \mathcal{R}$.

Example 1.2. Let $A = \{a, b\}$ and $C = \{a, b, \#\}$. Consider four endomorphisms f, g_a, g_b, h defined as $f(\#) = \#\#$, $g_a(\#) = a\#$, $g_b(\#) = b\#$, and $h(\#) = 1$, and on all other letters f, g_a, g_b, h behave like the identity. Consider the rational language $\mathcal{R} = h\,\{g_a, g_b\}^* f$ (where endomorphisms are applied right-to-left). A simple inspection shows that $\{\varphi(\#) \mid \varphi \in R\} = \{vv \mid v \in A^*\}$, which is not context-free.

1.4. *Complexity*

We use the standard $\mathcal{O}$-notation for functions from $\mathbb{N}$ to $\mathbb{R}_{\geq 0}$. A function f is called *quasi-linear* if $f(n) \in \mathcal{O}(n \log n)$. We say that f is *singly exponential* if $f(n) \in 2^{\mathcal{O}(p(n))}$ where $p(n)$ is a polynomial. We also use the standard meaning of complexity classes like NP, $\mathsf{NSPACE}(f)$, $\mathsf{DSPACE}(f)$ and $\mathsf{DTIME}(f)$ as in [17].

Let $\mathcal{C}$ and $\mathcal{D}$ be two domains and for each $x \in \mathcal{C} \cup \mathcal{D}$ we let $\langle x \rangle \in \{0,1\}^*$ denote some binary encoding. We assume that for every $x \in \mathcal{C}$ its *input size* is defined as a natural number which might be different from the binary length of $\langle x \rangle$. For example, in our case we define the input size of an equation over a free group or monoid to be the length of the equation plus the number of generators of the group or monoid. As usual, we omit details on the specific encoding and how to check that a binary string y is of the form $y = \langle x \rangle$ for some $x \in \mathcal{C}$. In our case, we content ourselves that the encoding of a word of length n over some alphabet Γ uses at most $\mathcal{O}(n \log |\Gamma|)$ bits and that the check $y = \langle x \rangle$ can be done deterministically in linear space with respect to the binary length of y.

A function $t : \mathcal{C} \to \mathcal{D}$ is computable in $\mathsf{NSPACE}(f)$ if there is a nondeterministic Turing machine M with a two-way read-only input tape, a work tape, and a write-only output tape. The input $x \in \mathcal{C}$ is given as the binary string $\langle x \rangle$. During the computation the machine writes some binary string on the output tape from left to right such that for the entire computation the size of M's work tape is bounded by $\mathcal{O}(f(n))$ where n is the input size of x. There must be at least one run of the machine where M stops and if M stops, then output must be the correct value $\langle f(x) \rangle$. We rely on a result by Immerman and Szelepcsényi which implies that $\mathsf{NSPACE}(f)$ is (effectively) closed under complementation for functions f satisfying $\log n \in \mathcal{O}(f(n))$ [17, Theorem 7.6]). As a consequence, "trimming" an automaton will become possible in $\mathsf{NSPACE}(n \log n)$ in Sec. 3.8. Recall that every $\mathsf{NSPACE}(n \log n)$-computable function can also be simulated by some deterministic algorithm in time $2^{\mathcal{O}(n \log n)}$ (see [17, Theorem 3.3]).

1.5. *Word equations over monoids with rational constraints*

Let A be an alphabet of *constants* with involution and let $\pi : A^* \to M$ be a surjective morphism onto a monoid with involution M. Furthermore, let $\mathcal{X}$ be a set of *variables*. We may assume that $\mathcal{X}$ is endowed with an involution without fixed points. Thus, $X \neq \overline{X}$ for all $X \in \mathcal{X}$.

Definition 1.3. A *word equation with rational constraint over* M is a pair (U, V) of words $U, V \in (A \cup \mathcal{X})^*$ which has the following attributes.

- The *input size* of the equation is defined as $|A| + |UV|$.
- The rational constraint is given by a homomorphism $\nu : (A \cup \mathcal{X})^* \to N$, where N is a finite monoid.
- A *solution* of the equation (U, V) with constraint ν is given by a map

$$\sigma : \mathcal{X} \to A^*$$

which extends to a homomorphism $\sigma : (A \cup \mathcal{X})^* \to A^*$ that fixes the constants, such that for all $X \in \mathcal{X}$:

(1) $\sigma(\overline{X}) = \overline{\sigma(X)}$, i.e. $\sigma : \mathcal{X} \to A^*$ is a morphism,
(2) $\nu(X) = \nu\sigma(X)$, i.e. the solution respects the constraint on X,
(3) $\pi\sigma(U) = \pi\sigma(V)$, i.e. $\sigma(U)$ and $\sigma(V)$ are equal in the monoid M.

Note that we constrain the solutions to be in a recognizable set (see the definitions in Sec. 1.3), but in this case the notions of recognizable and rational sets are the same, since we are in the free monoid $(A \cup X)^*$.

2. Solution Sets for Equations Over Free Monoids With Involution and Free Groups: The Main Results

Let $A_\pm = A_+ \cup \{\overline{a} \,|\, a \in A_+\}$ be a finite alphabet with involution and assume that the involution is without fixed points: $\overline{a} \neq a$ for all $a \in A_\pm$. We let $\mathrm{F}(A_+)$ be the free group over A_+ and we realize the involution inside $\mathrm{F}(A_+)$ by $\overline{a} = a^{-1}$. Thus

$$A_\pm = A_+ \cup \left\{a^{-1} \,|\, a \in A_+\right\} \subseteq \mathrm{F}(A_+) \subseteq A_\pm^*.$$

Following standard terminology, a word $w \in A_\pm^*$ is *reduced* if it does not contain any factor $a\overline{a}$ where $a \in A_\pm$. The set of reduced words is a regular subset $\mathbb{F} \subseteq A_\pm^*$ which is closed under involution. We fix $\mathbb{F}$ as a set of normal forms for $\mathrm{F}(A_+)$; thus, as a set, we identify $\mathrm{F}(A_+)$ with $\mathbb{F}$. The inclusion $A_\pm \subseteq \mathrm{F}(A_+)$ induces the canonical projection $\pi : A_\pm^* \to \mathrm{F}(A_+)$. Given a word w we obtain $\pi(w)$ by a repeated cancellation of all factors $a\overline{a}$; and w is reduced if and only if $\pi(w) = w$.

We shall also use a special symbol # which is not in $A_\pm$ and serves as "marker". For example, we will encode a system of equations $\{(U_i, V_i) \,|\, 1 \leq i \leq s\}$ as a single equation

$$(U_1\# \cdots \#U_s, V_1\# \cdots \#V_s). \tag{2.1}$$

If we require that no $\sigma(X)$ is allowed to use #, where X is a variable, then

$$\forall\, i : \pi\sigma(U_i) = \pi\sigma(V_i) \Leftrightarrow \pi\sigma(U_1\# \cdots \#U_s) = \pi\sigma(V_1\# \cdots \#V_s) \tag{2.2}$$

since positions of the # letters must be the same on both sides. In our context, rational constraints are the most convenient way to ensure that no # appears in $\sigma(X)$, see Sec. 3.2. We let

$$A = A_\pm \cup \{\#\}$$

with $\overline{\#} = \#$. Thus, $\{1, \#\}$ forms a group which is isomorphic to $\mathbb{Z}/2\mathbb{Z}$ if we let $\#^{-1} = \#$.

In order to have a uniform statement we let $\mathbb{M}(A)$ be either the free monoid with involution A^* or the free product of the free group $\mathrm{F}(A_+)$ with the cyclic group $\{1, \#\}$ of order 2. Thus, henceforth:

$$\mathbb{M}(A) = A^* \quad \text{or} \quad \mathbb{M}(A) = A^* / \{a\overline{a} = 1 \,|\, a \in A\},$$

and $\pi : A^* \to \mathbb{M}(A)$ is the canonical projection induced by the inclusion $A \subseteq \mathbb{M}(A)$. In both cases π is injective on $\mathbb{F} \subseteq A^*$, and if $\mathbb{M}(A) = A^*$, then π is just the identity.

Given a word equation (U, V) with $UV \in (A_\pm \cup \mathcal{X})^*$ over $\mathbb{M}(A)$, we say that a solution σ is a *solution in reduced words* if $\sigma(X) \in \mathbb{F}$ for all $X \in \mathcal{X}$. We will realize this condition as a rational constraint μ into a finite monoid N with a zero element $0 \in N$ such that $\mu(w) \neq 0$ if and only if $w \in \mathbb{F}$.

Theorem 2.1. *Let (U, V) be an equation over $\mathbb{M}(A)$ of input size $n = |A| + |UV|$ (according to Definition* 1.3*) and in variables $X_1, \overline{X_1}, \ldots, X_m, \overline{X_m}$. Then there is an* $\mathsf{NSPACE}(n \log n)$ *algorithm which computes $c_1, \ldots, c_m \in C$, where $C \supseteq A$ is an extended alphabet of size $|C| \in \mathcal{O}(n)$, and a trim NFA $\mathcal{A}$ which produces the set of solutions in reduced words. That is,*

$$\begin{aligned}&\{(\sigma(X_1), \ldots, \sigma(X_m)) \in \mathbb{F} \times \cdots \times \mathbb{F} \mid \pi\sigma(U) = \pi\sigma(V)\} \\ &\quad = \{(h(c_1), \ldots, h(c_m)) \in C^* \times \cdots \times C^* \mid h \in L(\mathcal{A})\}.\end{aligned} \tag{2.3}$$

The NFA has the following properties.

(1) *It is nonempty if and only if the equation (U, V) has some solution.*
(2) *It has a directed cycle if and only if (U, V) has infinitely many solutions.*

These properties can also be decided in $\mathsf{NSPACE}(n \log n)$.

Recall that the input size n used in the statement of the theorem might be smaller than the length of some binary encoding for the input. If the number of distinct symbols used in the equation is constant, then our algorithm is quasilinear in the input size; if, on the other hand, the number of distinct symbols used in the equation is linear, then we need linear space, only.

Theorem 2.1 yields the characterization of solutions sets as EDT0L languages. To do so, we identify a tuple of words $(w_1, \ldots, w_k) \in \mathbb{F}$ with the single word $w_1\# \cdots \#w_k \in A^*$.

Let (U, V) be an equation as in Theorem 2.1. For any subset $\{Z_1, \ldots, Z_k\}$ of variables appearing in UV we define the solution set as

$$\mathrm{Sol}_{\mathcal{Z}}(U, V) = \{\sigma(Z_1)\# \cdots \#\sigma(Z_k) \,|\, \sigma \text{ solves } (U, V) \text{ in reduced words}\}\,. \tag{2.4}$$

Note that for $k = 0$ we have $\mathrm{Sol}_{\emptyset}(U, V) = \emptyset$ if the equation (U, V) has no solution and $\mathrm{Sol}_{\emptyset}(U, V) = \{1\}$ otherwise. Considering subsets of variables allows for some flexibility. In particular, we can introduce auxiliary variables which do not impact the solution set. If, however, every variable occurring in UV is either of the form Z_i or $\overline{Z_i}$ for some $1 \le i \le k$, then we say that $\mathrm{Sol}_{\mathcal{Z}}(U, V)$ is a *full solution set.*

Corollary 2.2. *Let (U, V) be an equation as in Theorem* 2.1 *and let $\{Z_1, \ldots, Z_k\}$ be any subset of variables appearing in UV. Then $\mathrm{Sol}_{\mathcal{Z}}(U, V)$ is an EDT0L language. More precisely, if $\mathcal{A}$ is the trim NFA constructed in Theorem* 2.1, *then we can find $c'_1, \ldots, c'_k \in C$ such that*

$$\mathrm{Sol}_{\mathcal{Z}}(U, V) = \{h(c'_1\# \cdots \#c'_k) \,|\, h \in L(\mathcal{A})\}\,.$$

In particular, the full solution set is EDT0L.

Proof. The language characterization follows from Definition 1.1 of an EDT0L language, given that each Z_j corresponds to some X_i in Theorem 2.1. □

Note that Theorem 2.1 shifts the traditional perspective from solving an equation to an effective construction of some NFA producing an EDT0L set. Once the NFA is constructed, the existence of a solution, or whether the number of solutions in reduced words is zero, finite or infinite, become graph properties of the NFA. Thus, the algorithmic difficulty of solving equations and describing their solution set reduces to the complexity of building a nondeterministic finite automaton for a given input.

3. Proof of Theorem 2.1 in the Monoid Case: $\mathbb{M}(A) = A^*$

In this section, we prove Theorem 2.1 in the monoid case. Before delving into the proof, we introduce in Secs. 3.1–3.7 further necessary terminology and notation.

Let $\mathbb{M}(A) = A^*$. In this case $\pi = \mathrm{id}_{A^*}$ and so π is not needed in the rest of this section. Without restriction, we may assume $|A_+| \geq 1$. Let $\mathcal{X}_{\text{init}} = \left\{X_1, \overline{X_1}, \ldots, X_m, \overline{X_m}\right\}$ be the initial set of variables, that is, for each $1 \leq i \leq m$ either X_i or $\overline{X_i}$ occur in UV.

Let $\kappa \in \mathcal{O}(1)$ be some "large enough" constant, whose exact value will be discussed in Sec. 3.10.4, and choose an alphabet C of *constants* and an alphabet Ω of *variables* such that

$$C \supseteq A, |C| = \kappa \cdot n \quad \text{and} \quad \Omega \supseteq \mathcal{X}_{\text{init}}, |\Omega| = 6n.$$

Fix $\Gamma = C \cup \Omega$. We assume that C and Ω are sets with involution and that, inside $\Gamma = C \cup \Omega$, the marker $\#$ is the only self-involuting symbol. Thus, $\overline{\#} = \#$ and $\overline{x} \neq x$ for all $x \in \Gamma \backslash \{\#\}$.

By Σ we denote the set of C-morphisms $\sigma : \Gamma^* \to C^*$. Every solution will be drawn from Σ.

3.1. *The initial word equation* W_{init}

For technical reasons we need that for every variable X_i which appears in UV there is some factor $\#X_i\#$ appearing in the initial equation. Instead of viewing equations as equalities between two words U and V, we will treat equations as a statement about a single word $W \in \Gamma^*$, as follows. This will require us to redefine the notion of solution as well.

We define the *initial equation* $W_{\text{init}} \in (A \cup \mathcal{X}_{\text{init}})^*$ as:

$$W_{\text{init}} = \#X_1\# \cdots \#X_m\#U\#V\#\overline{U}\#\overline{V}\#\overline{X_m}\# \cdots \#\overline{X_1}\#. \tag{3.1}$$

Then for every $\sigma \in \Sigma$ we have

$$\sigma(U) = \sigma(V) \Leftrightarrow \sigma(W_{\text{init}}) = \sigma(\overline{W_{\text{init}}})$$

and

$$\begin{aligned}&\{(\sigma(X_1), \ldots, \sigma(X_m)) \in \mathbb{F} \times \cdots \times \mathbb{F} \mid \sigma \in \Sigma \wedge \sigma(U) = \sigma(V)\}\\ &\quad = \{(\sigma(X_1), \ldots, \sigma(X_m)) \in \mathbb{F} \times \cdots \times \mathbb{F} \mid \sigma \in \Sigma \wedge \sigma(W_{\text{init}}) = \sigma(\overline{W_{\text{init}}})\}.\end{aligned}$$

We have the following symmetry: if $w \leq W_{\text{init}}$ is a factor and no # appears in w, then $\overline{w} \leq W_{\text{init}}$, too. The number of # letters in W_{init} is odd, and there is a distinguished # exactly in the middle of W_{init}.

Observe that W_{init} is longer than UV, but clearly linear in n. More concretely, since $m \leq |UV|$ and $n = |UV| + |A| > |UV| + 1$, we get the bound:

$$|W_{\text{init}}| \leq 4m + 5 + 2 \cdot |UV| \leq 6 \cdot |UV| + 5 < 6(|UV| + 1) < 6n. \tag{3.2}$$

Also observe that $\sum_{X \in \mathcal{X}_{\text{init}}} |W_{\text{init}}|_X \leq 2m + 2\,|UV| \leq 4n$.

3.2. *The finite monoid $N_{\mathbb{F}}$*

In order to ensure that solutions are in reduced words which do not contain the symbol #, we introduce a morphism to a fixed finite monoid $N_{\mathbb{F}}$ which plays the role of (a specific) rational constraint. We define $N_{\mathbb{F}}$ as follows: $N_{\mathbb{F}} = \{1, 0\} \cup (A_\pm \times A_\pm)$ with multiplication given by $1 \cdot x = x \cdot 1 = x$, $0 \cdot x = x \cdot 0 = 0$, and

$$(a, b) \cdot (c, d) = \begin{cases} (a, d) & \text{if } b \neq \overline{c}, \\ 0 & \text{otherwise.} \end{cases}$$

The monoid $N_{\mathbb{F}}$ has a natural involution given by $\overline{1} = 1$, $\overline{0} = 0$, and $\overline{(a, b)} = (\overline{b}, \overline{a})$.

The morphisms to $N_{\mathbb{F}}$ are defined on subsets of Γ, and although they change during the algorithm, they always extend the following fixed morphism

$$\mu_0 : A^* \to N_{\mathbb{F}}$$

which is defined by

$$\mu_0(\#) = 0, \quad \mu_0(a) = (a, a)$$

for $a \in A_\pm$. It is clear that μ_0 respects the involution and $\mu_0(w) = 0$ if and only if either w contains # or w is not reduced. If, on the other hand, $1 \neq w \in A_\pm^*$ is reduced, then $\mu_0(w) = (a, b)$, where a is the first and b the last letter of w. An additional feature is that $\mu(w) = 1$ if and only if w is the empty word.

Defining $\mu(X)$ for a variable X has the following meaning for a solution σ with $\sigma(X) \in A_\pm^*$: the value $\mu(X) = 0$ is not possible in any solution, $\mu(X) = 1$ implies $\sigma(X) = 1$, and $\mu(X) = (a, b) \Leftrightarrow \sigma(X) \in \mathbb{F} \cap a\mathbb{F} \cap \mathbb{F}b$.

3.3. *Types*

Later in the proof we will need to perform compression of large blocks of letters in an efficient manner. This will be achieved by putting a partially commutative structure on the monoid we work with. The partial commutativity will be induced by *types*, which we introduce below. The basic idea is that we assign a variable X the "type" $\theta(X) = c$ when we predict that in some solution $\sigma(X) \in c^*$ (so X and c commute), and we assign a constant b the "type" $\theta(b) = c$ when we rename some letters b as c.

Besides the initial alphabet A and the global alphabet C, we also need a *current* alphabet of constants B, where $A \subseteq B = \overline{B} \subseteq C$, and a *current* set of variables $\mathcal{X} = \overline{\mathcal{X}} \subseteq \Omega$. Let $\Delta = B \cup \mathcal{X}$. A *type* is a partially defined function $\theta : (\Delta \backslash A) \to (B \backslash A)$ which respects the involution. We identify θ with the relation $\{(\theta(x), x) \in \Delta \times \Delta \,|\, \theta(x) \text{ is defined}\}$. We obtain an independence relation

$$\theta = \{(\theta(x), x) \in \Delta \times \Delta \,|\, \theta(x) \text{ is defined for } x\}$$

and hence a free partially commutative monoid

$$M(\Delta, \theta) = \Delta^* / \{x\theta(x) = \theta(x)x \,|\, \theta(x) \text{ is defined for } x\}.$$

If the domain where θ is defined is empty, then $M(\Delta, \theta) = M(\Delta, \emptyset)$ is the free monoid Δ^*.

Remark 3.1. By definition, the size $|\theta|$ is bounded by $|\Delta|$. Hence, it is linear in n and the specification of θ needs $\mathcal{O}(n \log n)$ bits.

Definition 3.2. Let B satisfy $A \subseteq B = \overline{B} \subseteq C$, $\mathcal{X} = \overline{\mathcal{X}} \subseteq \Omega$, and θ be a type. The notation

$$M(B, \mathcal{X}, \theta, \mu)$$

denotes the free partially commutative monoid with involution $M(B \cup \mathcal{X}, \theta)$, equipped with a morphism $\mu : M(B \cup \mathcal{X}, \theta) \to N_{\mathbb{F}}$ such that $\mu(a) = \mu_0(a)$ for all $a \in A$, where $\mu_0 : A^* \to N_{\mathbb{F}}$ is the morphism specified in Sec. 3.2. We call $M(B, \mathcal{X}, \theta, \mu)$ a *structured monoid*.

A *morphism* φ from $M(B, \mathcal{X}, \theta, \mu)$ to $M(B', \mathcal{X}', \theta', \mu')$ is a morphism of monoids with involution $\varphi : M(B, \mathcal{X}, \theta, \mu) \to M(B', \mathcal{X}', \theta', \mu')$ such that $\mu'\varphi = \mu$.

Definition 3.2 implies that whenever $\theta(x)$ is defined, then $\mu(x\theta(x)) = \mu(\theta(x)x)$ (because μ is a homomorphism). Henceforth we use the following conventions. If $B' \subseteq B$ and $\mathcal{X}' \subseteq \mathcal{X}$ with $A \subseteq B' = \overline{B'}$ and $\mathcal{X}' = \overline{\mathcal{X}'}$, then $M(B', \mathcal{X}', \theta, \mu)$ denotes the structured monoid $M(B', \mathcal{X}', \theta', \mu')$ where θ' and μ' are induced by the restrictions of θ and μ to $B' \cup \mathcal{X}'$. Moreover, if $M(B, \mathcal{X}, \theta, \mu)$ is known from the context, then we abbreviate $M(B, \emptyset, \theta, \mu)$ as $M(B)$. Since no letter from A is involved in a type, $M(A)$ is the free monoid with involution A^* together with the morphism $\mu_0 : A^* \to N_{\mathbb{F}}$, and

$$M(A) = M(A, \emptyset, \emptyset, \mu_0) \subseteq M(B) \subseteq M(B, \mathcal{X}, \theta, \mu) \xrightarrow{\mu} N_{\mathbb{F}}.$$

3.4. *Reference list of symbols*

In Table 1, we summarize notations introduced so far for easy reference. These conventions hold unless stated otherwise. They also apply to "primed" symbols such as B', where B' denotes a set with $A \subseteq B' = \overline{B'} \subseteq C$.

Table 1. Reference list of symbols.

$A_+ \subseteq A_\pm$, the initial alphabets without self-involuting letters.
$A_\pm \cup \{\#\} = A \subseteq B = \overline{B} \subseteq C$.
$\Gamma = C \cup \Omega$ and $x = \overline{x} \in \Gamma$ implies $x = \#$.
$\mathcal{X} = \overline{\mathcal{X}} \subseteq \Omega$, the current set of variables.
$n = \vert A\vert + \vert UV\vert$, $\vert C\vert = \kappa n$ and $\vert\Omega\vert = 6n$.
$\Delta = B \cup \mathcal{X}$.
$\mu : \Delta \to N_{\mathbb{F}}$, a morphism with $\mu(a) = \mu_0(a)$ for $a \in A$.
$\theta : (\Delta \backslash A) \to (B \backslash A)$, the type defining an independence relation.
$M(\Delta, \theta)$, free partially commutative monoid defined by Δ and θ.
$M(B, \mathcal{X}, \theta, \mu) = M(\Delta, \theta)$ together with μ which extends $\mu_0 : A^* \to N_{\mathbb{F}}$.
$M(B)$, submonoid of $M(B, \mathcal{X}, \theta, \mu)$ together with the restriction of θ, μ.
$a, b, c, \ldots$ refer to letters in C.
$u, v, w, \ldots$ refer to words in C^*.
$X, Y, Z, \ldots$ refer to variables in Ω.
$x, y, z, \ldots$ refer to words in Γ^*.

3.5. *Extended equations and their solutions*

The states of the NFA we are going to construct correspond to equations derived from our initial equation. Each state contains such an equation, together with the specification of which set of constants, variables and types are used. Moreover, we keep track of the morphism μ which represents the constraint. Formally, we use the notion of *extended equation*. The notions we introduce now are quite technical, but the reader should keep in mind that the most important fact is that an extended equation contains an equation which is a modification of the initial equation, and this equation has bounded length. When types are present, this equation is an element in a free partially commutative monoid rather than simply a word in a free monoid.

Definition 3.3. An *extended equation* is a tuple $(W, B, \mathcal{X}, \theta, \mu)$, where W is a word in $(B \cup \mathcal{X})^*$ such that:

(1) $|W| \leq 204n$.
(2) If $\theta = \emptyset$, then $\sum_{X \in \mathcal{X}} |W|_X \leq 4n$. Otherwise $\sum_{X \in \mathcal{X}} |W|_X \leq 12n$.
(3) $|W|_\# = |W_{\text{init}}|_\#$ and $W \in \#(B \cup \mathcal{X})^*\#$.
(4) Every x with $\# \neq x \in B \cup \mathcal{X}$ satisfies $\mu(x) \neq 0$.
(5) Every $X \in \mathcal{X}$ appears in W.
(6) If $x \leq W$ is a factor with $|x|_\# = 0$, then $\overline{x} \leq W$, too.

Remark 3.4. As noted above, the word W (including the notion of factor) is to be seen as representing an element in the free partially commutative monoid $M(B, \mathcal{X}, \theta, \mu) = M(B \cup \mathcal{X}, \theta)$. Note that by definition $|\theta| \leq |B \cup \mathcal{X}|$ (see Remark 3.1). The bounds on the length of W, and on the number of variables appearing in W, will be explained in later sections (Sec. 3.10.3), where we will show that we can find all solutions to an input equation by considering modified equations that satisfy these restrictions. What is important for now is that $|W| \in \mathcal{O}(n)$ which means the number of extended equations is finite.

Definition 3.5. Let $V = (W, B, \mathcal{X}, \theta, \mu)$ be an extended equation. The *weight* $\|V\|$ of V is a 4-tuple of natural numbers, $\|V\| = (\omega_1, \omega_2, \omega_3, \omega_4)$, where

$$\begin{aligned}\omega_1 &= |W|\,,\\ \omega_2 &= |W| - |\{a \in B \,|\, |W|_a \geq 1\}|\,,\\ \omega_3 &= |W| - |\theta|\,,\\ \omega_4 &= |B|\,.\end{aligned}$$

Remark 3.6. We order tuples in $\mathbb{N}^\ell$ lexicographically. The lexicographic ordering is chosen to function as follows. If we start at an equation of high weight, then the weight of the equation reduces by “compression”. The first component gives more weight to longer equations. If two equations have the same length, then we declare the equation in which more distinct constants appear to be smaller because the term $|\{a \in B \,|\, |W|_a \geq 1\}|$ appears with a negative sign. If two equations have the same length and use the same number of distinct constants, we declare the equation in which more symbols are typed to be smaller. Finally, if both equations have the same length, the same number of distinct letters in use, and the same number of typed symbols, then we declare the equation defined over the smaller set B to be smaller.

Since for every extended equation we have a current alphabet B, we need the notion of a B-solution, which can then be extended to a solution over the desired alphabet A. The next few pages are somewhat technical, but will be used to justify that when we modify extended equations in certain ways, solutions are preserved.

Definition 3.7. Let $V = (W, B, \mathcal{X}, \theta, \mu)$ be an extended equation.

- A *B-solution* at V is a B-morphism $\sigma : M(B, \mathcal{X}, \theta, \mu) \to M(B, \emptyset, \theta, \mu)$ such that $\sigma(W) = \sigma(\overline{W})$ and $\sigma(X) \in y^*$ whenever $(X, y) \in \theta$.
- A *solution* at V is a pair (α, σ) where σ is a B-solution and $\alpha : M(B, \emptyset, \theta, \mu) \to A^*$ is an A-morphism (which implies $\mu = \mu_0\alpha$). Moreover, if the set $\mathcal{X}$ in V is nonempty, then we require that α is nonerasing, that is, $\alpha(a) \neq 1$ for all $a \in B$.

The *weights* $\|\alpha, \sigma\|$ and $\|\alpha, \sigma, V\|$ of a solution (α, σ) at V are defined as

$$\|\alpha, \sigma\| = \sum_{X \in \mathcal{X}} |\alpha\sigma(X)| \in \mathbb{N}, \tag{3.3}$$

$$\|\alpha, \sigma, V\| = (\|\alpha, \sigma\|\,, \|V\|) \in \mathbb{N}^5. \tag{3.4}$$

Remark 3.8. Let $V = (W, B, \mathcal{X}, \theta, \mu)$ be an extended equation with a solution (α, σ). Then $\sigma(X)$ cannot have any factor of the form # or $a\overline{a}$ with $a \in B$ because

$0 \neq \mu(X) = \mu_0 \alpha\sigma(X)$. In particular, $\alpha\sigma(X)$ is a reduced word in $A_{\pm}^*$. Hence, $\alpha\sigma$ satisfies the constraint $\alpha\sigma(X) \in \mathbb{F}$. Note that *a priori* we do not exclude the possibility that factors $a\overline{a}$ appear in W, since for example it could be that W_{init} contains a factor aX and some solution $\sigma(X)$ begins with $\overline{a}$.

The next two lemmas show how morphisms between structured monoids transform solutions of extended equations. These two lemmas will play an important role in the proof of the algorithm "soundness".

In the first lemma we consider the morphisms which leave all constants invariant, and conclude that such a morphism decreases the weight of a solution. In addition, this lemma specifies a situation, in part (iv), when the weight strictly decreases.

Lemma 3.9. *Let* $V = (W, B, \mathcal{X}, \theta, \mu)$ *and* $V' = (W', B, \mathcal{X}', \theta', \mu')$ *be extended equations such that* $\theta(a) = \theta'(a)$ *and* $\mu(a) = \mu'(a)$ *for all* $a \in B$. *In other words,* $M(B) = M(B, \emptyset, \theta, \mu) = M(B, \emptyset, \theta', \mu')$.

Let $\tau : M(B, \mathcal{X}, \theta, \mu) \to M(B, \mathcal{X}', \theta', \mu')$ *be a* B*-morphism such that* $W' = \tau(W)$ *and* $\alpha : M(B) \to M(A, \emptyset, \emptyset, \mu_0)$ *be an* A*-morphism such that* $\alpha(a) \neq 1$ *for all* $a \in B$.

Given a B*-solution* σ' *at* V', *define a* B*-morphism* $\sigma : M(B, \mathcal{X}, \theta, \mu) \to M(B)$ *by* $\sigma(X) = \sigma'\tau(X)$.

Then the following assertions hold.

(i) (α, σ) *is a solution at* V *and* (α, σ') *is a solution at* V'.

(ii) $\alpha\sigma(W) = \alpha\sigma'(W')$.

(iii) $\|\alpha, \sigma\| \geq \|\alpha, \sigma'\|$.

(iv) *If there is some* X *with* $\tau(X) \in \mathcal{X}'^* a \mathcal{X}'^*$ *where* $a \in B$ *and* $\alpha(a) \neq 1$, *then* $\|\alpha, \sigma\| > \|\alpha, \sigma'\|$.

Proof. (i) Since σ' is a B-solution at V' we have

$$\sigma(W) = \sigma'\tau(W) = \sigma'(\overline{\tau(W)}) = \overline{\sigma'\tau(W)} = \overline{\sigma(W)} = \sigma(\overline{W}).$$

By hypothesis, $\alpha(a) \neq 1$ for all $a \in B$. Hence, (α, σ) is a solution at V. Since $M(B) = M(B, \emptyset, \theta, \mu) = M(B, \emptyset, \theta', \mu')$, we have (α, σ') is a solution at V'.

(ii) The assertion $\alpha\sigma(W) = \alpha\sigma'(W')$ is trivial since $W' = \tau(W)$, $\sigma = \sigma'\tau$.

(iii) For each X write $\tau(X)$ as a word

$$\tau(X) = x_{X,1} \cdots x_{X,\ell_X}$$

with $x_{X,i} \in B \cup \mathcal{X}'$. Since every $X' \in \mathcal{X}'$ appears somewhere in $\tau(W)$ (by Definition 3.3(5)) we obtain: $\mathcal{X}' \subseteq \bigcup \{x_{X,i} \mid X \in \mathcal{X} \wedge 1 \leq i \leq \ell_X\}$. Hence

$$\|\alpha, \sigma\| = \sum_{X \in \mathcal{X}} |\alpha\sigma(X)| = \sum_{X \in \mathcal{X}} |\alpha\sigma'\tau(X)| \tag{3.5}$$

$$= \sum_{X \in \mathcal{X}} |\alpha\sigma'(x_{X,1} \cdots x_{X,\ell_X})| \sum_{X \in \mathcal{X}, 1 \leq i \leq \ell_X} |\alpha\sigma'(x_{X,i})| \tag{3.6}$$

$$\geq \sum_{X' \in \mathcal{X}'} |\alpha\sigma'(X')| = \|\alpha, \sigma'\|\,. \tag{3.7}$$

(iv) If there is some X with $\tau(X) \in \mathcal{X}'^* a \mathcal{X}'^*$ where $a \in B$ and $\alpha(a) \neq 1$, then some $x_{X,i} = a \notin \mathcal{X}'$ with $\alpha\sigma'(a) = \alpha(a) \neq 1$. Hence, $|\alpha\sigma'(x_{X,i})| \geq 1$; and the $\geq$ in (3.7) becomes the inequality $>$. □

In the second lemma we consider the morphisms which leave all variables invariant, and conclude that such a morphism does not change the weight of a solution.

Lemma 3.10. *Let $V = (W, B, \mathcal{X}, \theta, \mu)$ and $V' = (W', B', \mathcal{X}, \theta', \mu')$ be extended equations, $h : M(B', \mathcal{X}, \theta', \mu') \to M(B, \mathcal{X}, \theta, \mu)$ be an $(A \cup \mathcal{X})$-morphism , and $\alpha : M(B) \to M(A, \emptyset, \emptyset, \mu_0)$ be an A-morphism where $M(B) = M(B, \emptyset, \theta, \mu)$ such that the following conditions are satisfied.*

- *$W = h(W')$.*
- *$\alpha(a) \neq 1$ for all $a \in B$.*
- *If $\mathcal{X} \neq \emptyset$, then $h(a') \neq 1$ for all $a' \in B'$.*
- *If $\theta(X) = c \in B$ for some $X \in \mathcal{X}$, then $c \in B', \theta'(X) = c$, and $h(c) \in c^*$.*

Given a B'-solution σ' at V', define a B-morphism $\sigma : M(B, \mathcal{X}, \theta, \mu) \to M(B)$ by $\sigma(X) = h\sigma'(X)$. Then (α, σ) is a solution at V and $(\alpha h, \sigma')$ is a solution at V'. Moreover, $\alpha\sigma(W) = \alpha h\sigma'(W')$ and

$$\|\alpha, \sigma\| = \|\alpha h, \sigma'\| .$$

Proof. By definition, $\mu h = \mu'$ and $\mu_0\alpha = \mu$. Hence $(\alpha h, \sigma')$ is a solution at V'. Now, $h(X) = X$ for all $X \in \mathcal{X}$. Hence, $\sigma(h(X)) = \sigma(X) = h\sigma'(X)$. For $b' \in B'$ we obtain $\sigma h(b') = h(b') = h\sigma'(b')$ since σ' and σ are the identity on B' and B, respectively. It follows that $\sigma h = h\sigma'$ and hence, $\alpha\sigma(W) = \alpha h\sigma'(W')$. Next,

$$\sigma(W) = \sigma(h(W')) = h(\sigma'(W')) = h(\sigma'(\overline{W'})) = \sigma(h(\overline{W'})) = \sigma(\overline{h(W')}) = \sigma(\overline{W}).$$

Moreover, if $X \in \mathcal{X}$ and $\theta(X)$ is defined, then $\theta(X) = \theta'(X) = c \in B \cap B'$, and $h(c) \in c^*$ by hypothesis. Hence, $\sigma'(X) \in c^*$ and therefore $\sigma(X) = h\sigma'(X) \in c^*$, too. Thus, σ is a B-solution at V and, consequently, (α, σ) a solution at V. Finally, since $\sigma(X) = h\sigma'(X)$ we obtain

$$\|\alpha, \sigma\| = \sum_{X \in \mathcal{X}} |\alpha\sigma(X)| = \sum_{X \in \mathcal{X}} |\alpha h\sigma'(X)| = \|\alpha h, \sigma'\| .$$ □

During the process of finding a solution, the parameters $W, B, \mathcal{X}, \theta, \mu$ change. We describe the possible changes in terms of a directed graph, which will be converted into an NFA.

3.6. *The NFA $\mathcal{F}$ and the trimmed NFA $\mathcal{A}$*

We are ready to define the NFA $\mathcal{A}$ mentioned in Theorem 2.1 in the case where $\mathbb{M}(A) = A^*$ is a free monoid with involution.

3.6.1. *States*

We start by building an NFA $\mathcal{F}$ whose states are all the extended equations $(W, B, \mathcal{X}, \theta, \mu)$ according to Definition 3.3. We will later obtain $\mathcal{A}$ by trimming, that is, by removing all states which are not on accepting paths. Thus, the only difference between $\mathcal{F}$ and $\mathcal{A}$ is that $\mathcal{A}$ does not have superfluous states.

Lemma 3.11. *An extended equation $V = (W, B, \mathcal{X}, \theta, \mu)$ can be specified using at most $\mathcal{O}(n \log n)$ bits, so $\mathcal{F}$ has not more than singly exponentially many states.*

Proof. We claim that each component of V can be specified using $\mathcal{O}(|\Gamma|) = \mathcal{O}(n)$ letters from Γ plus a finite alphabet. Since $|\Gamma| \in \mathcal{O}(n)$, we can encode each letter in Γ plus the finite alphabet as a binary number of length at most $\mathcal{O}(\log n)$ bits. Thus V can be encoded by a binary string of length in $\mathcal{O}(n \log n)$. It follows that the total number of extended equations is at most $2^{\mathcal{O}(n \log n)}$.

To establish the claim, notice that $W \in \Gamma^*$ with $|W| \leq 204n$, $B \cup \mathcal{X} \subseteq \Gamma$, $\theta \subset \Gamma \times \Gamma$ and $|\theta| \leq |B \cup \mathcal{X}|$. Since $\mu : B \cup \mathcal{X} \to N_{\mathbb{F}}$ and $N_{\mathbb{F}}$ is finite, μ can be encoded as a list $\{(c, \mu(c)) \mid c \in B \cup \mathcal{X}\}$, using letters from Γ plus the finite alphabet $N_{\mathbb{F}}$. □

Initial states. An *initial state* is any state of the form $(W_{\text{init}}, A, \mathcal{X}_{\text{init}}, \emptyset, \mu_{\text{init}})$, where

$$\mu_{\text{init}} : (A \cup \mathcal{X}_{\text{init}}) \to N_{\mathbb{F}}$$

is a morphism extending μ_0 such that $\mu_{\text{init}}(X) \neq 0$ for all $X \in \mathcal{X}_{\text{init}}$.

If (α, σ) is a solution of $(W_{\text{init}}, A, \mathcal{X}_{\text{init}}, \emptyset, \mu_{\text{init}})$, then necessarily $\alpha = \text{id}_{A^*}$ since α leaves the letters from A invariant. Moreover, we know that $\mu_{\text{init}}(X) = \mu_0\sigma(X)$. This means that the initial value of $\mu_{\text{init}}(X)$ tells us whether $\sigma(X) = 1$; and if $\sigma(X) \neq 1$, then $\mu_{\text{init}}(X) = (a, b)$ and $\sigma(X) \in aA^* \cap A^*b$. Hence, $\mu_{\text{init}}(X)$ specifies the first and last letters of the reduced word $\sigma(X)$ whenever $\sigma(X) \neq 1$. Moreover, $\mu_{\text{init}}(X) \neq 0$ implies $\alpha\sigma(X) \in \mathbb{F}$. Hence, $\alpha\sigma(X)$ is a reduced word in $A_{\pm}^*$.

Final states. We choose and fix "distinguished" letters $c_1, \ldots, c_m \in C \backslash A$ such that $c_i \neq c_j \neq \overline{c_i}$ for all $i \neq j$. We say that a state $(W, B, \emptyset, \emptyset, \mu)$ is *final* if

(1) $W = \overline{W}$,
(2) The word W has a prefix of the form $\#c_1\# \cdots \#c_m\#$.

Every final state has the unique B-solution $\sigma = \text{id}_B$ because final states do not have any variables.

Remark 3.12. The names *initial* and *final* refer to the phase in the construction of the graph at which a state is produced, rather than being start or accept states for the NFA. That is, when we obtain the EDT0L language characterization, the start states of the NFA recognizing the rational language of endomorphisms correspond to the final states defined here, and the accept states correspond to the initial states.

3.7. *Transitions*

We define two different forms of transitions, based on substitutions and compressions. Both forms are labeled by an endomorphism of C^* which induces a morphism between partially commutative monoids $M(B, \emptyset, \theta, \mu)$ and $M(B', \emptyset, \theta', \mu')$.

The direction of each transition is opposite to that of the morphism labeling the transition. Suppose we have a path p from an initial to a final state. A very important (and, perhaps, initially counterintuitive) fact is that in order to produce solutions, our algorithm follows the path p backwards, that is, from the final to the initial state; we compose the morphisms labeling the transformations in such a directed path p from the last edge to the first one, in order to produce the solutions. This is in agreement with our initial and final states being accept and start states in the NFA, respectively.

3.7.1. *Substitutions*

A *substitution transition* transforms the variables and does not affect the constants. Let $V = (W, B, \mathcal{X}, \theta, \mu)$ and $V' = (W', B, \mathcal{X}', \theta', \mu')$ be states in $\mathcal{F}$ sharing the same set of constants B; and assume that V is not final and that V' is not an initial state. Moreover, let $\theta(b) = \theta'(b)$, and $\mu(b) = \mu'(b)$ for all $b \in B$. Therefore $M(B) = M(B, \emptyset, \theta, \mu) = M(B, \emptyset, \theta', \mu')$.

Let $\tau : M(B, \mathcal{X}, \theta, \mu) \to M(B, \mathcal{X}', \theta', \mu')$ be any B-morphism such that $\tau(W) = W'$, τ modifies only X and $\overline{X}$ for some variable X, leaves all $x \in (B \cup \mathcal{X}) \setminus \{X, \overline{X}\}$ invariant, and

$$\tau(X) \in (B \cup \mathcal{X}')^* \quad \text{with } |\tau(X)| \leq 3.$$

Furthermore, we only allow the following choices for $\tau(X), \mathcal{X}$ and $\mathcal{X}'$:

(i) $\tau(X) = 1$ and $\mathcal{X}' = \mathcal{X} \setminus \{X, \overline{X}\}$.
(ii) $\tau(X) = uX$ and $\mathcal{X}' = \mathcal{X}$ with $u \in B^*$ and $1 \leq |u| \leq 2$.
(iii) $\tau(X) = cX'X$ and $\mathcal{X} = \mathcal{X}' \setminus \{X', \overline{X'}\}$ with $c \in B$ and $\theta'(X') = c$.

In each of these three cases we define the substitution transition:

$$V = (W, B, \mathcal{X}, \theta, \mu) \xrightarrow{\varepsilon} (\tau(W), B, \mathcal{X}', \theta', \mu') = V'.$$

Here, the label ε denotes the identity morphism id_{C^*}, it restricts to the identity morphism from $M(B, \emptyset, \theta', \mu')$ to $M(B, \emptyset, \theta, \mu)$, and it will be applied in the opposite direction from τ and the transition. Note that after having performed a substitution transition we have $\|V'\| < \|V\|$ if and only if τ is defined by $\tau(X) = 1$ for some X.

3.7.2. *Compressions*

A *compression transition* affects the constants, but does not change the variables. Let $V = (W, B, \mathcal{X}, \theta, \mu)$ and $V' = (W', B', \mathcal{X}, \theta', \mu')$ be states in $\mathcal{F}$ sharing the same set of variables $\mathcal{X}$ and assume V is not a final state, $\theta(X) = \theta'(X)$ and $\mu(X) = \mu'(X)$ for all $X \in \mathcal{X}$.

Let $h : M(B', \mathcal{X}, \theta', \mu') \to M(B, \mathcal{X}, \theta, \mu)$ be any $(A \cup \mathcal{X})$-morphism such that $W = h(W')$ and

(1) if V' is non-final, then $1 \leq |h(c)| \leq 2$ for all $c \in B'$,
(2) if V' is final, then $\sum_{c \in B'} |h(c)| \leq |W|$.

In case that either $\|V\| > \|V'\|$ or V' is final and $h \neq \mathrm{id}_{B^*}$, we define a compression transition in $\mathcal{F}$ by

$$V = (h(W'), B, \mathcal{X}, \theta, \mu) \xrightarrow{h} (W', B', \mathcal{X}, \theta', \mu') = V',$$

where the transition label h is given by an endomorphism $h \in \mathrm{End}(C^*)$ which induces the morphism $h : M(B', \mathcal{X}, \theta', \mu') \to M(B, \mathcal{X}, \theta, \mu)$ and which leaves all letters not in B' invariant. The direction of the morphism h is again opposite to that of the transition.

Remark 3.13. The reason that we have to treat transitions to final states differently is two-fold. First, the coexistence of "singular" and "nonsingular" solutions is possible. In the singular case we have $\sigma(X) = 1$ for some X and in the nonsingular case we have $\sigma(X) \neq 1$ for all X. Say there are solutions σ and σ' such that $\sigma(X_1) = 1$ and $\sigma'(X_1) = a \in A_\pm$. Then for some $h, h' \in L(\mathcal{A})$ and some c_1 we must have $h(c_1) = 1$ and $h'(c_1) = a$. Thus in transformations to a final state we must allow that h maps some letters to the empty word. In all other situations this is forbidden. Thus, if $V \xrightarrow{h} V'$ is a compression transition and V' is final, then we allow $\|V\| < \|V'\|$.

Second, if a state $V = (W, B, \emptyset, \theta, \mu)$ has no variables, then W has prefix $\#u_1\# \cdots \#u_m\#$ with $u_i \in C^*$. In this case we wish to allow a compression transition h to a final state in one step. By imposing the condition $\sum_{c \in B'} |h(c)| \leq |W|$ we make sure the specification of h fits into our linear space bound, which is crucial in our complexity analysis below.

Example 3.14. Let $U = aX$ and $V = aaab$ be an equation, for the purposes of demonstrating how the graph or NFA works. We have

$$W_{\mathrm{init}} = \#X\#aX\#aab\#\overline{X}\overline{a}\#\overline{b}\overline{a}\overline{a}\#\overline{X}\#.$$

A path from initial to final states in the graph $\mathcal{F}$ for this equation is shown in Fig. 2, where for simplicity we label states by a prefix of W in each extended equation.

The first four transitions are substitutions $\tau_1(X) = \tau_2(X) = aX, \tau_3(X) = bX$, $\tau_4(X) = 1$ so h_1, h_2, h_3, h_4 are just id_{C^*}, and the map $h_5(c_1) = aab$ is a compression to a final state. A solution for X can be obtained by applying the maps to c_1 in reverse order to the path labeling, so we get $\sigma(X) = h_1h_2h_3h_4h_5(c_1) = h_1h_2h_3h_4(aab) = aab$.

$$\begin{tikzcd}
\boxed{\#X\#aX\#aaab\#\cdots} \arrow[d, "h_1"', "\tau_1(X)=aX"] \\
\boxed{\#aX\#aaX\#aaab\#\cdots} \arrow[d, "h_2"', "\tau_2(X)=aX"] \\
\boxed{\#aaX\#aaaX\#aaab\#\cdots} \arrow[d, "h_3"', "\tau_3(X)=bX"] \\
\boxed{\#aabX\#aaabX\#aaab\#\cdots} \arrow[d, "h_4"', "\tau_4(X)=1"] \\
\boxed{\#aab\#aaab\#aaab\#\cdots} \arrow[d, "h_5"', "h_5(c_1)=aab"] \\
\boxed{\#c_1\#ac_1\#ac_1\#\cdots}
\end{tikzcd}$$

Fig. 2. A path in $\mathcal{F}$ from initial to final state for the equation $aX = aaab$. The solution $\sigma(X)$ is obtained by applying the maps h_1, h_2, h_3, h_4, h_5 to c_1 in reverse order, that is, $\sigma(X) = h_1h_2h_3h_4h_5(c_1)$.

3.8. *Proof that the NFA is constructed in quasi-linear space*

We can now give the algorithm to construct the trim NFA $\mathcal{A}$ in $\mathsf{NSPACE}(n \log n)$. We first give an algorithm to construct $\mathcal{F}$, then use this to construct $\mathcal{A}$.

Lemma 3.15. *Given a tuple $V = (W, B, \mathcal{X}, \theta, \mu)$, where $W \in \Gamma^*, B \subseteq C, \mathcal{X} \subseteq \Omega, \theta$ is a type, and $\mu : (B \cup \mathcal{X}) \to N$ is a mapping, we can check within $\mathsf{NSPACE}(n \log n)$ whether V is an extended equation (that is, V is a state in $\mathcal{F}$) and furthermore decide whether the state V is initial or final.*

Proof. As noted in Lemma 3.11, writing down any extended equation requires at most $\mathcal{O}(n \log n)$ bits, so if V requires more space we reject it as a valid input. If V fits into the allowed space, then go through the conditions listed in Definition 3.3. It is obvious how to check the first five conditions. For example, if $|W| > 204n$, then we reject immediately.

The most involved test is to see that for every factor u of every u_i with the interpretation $u_i \in M(\Gamma, \theta)$ the element $\overline{u}$ also appears in $W \in M(\Gamma, \theta)$. For this test, we invoke the algorithm that solves the uniform factor problem in free partially commutative monoids as explained in Sec. 1.2. Recall that the uniform factor problem refers to an input of the form (Γ, θ, u, w). In our case the input has the specific form $(\Gamma, \theta, \overline{u}, W)$. We presented a nondeterministic algorithm using linear space in the input size, where the input size of a tuple (Γ, θ, u, w) is $(|\Gamma|+|\theta|+|uw|)\log|\Gamma|$, as we need $\mathcal{O}(\log|\Gamma|)$ bits to encode letters. Since $(|\Gamma|+|\theta|+|uw|)\log|\Gamma| \in \mathcal{O}(n\log n)$, the call of such a subroutine fits into our space bound.

Having completed the check that V is a state of $\mathcal{F}$, it is easy to check whether it is initial ($W = W_{\text{init}}$, $B = A$, $\theta = \emptyset$) or final ($W = \overline{W}$, $\theta = \emptyset$, $\mathcal{X} = \emptyset$); since $\theta = \emptyset$ in both cases we are just checking $W = W_{\text{init}}, W = \overline{W}$ in a free monoid. □

In the following, when we say that $V = (W, B, \mathcal{X}, \theta, \mu)$ is a state in $\mathcal{F}$, this means V is given as a tuple for which the syntax check according to Lemma 3.15 that V is indeed a state was performed.

Lemma 3.16. *Given states $V = (W, B, \mathcal{X}, \theta, \mu), V' = (W', B', \mathcal{X}', \theta', \mu')$ in $\mathcal{F}$, and a mapping $h : B' \to B^*$, we can check within* $\mathsf{NSPACE}(n\log n)$ *whether the triple (V, V', h) encodes an transition $V \xrightarrow{h} V'$ in the graph $\mathcal{F}$.*

Proof. We assume h is specified as a tuple requiring at most $\mathcal{O}(n\log n)$ bits. In order to check whether $V \xrightarrow{h} V'$ is a compression transition we must have $h \neq \mathrm{id}_{B^*}$ and then we go through the conditions of Sec. 3.8, most of which are immediate to verify. Among these, we have to compute $h(W')$ as a word in $(B \cup \mathcal{X})^*$ and then see if $W = h(W') \in M(B \cup \mathcal{X}, \theta)$. The test $W = h(W') \in M(B \cup \mathcal{X}, \theta)$ is a special case of the uniform factor problem in free partially commutative monoids, as already discussed in the proof of Lemma 3.15.

For a substitution transition, a necessary condition is $B = B'$ and $h = \mathrm{id}_B$, which is trivial to check. Next we guess some mapping $\tau : \mathcal{X} \to (B \cup \mathcal{X}')^*$ with $|\tau(X)| \leq 3$ for all $X \in \mathcal{X}$. Just as above we check $\tau(W) = W' \in M(B' \cup \mathcal{X}', \theta')$ and the other requirements for substitutions listed in Sec. 3.7.1. □

As usual in automata theory we modify the NFA $\mathcal{F}$ by removing all states which are not on a path from some initial to some final state. If there is no such path, then $L(\mathcal{F})$ is the empty set. The resulting NFA will be denoted as $\mathcal{A}$. We have $L(\mathcal{A}) = L(\mathcal{F})$. Moreover, $L(\mathcal{A}) = \emptyset$ if and only if the automaton $\mathcal{A}$ is empty.

The key tool used to build the trim NFA $\mathcal{A}$ is $\textsc{IsPath}(V, V')$, which we define to be a Boolean predicate that yields *true* if and only if there is a path from state V to V' in the graph $\mathcal{A}$.

Lemma 3.17. *Let V, V' represent two states in the graph $\mathcal{F}$. Then the predicate* $\textsc{IsPath}(V, V')$ *can be evaluated in* $\mathsf{NSPACE}(n\log n)$.

Proof. Define the language $L_{\mathcal{F}} = \{(V, V') \mid \textsc{IsPath}(V, V') = \text{true}\}$. On input (V, V') we can guess a path $V = V_0, V_1, h_1, V_2, h_2, \ldots, V' = V_k, h_k$ in $\mathcal{F}$ from V

to V' and check for each i whether (V_{i-1}, V_i, h_i) encodes a transition by using Lemmas 3.15 and 3.16. Thus, $L_{\mathcal{F}} \in \mathsf{NSPACE}(n \log n)$.

Since $\mathsf{NSPACE}(n \log n)$ is closed under complementation by Immerman and Szelepcsényi (see [17, Theorem 7.6]), we also have

$$\overline{L_{\mathcal{F}}} = \{(V, V') \mid \nexists \text{ a path from } V \text{ to } V' \text{ in } \mathcal{F}\} \in \mathsf{NSPACE}(n \log n).$$

Thus, the predicate $\textsc{IsPath}(V, V')$ can be evaluated in $\mathsf{NSPACE}(n \log n)$ by running two procedures simultaneously to determine if $(V, V') \in L_{\mathcal{F}}$ or $(V, V') \in \overline{L_{\mathcal{F}}}$. □

Proposition 3.18. *We can construct the trim NFA $\mathcal{A}$ in $\mathsf{NSPACE}(n \log n)$. Within the same space complexity we can decide whether $\mathcal{A}$ is empty, or whether $\mathcal{A}$ contains a directed cycle.*

Proof. For each V that is a state of $\mathcal{F}$ output V as an initial node of $\mathcal{A}$ if both (1) V is initial in $\mathcal{F}$, and (2) there exists some path to a final state in $\mathcal{F}$. We check (1) using Lemma 3.15. For (2) we run through all final states V' of $\mathcal{F}$ and evaluate the predicate $\textsc{IsPath}(V, V')$. If at some point $\textsc{IsPath}(V, V')$ becomes true, we output V as an initial node in $\mathcal{A}$. If no initial node in $\mathcal{A}$ is found, then we stop; the output is $\mathcal{A} = \emptyset$. Hence, we continue only if there is at least one initial node.

Next, we construct all transitions of $\mathcal{A}$ as follows. We list all triples (V, V', h) where $V \xrightarrow{h} V'$ is a transition in $\mathcal{F}$. For each such triple we consider all states V_0 of $\mathcal{A}$ which are initial, and for each V_0 we evaluate $\textsc{IsPath}(V_0, V)$. If no such V_0 is found where $\textsc{IsPath}(V_0, V)$ is true, then we move to the next triple (V, V', h). If at least one such V_0 exists, we list all states V_f of $\mathcal{F}$ which are final. For each V_f we evaluate $\textsc{IsPath}(V', V_f)$. If no such V_f is found where $\textsc{IsPath}(V', V_f)$ is true, then we move to the next triple (V, V', h). Otherwise we output (V, V', h) as a transition of $\mathcal{A}$. If, moreover, V' is final in $\mathcal{F}$, then we mark that transition in order to indicate that V' is final in $\mathcal{A}$, too. We then move to the next triple (V, V', h).

Having these two lists at hand we have constructed the trim NFA $\mathcal{A}$.

Finally, to check for a directed cycle we enumerate all pairs $(V, V') \in \mathcal{A} \times \mathcal{A}$ with $V \neq V'$ and for each pair evaluate $\textsc{IsPath}(V, V')$ and $\textsc{IsPath}(V', V)$. □

With the assertion in Proposition 3.18 the algorithmic part of the proof of the monoid version of Theorem 2.1 is finished. It remains to show the soundness and completeness of the construction. This requires purely existential statements, where no reference to effectiveness is necessary.

3.9. *Soundness*

In this section, we prove *soundness*, that is, any output we obtain by following the transitions in the NFA $\mathcal{A}$ from an initial to a final state, and then applying the corresponding maps in reverse order to the distinguished letters, gives a correct solution to the equation W_{init}.

Recall that we have chosen distinguished letters $c_1, \dots, c_m \in C$, and that if $(W, B, \emptyset, \emptyset, \mu)$ is a final state, then $W = \overline{W}$ and $W \in \# c_1 \# \cdots \# c_m \# B^*$.

Proposition 3.19. *Let $V_0 \xrightarrow{h_1} \cdots \xrightarrow{h_t} V_t$ be a path in $\mathcal{A}$ of length t, where $V_0 = (W_{\text{init}}, A, \mathcal{X}_{\text{init}}, \emptyset, \mu_{\text{init}})$ is an initial and $V_t = (W, B, \emptyset, \emptyset, \mu)$ is a final state. Then V_0 has a solution $(\text{id}_{A^*}, \sigma)$ with $\sigma(W_{\text{init}}) = h_1 \cdots h_t(W)$. Moreover, for $1 \leq i \leq m$ we have*

$$\sigma(X_i) = h_1 \cdots h_t(c_i).$$

Proof. Let $s \geq 0$ and $V_0 \xrightarrow{h_1} \cdots \xrightarrow{h_s} V_s$ be any path to some state $V_s = (W_s, B, \mathcal{X}, \theta, \mu)$ such that σ_s is a B-solution at V_s. We claim that V_0 and V_s have solutions $(\text{id}_{A^*}, \sigma)$ and $(\text{id}_{A^*} h_1 \cdots h_s, \sigma_s)$, respectively, with

$$\sigma(W_{\text{init}}) = h_1 \cdots h_s \sigma_s(W_s). \tag{3.8}$$

Claim (3.8) is trivial for $s = 0$ and for $s > 0$ it follows by induction using Lemma 3.10 or Lemma 3.9, depending on whether h_s is a substitution transition or a compression transition. Now for $s = t$ we have $\overline{W} = W$ by the definition of a final state. Since no variables occur in W, $\sigma_t = \text{id}_{B^*}$ is the (unique) B-solution of W, so $\sigma(W_{\text{init}}) = h_1 \cdots h_t(W)$.

By definition $\# X_1 \# \cdots \# X_m \#$ is a prefix of W_{init} and $\# c_1 \# \cdots \# c_m \#$ is a prefix of W for the final state V_t, but $h = \text{id}_{A^*} h_1 \cdots h_t$ is an A-morphism from B^* to A^* with $|h(c)|_{\#} = 0$ for all $c \in B$. This implies

$$\sigma(\# X_1 \# \cdots \# X_m \#) = h(\# c_1 \# \cdots \# c_m \#).$$

In particular, $\sigma(X_i) = h_1 \cdots h_t(c_i)$ for $1 \leq i \leq m$. □

Using the notation of Theorem 2.1 we have shown soundness, that is, every output we obtain is a solution in reduced words.

Corollary 3.20. *The following inclusion holds*:

$$\begin{aligned} &\{(h(c_1), \dots, h(c_m)) \in C^* \times \cdots \times C^* \mid h \in L(\mathcal{A})\} \subseteq \\ &\bigcup_{\{\mu | \mu(X) \neq 0\}} \{(\sigma(X_1), \dots, \sigma(X_m)) \\ &\in \mathbb{F}^m \mid \sigma \in \Sigma \wedge \sigma(W_{\text{init}}) = \overline{\sigma(W_{\text{init}})} \wedge \mu = \mu_0 \sigma\}, \end{aligned}$$

where Σ denotes the set of C-morphisms $\sigma : \Gamma^* \to C^*$.

Proof. Follows from Proposition 3.19. □

Corollary 3.21. *If the NFA $\mathcal{A}$ is nonempty, then there is some solution σ which maps all variables X_i to reduced words in $A_{\pm}^*$ and which satisfies $\sigma(W_{\text{init}}) = \overline{\sigma(W_{\text{init}})}$.*

If the NFA $\mathcal{A}$ contains a directed cycle, then there are infinitely many such σ.

Proof. The first part follows from Proposition 3.19.

Now assume that $\mathcal{A}$ contains a directed cycle. Then for every $t_0 \in \mathbb{N}$ we can choose a path $V_0 \xrightarrow{h_1} \cdots \xrightarrow{h_t} V_t$ from an initial state V_0 to some final state V_t with $t > t_0$. For each $0 \leq s \leq t$ define $\alpha_s = \mathrm{id}_{A^*} h_1 \cdots h_s$. Thus, $\alpha_0 = \mathrm{id}_{A^*}$. We view $\alpha_s \in \mathrm{End}(C^*)$, and let (α_s, σ_s) be the corresponding solution at V_s, which exists due to (3.8).

For every transition $V_{i-1} \xrightarrow{h_i} V_i$ which is defined either by a compression, or by a substitution of type (i), we have $\|V_{i-1}\| > \|V_i\|$. Since $\|V\| \in \mathcal{O}(n^4)$ for all states, there is a constant κ' such that every path of length $\kappa' n^4$ must include a substitution of type (ii) or (iii). Thus, we may assume that for a large enough t there are more than t_0 transitions where $V_{i-1} \xrightarrow{h_i} V_i$ is defined by a substitution of type (ii) or (iii), i.e. with $\tau(X) \in \Gamma^* C \Gamma^*$.

By the definition of $\mathcal{A}$ we have $\alpha_s(c) \neq 1$ for all $c \in C$ whenever $s < t$. (The final transition is an exception.) By Lemmas 3.9 and Lemma 3.10, we have

$$\|\alpha_0, \sigma_0\| \geq t_0.$$

since for each compression transition the weight is unchanged, and for each substitution the weight decreases, and in particular, it decreases strictly at least t_0 times. The result follows since $\alpha_0 = \mathrm{id}_{A^*}$. Hence, there infinitely many solutions σ_0. □

3.10. *Completeness*

Now we show that every solution of the equation W_{init} can be obtained from $\mathcal{A}$.

Let us fix some state $V = (W, B, \mathcal{X}, \emptyset, \mu)$ and assume that V has a solution (α, σ). We will show that if V is "small enough", then $\mathcal{A}$ contains a path $V \xrightarrow{h_1} V_1 \cdots \xrightarrow{h_t} V_t$ to some final state $V_t = (W', B', \emptyset, \emptyset, \mu')$ such that $\sigma(W) = h_1 \cdots h_t(W')$. Let us make precise what "small" means.

Definition 3.22. A state $V = (W, B, \mathcal{X}, \emptyset, \mu)$ is called *small* if

$$|W| \leq 96n + 6\,|W_{\mathrm{init}}|\,.$$

Clearly every initial state is small. Final states need not be small.

3.10.1. *Forward property of transitions*

The existence of a path $V \xrightarrow{h_1} V_1 \cdots \xrightarrow{h_t} V_t$ to some final state $V_t = (W', B', \emptyset, \emptyset, \mu')$ such that $\sigma(W) = h_1 \cdots h_t(W')$ relies on the following technical concept.

Definition 3.23. Let $V = (W, B, \mathcal{X}, \theta, \mu) \xrightarrow{h} (W', B', \mathcal{X}', \theta', \mu') = V'$ be a transition in $\mathcal{A}$ and (α, σ) be a solution at V. We say that the triple $(V \xrightarrow{h} V', \alpha, \sigma)$ satisfies the *forward property* if there exists a solution $(\alpha h, \sigma')$ at V' such that

$$\alpha\sigma(W) = \alpha h \sigma'(W').$$

By a slight abuse of language: if $V \xrightarrow{h} V'$ is a transition in $\mathcal{A}$ and the solution (α, σ) at the source V is clear from the context, then we say also that the transition $V \xrightarrow{h} V'$ satisfies the forward property. In particular, if we follow a path from V having a solution (α, σ) to some state $V' = (W', B', \emptyset, \theta', \mu')$ by transitions satisfying the forward property, then V' has some solution. But as V' uses no variables, we obtain $W' = \overline{W'}$.

Lemma 3.24. *Let* $V = (W, B, \mathcal{X}, \theta, \mu) \xrightarrow{\varepsilon} (\tau(W), B, \mathcal{X}', \theta', \mu') = V'$ *be a substitution transition* (*according to Sec.* 3.7.1) *and* $\theta(Y) = \theta'(Y)$ *for all* $Y \in \mathcal{X} \cap \mathcal{X}'$. *In each of the following cases* $(V \xrightarrow{\varepsilon} V', \alpha, \sigma)$ *satisfies the forward property*:

(1) $\sigma(X) = 1$ *and the transition* $V \xrightarrow{\varepsilon} V'$ *removes* X *by* $\tau(X) = 1$;

(2) $\theta = \emptyset, \sigma(X) = av, \mu'(X) = \mu(v)$, *and the transition* $V \xrightarrow{\varepsilon} V'$ *is defined by* $\tau(X) = aX$;

(3) $\theta(X) = \emptyset, \sigma(X) = cuv, u \in c^*, \mu'(X') = \mu(u), \mu'(X) = \mu(v)$, *and the transition* $V \xrightarrow{\varepsilon} V'$ *is defined by* $\tau(X) = cX'X$ *with* $\theta'(X') = c$;

(4) $\theta(X) = c, \sigma(X) = cu, \mu'(X) = \mu(u)$, *and the transition* $V \xrightarrow{\varepsilon} V'$ *substitutes* X *by* $\tau(X) = cX$.

Proof. Let $V \xrightarrow{\varepsilon} V'$ be defined by $\tau : M(B, \mathcal{X}, \theta, \mu) \to M(B, \mathcal{X}', \theta', \mu')$. It is enough to show that V' has a B-solution with $\sigma = \sigma'\tau$.

(1) Let σ' be the restriction of σ to $\mathcal{X}' = \mathcal{X} \setminus \{X, \overline{X}\}$. Then we have $\sigma = \sigma'\tau$.

(2) Recall that by definition of a substitution transitions, we have $\theta' = \emptyset$, too. Define σ' by $\sigma'(X) = v$ and $\sigma'(Y) = \sigma(Y)$ for $Y \neq X, \overline{X}$. Since $\mu'(X) = \mu(v)$, we obtain σ' as a morphism; and we have $\sigma = \sigma'\tau$.

(3) Define $\sigma'(X') = u$, $\sigma'(X) = v$ and $\sigma'(Y) = \sigma(Y)$ for $Y \neq X', \overline{X'}, X, \overline{X}$. Then we have $\sigma = \sigma'\tau$.

(4) Define $\sigma'(X) = u$ and $\sigma'(Y) = \sigma(Y)$ for $Y \neq X, \overline{X}$. Since $\theta(X) = c$ and σ is a solution, we have $u \in c^*$ and as τ is a morphism we have $\theta'(X) = c$, too. Then we have $\sigma = \sigma'\tau$.

In all cases it is clear that σ' is a B-solution. □

Lemma 3.25. *Let* $B' \subseteq B$ *and* $V = (h(W'), B, \mathcal{X}, \theta, \mu) \xrightarrow{h} (W', B', \mathcal{X}, \theta', \mu') = V'$ *be a compression transition* (*according to Sec.* 3.8). *If* $\sigma : \mathcal{X} \to M(B, \emptyset, \theta, \mu)$ *factors through morphisms as*

$$\sigma : \mathcal{X} \xrightarrow{\sigma'} M(B', \emptyset, \theta', \mu') \xrightarrow{h} M(B, \emptyset, \theta, \mu)$$

such that $\sigma'(X) \in c^*$ *whenever* $\theta'(X) = c$, *then* $(\alpha h, \sigma')$ *is a solution at* V' *and* $(V \xrightarrow{h} V', \alpha, \sigma)$ *satisfies the forward property.*

Proof. We have $\sigma h = h\sigma'$ and hence, $\alpha\sigma(W) = \alpha h\sigma'(W')$. □

Frequently, we cannot apply Lemma 3.25 because σ cannot be written as $h\sigma'$. The typical example is that $B' \subsetneq B$, but some $\sigma(X)$ uses a letter from $B\backslash B'$, and $h(a) = a$ for all $a \in B'$. This type of "alphabet reduction", switching from a larger alphabet B to some proper subset B', is needed only if the type relations θ, θ' are empty. Therefore the following lemma applies in this situation.

Lemma 3.26. *Let $B' \subsetneq B$ and $V = (W, B, \mathcal{X}, \emptyset, \mu) \xrightarrow{\varepsilon} (W', B', \mathcal{X}, \emptyset, \mu') = V'$ be a compression transition which is induced by the identity id_{C^*}. Thus, ε becomes the canonical inclusion of $M(B', \emptyset, \emptyset, \mu')$ into $M(B, \emptyset, \emptyset, \mu)$. In particular, $W = W'$ and μ' is the restriction of μ.*

Let (α, σ) be a solution at V. Define a B'-morphism $\beta : M(B, \emptyset, \emptyset, \mu) \to M(B', \emptyset, \emptyset, \mu')$ by $\beta(b) = \alpha(b)$ for $b \in B\backslash B'$ and $\beta(b) = b$ for $b \in B'$. Let $\sigma'(X) = \beta\sigma(X)$. Then $(\alpha\varepsilon, \sigma')$ is a solution at V' with $\alpha\sigma(W) = \alpha\varepsilon\sigma'(W')$. In particular, $(V \xrightarrow{\varepsilon} V', \alpha, \sigma)$ satisfies the forward property.

Proof. Since $\alpha : M(B, \emptyset, \emptyset, \mu) \to M(A, \emptyset, \emptyset, \mu_0)$ is an A-morphism with $\mu(a) = \mu_0(a)$ for all $a \in A$, we have $\mu\beta(b) = \mu\alpha(b) = \mu_0\alpha(b) = \mu(b)$ for all $b \in B\backslash B'$ and β is indeed a B'-morphism from $M(B, \emptyset, \emptyset, \mu)$ to $M(B', \emptyset, \emptyset, \mu')$.

Note that $M(B', \mathcal{X}, \emptyset, \mu')$ is a submonoid of $M(B, \mathcal{X}, \emptyset, \mu)$ and ε realizes the inclusion of these free monoids. Hence $W = \varepsilon(W') = W'$ as words. In particular, $\sigma(W) = \sigma(\overline{W})$ implies $\sigma'(W') = \sigma'(\overline{W}')$. Thus, $(\alpha\varepsilon, \sigma')$ solves V'.

Finally, by definition of β we have $\alpha = \alpha\beta$ because α is an A-morphism. Hence $\alpha = \alpha\varepsilon\beta$ and we obtain

$$\alpha\varepsilon\sigma'(W') = \alpha\varepsilon\sigma'(W) = \alpha\varepsilon\beta\sigma(W) = \alpha\sigma(W). \qquad \square$$

Definition 3.27. Let $\sigma : \Gamma \to C^*$ be any C-morphism and $W \in \Gamma^*$. The word W is realized as a sequence of *positions*, say $1, 2, \ldots, |W|$, and each position is labeled by a letter from Γ. If $W = u_0 x_1 u_1 \cdots x_m u_m$, with $u_i \in C^*$ and $x_i \in \Omega$, then we have $\sigma(W) = u_0\sigma(x_1)u_1 \cdots \sigma(x_m)u_m$. The positions in $\sigma(W)$ corresponding to the positions of the u_i's are henceforth called *visible*.

Given $w = \sigma(W)$, each visible position in w can be uniquely identified with a position in W, both positions having the same label in C. Following a path satisfying the forward property makes the length of the equation oscillate. In particular, throughout the *compression method* below the algorithm progresses from small state to small state, but in between the states are not necessarily small.

Proposition 3.28 shows that every solution can be found by tracing a path in $\mathcal{A}$.

Proposition 3.28. *Let $V = (W, B, \mathcal{X}, \emptyset, \mu)$ be small and let (α, σ) be a solution at V. Then $\mathcal{A}$ contains a path $V \xrightarrow{h_1} V_1 \cdots \xrightarrow{h_t} V_t$ to some final state V_t of transitions satisfying the forward property.*

In particular, if V is an initial state, then we have $\sigma(X_i) = h_1 \cdots h_t(c_i)$ for all $1 \leq i \leq m$, where $c_1, \ldots, c_m$ are the distinguished letters.

3.10.2. *Reduction of Proposition* 3.28 *to Lemma* 3.29

As a base case we let $\mathcal{X} = \emptyset$: thus, $V = (W, B, \emptyset, \emptyset, \mu)$. If V is final, then there is nothing to do. Otherwise, by definition of an extended equation, we have $W \in \#B^*\#$ and $|W|_\# = |W_{\text{init}}|_\#$. Since $\mathcal{X} = \emptyset$, we have $(\alpha, \sigma) = (\alpha, \mathrm{id}_{B^*})$ and we can write

$$W = \#u_1\# \cdots \#u_m\#u_{m+1}\#u_{m+2}\#\overline{u_{m+2}}\#\overline{u_{m+1}}\#\overline{u_m}\# \cdots \#\overline{u_1}\#.$$

Define $B_1 = A \cup \{c_1, \overline{c_1}, \ldots, c_{m+2}, \overline{c_{m+2}}\}$ as a disjoint union where $c_1, \ldots, c_m$ are the distinguished letters. Define $V_1 = (W_1, B_1, \emptyset, \emptyset, \mu_1)$ with

$$W_1 = \#c_1\# \cdots \#c_m\#c_{m+1}\#c_{m+2}\#\overline{c_{m+2}}\#\overline{c_{m+1}}\#\overline{c_m}\# \cdots \#\overline{c_1}\#.$$

Defining $\mu_1(c_i) = \mu(u_i)$ and $h_1(c_i) = u_i$ yields the desired result. Clearly, $(\alpha h_1, \mathrm{id}_{B_1^*})$ is a solution at the final state V_1 and the compression transition $V \xrightarrow{h} V_1$ satisfies the forward property. (Note that we could have some $u_i = 1$, so this is where the case distinction discussed in Remark 3.13 is needed.)

The proof of Proposition 3.28 is by induction on the weight $\|\alpha, \sigma, V\|$. It covers the rest of this section. Throughout the proof, all transitions satisfy the forward property by Lemmas 3.24–Lemma 3.26; therefore, if we know that $V_i = (W_i, B_i, \mathcal{X}_i, \theta_i, \mu_i)$ has a B_i-solution σ_i for all $1 \le i \le s$, where s is some positive integer, then we obtain $\sigma(W) = h_1 \cdots h_s \sigma_s(W_s)$ by Definition 3.23.

Preprocessing. By the base case we may henceforth assume that $\mathcal{X} \neq \emptyset$. If we have $\sigma(X) = 1$ for some variable, then we follow a substitution transition removing the variable; and we are done by induction on the weight.

Thus, without restriction, we can assume $\sigma(X) \neq 1$ for all variables. For each $X \in \mathcal{X}$, if $\sigma(X) \in aB^*$ we follow a substitution transition defined by $\tau(X) = aX$. This has the effect of *popping out* constants at the start and end of each variable, since each X comes with its involution $\overline{X}$. Since W has at most $4n$ variables present, the length of W increases by at most $8n$ and the weight $\|\alpha\sigma\|$ decreases. In case that this substitution leads to a situation where a solution maps X to the empty word, we remove X and $\overline{X}$. After that we are done by induction on the weight (since $\|\alpha\sigma\|$ is the dominant term in the lexicographic ordering), *unless* we end with $|\tau(W)| > 96n + 6\,|W_{\text{init}}|$, that is, the new state is not small. In that case we will have $96n + 6\,|W_{\text{init}}| < |\tau(W)| \le 104n + 6\,|W_{\text{init}}|$. Thus, in proving a more general statement, we will not assume that V is small, but that

$$96n + 6\,|W_{\text{init}}| < |W| \le 104n + 6\,|W_{\text{init}}|\,.$$

So far, we did not discuss the size of B. Assume that we are in the situation of Lemma 3.26: there is B' with $A \subseteq B' \subsetneq B$ such that $W \in (B' \cup \mathcal{X})^*$, then we can use Lemma 3.26; and we are done by induction on the weight. Thus, after preprocessing we may assume that all letters in $B \backslash A$ appear in W, that is, $|W|_b \ge 1$ for all $b \in B \backslash A$.

During the preprocessing we decreased the weight, but at the end of this phase V may no longer be small. Therefore, the proof of Proposition 3.28 reduces to showing the following lemma.

Lemma 3.29. *Let $V = (W, B, \mathcal{X}, \emptyset, \mu)$ be a state with a solution (α, σ) such that $\mathcal{X} \neq \emptyset$ and $|W| \leq 104n + 6\,|W_{\text{init}}|$. Then $\mathcal{A}$ contains a path of transitions satisfying the forward property to some small state $V' = (W', B', \mathcal{X}', \emptyset, \mu')$ with a solution (α', σ') such that $\|\alpha, \sigma, V\| \geq \|\alpha', \sigma', V'\|$.*

3.10.3. *Proof of Lemma* 3.29

The assertion of the lemma is trivial if V is small, that is, if $|W| \leq 96n + 6\,|W_{\text{init}}|$. Hence, we may assume $96n + 6\,|W_{\text{init}}| < |W| \leq 104n + 6\,|W_{\text{init}}|$. Let $V = (W, B, \mathcal{X}, \emptyset, \mu)$ be a state with a fixed solution (α, σ) satisfying the hypothesis of Lemma 3.29. We describe a way to find a path through $\mathcal{A}$ in terms of a procedure which "knows" the solution (α, σ).

Block compression

We employ block compression only if W contains a factor b^2, where $b \in B$ and $b \neq \#$. Otherwise we move straight to the next procedure, called *pair compression*. During the procedure we will increase the length of W by $\mathcal{O}(n)$, but at the end we will arrive at an equation where $|W'| \leq |W|$; and importantly, W' will not contain any proper factor b^2 with $b \in B$ and $b \neq \#$. We give an demonstration of this procedure in Sec. 5.

Remark 3.30. While this procedure is technical, the idea is quite simple. The goal is to eliminate long blocks b^ℓ that are visible in the equation. To do so we use transitions which replace bb by b, just two letters at a time. Before we can apply such a compression, we must ensure the length of any maximal block b^ℓ with at least part of the block visible must be *even*. So first we follow various substitution and compression transitions to arrange this.

(1) **Recording the constants with large exponents**. Due to the previous substitutions $X \mapsto bX$ in the preprocessing step, we have that for each X if $bX \leq W$ and $b'X \leq W$ are factors with $b, b' \in B$, then $\# \neq b = b'$. For each $b \in B\backslash\{\#\}$ define two sets:

$$\Lambda_b = \left\{\lambda \geq 2 \,|\, \exists\, db^\lambda e \leq \sigma(W) : d \neq b \neq e \text{ and some } b \text{ in } db^\lambda e \text{ is visible}\right\},$$

$$\mathcal{X}_b = \{X \in \mathcal{X} \,|\, bX \leq W \wedge \sigma(X) \in bB^*\}.$$

Note that

$$\sum_b |\Lambda_b| + |\mathcal{X}_b| \leq |W|. \tag{3.9}$$

By Definition 3.3 we have $\Lambda_b = \Lambda_{\overline{b}}$. Another fact is crucial: it might be that there are $X \in \mathcal{X}\backslash\mathcal{X}_b$ with $\sigma(X) \in bB^*$, but then to the left of every occurrence of X there is (the same) letter $b' \in B\backslash\{\#, b, \overline{b}\}$. In this case the block compression procedure does not touch the variable X (although it may change $\sigma(X)$). If, on the other hand, $X \in \mathcal{X}_b$, then a factor bb crosses the left border for every occurrence of X. The first b in such a factor is visible in W, the second one is not.

(2) **Introducing the type and renaming of some constants.** For each $b \in B$ with $\Lambda_b \neq \emptyset$ we introduce a *fresh* letter $c_b \in C \setminus B$ with $\mu(c_b) = \mu(b)$. In addition, for each $\lambda \in \Lambda_b$ introduce a fresh letter $c_{\lambda,b}$ with $\mu(c_{\lambda,b}) = \mu(b)$. The fresh letters are chosen such that $\overline{c_b} = c_{\overline{b}}$ and $\overline{c_{\lambda,b}} = c_{\lambda,\overline{b}}$. Note that $c_{\lambda,b}$ and c_b are just names for formal symbols realized by fresh letters in the fixed extended alphabet C.

We let $B' = B \cup \bigcup \{c_b, \overline{c}_b, c_{\lambda,b}, \overline{c}_{\lambda,b} \mid \lambda \in \Lambda_b \wedge b \in B\}$ and we introduce a type by $\theta(c_{\lambda,b}) = c_b$ for all $\lambda \in \Lambda_b$. This yields a free partially commutative monoid $M(B', \mathcal{X}, \theta, \mu)$. We define an $\mathcal{X}$-morphism

$$h : M(B', \mathcal{X}, \theta, \mu) \to M(B, \mathcal{X}, \emptyset, \mu)$$

by $h(c_{\lambda,b}) = h(c_b) = b$. Next, we modify W: in every factor $db^\lambda e$ of $\sigma(W)$ with $d \neq b \neq e$ and $\lambda \in \Lambda_b$ we replace that factor by $dc_b^\lambda e$. This defines a new word W' such that $h(W') = W$. Note that so far, no $c_{\lambda,b}$ does appear in W'. Let $V' = (W', B', \mathcal{X}, \theta, \mu)$. Then V' is a state and we can follow the transition $V \xrightarrow{h} V'$. We have $\|V'\| < \|V\|$ since $\theta \neq \emptyset$ and this term appears before the number of constants in the weight of a state. (It might be that all b are gone, so we cannot make sure that the second component in the weight decreased.) Note that for each $\lambda \in \Lambda$ at least one position labeled by c_b is visible in W.

We rename $V' = (W', B', \mathcal{X}, \theta, \mu)$ as $V = (W, B, \mathcal{X}, \theta, \mu)$ and rename the solution as (α, σ).

(3) **Splitting the variables starting with special constants.** We skip this step if $\mathcal{X}_b = \emptyset$ for all b. Otherwise, for each $b \in B$ and $X \in \mathcal{X}_b$ we write $\sigma(X) = c_b^\ell w$ for some $\ell \geq 1$ with $w \notin \{b, c_b\} B^*$. We split the variable X by defining $\tau(X) = c_b X' X$ where $X' = X'_{b,X} \in \Omega \setminus \mathcal{X}$ is a fresh variable, which is assigned a type $\theta'(X') = c_b$. Moreover, we let $\mu'(X') = \mu(c_b)^{\ell-1}$, $\mu'(X) = \mu(w)$, $\sigma'(X') = c_b^{\ell-1}$, and $\sigma'(X) = w$. The new set of variables is a disjoint union

$$\mathcal{X}' = \mathcal{X} \cup \{X'_{b,X}, \overline{X'_{b,X}} \mid b \in B \wedge X \in \mathcal{X}_b\}.$$

We obtain a new state $V' = (\tau(W), B, \mathcal{X}', \theta', \mu')$ and a morphism

$$\tau : M(B, \mathcal{X}, \theta, \mu) \to M(B, \mathcal{X}', \theta', \mu').$$

The morphism τ defines a substitution transition $V \xrightarrow{\varepsilon} V'$ which pops a letter. The new solution at V' is (α, σ').

We rename $V' = (\tau(W), B, \mathcal{X}', \theta', \mu')$ as $V = (W, B, \mathcal{X}, \theta, \mu)$ and rename the solution as (α, σ). The next step introduces the letters $c_{\lambda,b}$ into W and $\sigma(W)$.

(4) **Identifying a position in each block $dc_b^\lambda e$.** We represent $W \in M(B, \mathcal{X}, \theta, \mu)$ by any word in $(B \cup \mathcal{X})^*$. For each letter c_b, we scan the word $\sigma(W)$ from left to

right and stop at each occurrence of a factor $dc_b^{\lambda}e$ where $\lambda \in \Lambda_b$ and $d \neq c_b \neq e$. At the stop we do the following.

- If at least one of the c_b's in this block is visible in W, then choose the leftmost corresponding visible position in W, and replace the label c_b at this visible position by $c_{\lambda,b}$. In $\sigma(W)$, replace $dc_b^{\lambda}e$ by $dc_{\lambda,b}c_b^{\lambda-1}e$. If no position of the c_b's in this block is visible in W, then we make no change.

Thus, from left to right, we transform the word W into an element $W' \in M(B, \mathcal{X}', \theta, \mu)$ and simultaneously $\sigma(W)$ into an element $\sigma'(W') \in M(B)$. We obtain a new state $V' = (W', B, \mathcal{X}, \theta, \mu)$ and we can follow the arc $V \xrightarrow{h} V'$ where h is the $\mathcal{X}$-morphism defined by a renaming $h(c_{\lambda,b}) = c_b$. Note that $\|V\| > \|V'\|$ since for each $c_{\lambda,b}$ a factor $c_{\lambda,b}c_b$ appears in W', so there are more letters visible in W' than in W, which decreases the second component in the weight of an extended equation. At V' we obtain a new solution (α, σ'); and as usual, we rename $V' = (\tau(W), B, \mathcal{X}', \theta', \mu')$ as $V = (W, B, \mathcal{X}, \theta, \mu)$ and rename the solution as (α, σ).

Due to partial commutation we have the following: if a factor $f \in d\left\{c_b, c_{\lambda,b}\right\}^{\ell} e$ occurs in $\sigma(W)$ with $d, e \notin \{c_b, c_{\lambda,b}\}$, then we have $\ell = \lambda \in \Lambda_b$, and $f = dc_{\lambda,b}c_b^{\lambda-1}e \in M(B, \emptyset, \theta, \mu)$. Moreover, if $\theta(X) = c_b$, then X commutes with the letter c_b, but X does not commute with any $c_{\lambda,b}$.

(5) **The block compression.** As long as there exists a letter c_b which occurs in $\sigma(W)$, perform the following loop, which also finishes the block compression. During the following loop we maintain the invariant: if $dc_{\lambda,b}c_b^{\ell}e$ and $d'c_{\lambda,b}c_b^{\ell'}e'$ are factors of $\sigma(W)$ with $d \neq c_b \neq e$ and $d' \neq c_b \neq e'$, then $\ell = \ell'$ and $\sigma(W)$ contains a factor $\overline{d}\,\overline{c_{\lambda,b}}\,\overline{c_b}^{\,\ell}\,\overline{e}$ as well. During the loop we perform various times a renaming in order to keep the notation V and (α, σ) at the current states. Initially we define a list

$$\Lambda_B = \{b \in B \,|\, \Lambda_b \neq \emptyset\}\,.$$

while $\Lambda_B \neq \emptyset$ **do**

(a) For some $b \in \Lambda_B$ remove b and $\overline{b}$ from Λ_B;

(b) Let $c = c_b$ and for all $\lambda \in \Lambda_b$ abbreviate $c_{\lambda,b}$ as c_λ.

(c) **while** $|\sigma(W)|_c \geq 1$ **do**

(i) For all X with $\theta(X) = c$ where $|\sigma(X)|$ is odd, follow a substitution transition of type $X \mapsto cX$. Hence, we may assume that $|\sigma(X)|$ is even for all X with $\theta(X) = c$.

(ii) Remove all X from $\mathcal{X}$ where $\sigma(X) = 1$. Observe, if there remains a variable X with $\theta(X) = c$, then $\sigma(W)$ contains a factor c^2.

(iii) For all c_λ where $\sigma(W)$ contains a factor $dc_\lambda c^{\ell}e$ where $d \neq c \neq e$ and ℓ is odd, follow a compression transition with $h(c_\lambda) = cc_\lambda$.
In order to see that this is possible observe that for every occurrence of such a factor $dc_\lambda c^{\ell}e$ there are only two possibilities. Either none of

the positions of $c_\lambda c^\ell$ are visible in W, or the position of c_λ is visible in W. Moreover, c commutes with c_λ and with all X where $\theta(X) = c$; and $|\sigma(X)|$ is even for those X. Thus, wherever c_λ is visible in W, the factor cc_λ is visible in $W \in M(B, \mathcal{X}, \theta, \mu)$.

Still, we need to be more precise in order to guarantee a weight reduction. The $\mathcal{X}$-morphism defined by $h(c_\lambda) = cc_\lambda$ leads to new element $W' \in M(B, \mathcal{X}, \theta, \mu)$ and a new solution $(\alpha h, \sigma')$. In case that no letter c occurs in $\sigma'(W')$ anymore, the letter c and the type becomes useless. Thus, if $|\sigma'(W')|_c = 0$, then we actually follow a compression transition

$$V \xrightarrow{h} (W', B', \mathcal{X}, \theta', \mu)$$

where $B' = B\backslash\{c, \overline{c}\}$ and hence $|\theta'| < |\theta|$. Nevertheless $\|V\| > \|V'\|$ since $|W'| < |W|$ due to compression.

(iv) If there exists a variable X with $\theta(X) = c$, then we know $\sigma(X) = c^2c^\ell$ where ℓ is even. We follow a substitution arc defined by $X \mapsto c^2X$ in order to guarantee that a factor c^2 becomes visible in W.

(v) Due to the previous steps: either we have $c \notin B$ or W contains a visible factor c^2. In the first case, we skip this step. Thus, we assume that W contains a visible factor c^2. Now, if $\sigma(W)$ contains a factor $dc_\lambda c^\ell e$ where $d \neq c \neq e$, then ℓ is even; and if $\theta(X) = c$, then $\sigma(X) = c^j$ and j is even, too. Thus we can follow a compression transition defined by $h(c) = c^2$. This leads to a new equation W' with $h(W') = W$ and new solution $\sigma'(W')$ and the number of occurrences of c and $\overline{c}$ is halved. Note that $\|V\| > \|V'\|$ since W contains a factor c^2. Hence, $|W| > |W'|$. Rename the parameters to $V, W, B, \mathcal{X}, \theta, \mu, \alpha, \sigma$.

endwhile

(d) Rename all c_λ by $c_{\lambda,b}$.

endwhile

Space requirements for the block compression

Let us show that the block compression can be realized inside $\mathcal{A}$.

Lemma 3.31. *Let $V = (W, B, \mathcal{X}, \emptyset, \mu)$ be the state after preprocessing, when we enter "block compression", and let $V' = (W', B', \mathcal{X}', \emptyset, \mu')$ be the state at the end of block compression. Then V', as well as all intermediate states between V and V', are in $\mathcal{A}$. Moreover, $|W'| \leq 104n + 6\,|W_{\text{init}}|$.*

Proof. At the end of block compression we have $\mathcal{X}' \subseteq \mathcal{X}$, and each visible position of the new letter $c_{\lambda,b}$ occupies a position where some letter b was visible in W. Thus, $|W'| \leq |W| \leq 104n + 6\,|W_{\text{init}}|$.

To show that the procedure stays inside $\mathcal{A}$ we calculate the maximum length of an intermediate equation during the process. We start block compression with $|W| \leq 104n + 6\,|W_{\text{init}}|$, and $|\mathcal{X}| \leq 4n$. In step (3) we add at most $8n$ new variables X' and at most $8n$ constants (we may substitute a variable X by $aX'XX''b$ in the case that $\sigma(X) = a^{\ell}wb^{\ell'}$). So the length of the intermediate equation at this step is at most $104n + 6\,|W_{\text{init}}| + 16n = 120n + |W_{\text{init}}|$. The only other step of block compression that adds length to the equation during the inner while-loop in step (5).

We start this loop with $|W| \leq 120n + |W_{\text{init}}|$ and with at most $8n$ typed variables (the variables that were added in step (3)). We perform the loop at step (5c) with one letter $c \in \Lambda_B$ fixed.

In step (i) we pop at most one c letter for each typed variable, and in step (ii) we pop c^2 for each typed variable, so we add at most $3 \cdot 8n = 24n$ c's, and then in step (v) we halve the number of c's, so overall we add at most $12n$ $c's$. We repeat this loop until all c's are eliminated. In each iteration we add at most $24n$ new c letters, but then divide the total number of c letters by 2. If we just consider the number of new c letters added from the start of the while loop, we see that after each iteration the number of new c letters remaining is at most:

Iteration	Number before step (i)	Number added	Number before step (v)	Number after step (v)
1	0	$24n$	$24n$	$12n$
2	$12n$	$24n$	$36n$	$18n$
3	$18n$	$24n$	$42n$	$21n$
4	$21n$	$24n$	$45n$	$23n$
5	$23n$	$24n$	$48n$	$24n$

Thus the total length of W is never more than

$$120n + 6\,|W_{\text{init}}| + 48n = 168n + 6\,|W_{\text{init}}|\,. \tag{3.10}$$

Since this call of the inner while-loop eliminates all occurrences of the letter c, at the end of each call the length of W returns to being bounded above by $120n + 6\,|W_{\text{init}}|$, when we repeat the while-loop at step (5c) for another constant in Λ_B, until $\Lambda_B = \emptyset$. Thus all states are in $\mathcal{A}$. □

For the final state $V' = (W', B', \mathcal{X}', \emptyset, \mu')$ the type relation is empty. If V' is small, that is, $|W'| \leq 96n + 6\,|W_{\text{init}}|$, then Lemma 3.29 is shown. Thus, without restriction we again have

$$96n + 6\,|W_{\text{init}}| < |W'| \leq 104n + 6\,|W_{\text{init}}|\,.$$

Pair compression

After block compression we run *pair compression*, following essentially the formulation of Jeż's original procedure [12]. We start a pair compression at a state $V_p = (W, B, \mathcal{X}, \emptyset, \mu)$ where we have:

- $|W|_b \geq 1$ for all $b \in B \backslash A$.
- $96n + 6\,|W_{\text{init}}| < |W| \leq 104n + 6\,|W_{\text{init}}|$.
- W doesn't contain any proper factor b^2 with $b \in B \backslash \#$.
- The current solution is denoted by (α, σ).

The goal of the process is to end at a state $V_q = (W'', B', \mathcal{X}'', \emptyset, \mu'')$ with $|W''| \leq 96n + 6\,|W_{\text{init}}|$ by some path satisfying the forward property and without increasing the weight. Moreover, there will be no types in this phase. Note that the constraints make sure that $\sigma(X)$ does not contain any factor $a\overline{a}$, but we cannot rule out that W contains such factors. However, the number of $a\overline{a}$ factors remains bounded by $|W_{\text{init}}|$, since they can only occur after preprocessing W_{init}.

Consider all partitions $B \backslash \{\#\} = L \cup R$ such that $b \in L \Leftrightarrow \overline{b} \in R$. Note that there is no overlap between factors $ab, cd \in LR$ unless $ab = cd$. Moreover

$$ab \in LR \Leftrightarrow \overline{b}\overline{a} \in LR.$$

For each choice of (L, R) we count the number positions in W where some factor $ab \in LR$ with $\overline{a} \neq b$ begins. We intend to compress all these factors into single letters.

Remark 3.32. We choose and fix one of the partitions (L, R) such that the number of factors $ab \in LR$ in $\sigma(W)$ such that $\overline{a} \neq b$ and at least one of a or b visible is maximal.

We say that $ab \in LR$ is *crossing* if W contains either a factor aX with $\sigma(X) \in bB^*$ or a factor $\overline{b}X$ with $\sigma(X) \in \overline{a}B^*$ (or both). In the first phase we run the following procedure.

Uncrossing. Create a list $\mathcal{L} = \{X \in \mathcal{X} \,|\, \exists\, b \in R : \sigma(X) \in bB^*\}$.
For each $X \in \mathcal{L}$:

- choose $b \in R$ such that $\sigma(X) \in bB^*$ and follow a substitution transition $X \mapsto bX$.

This concludes the "uncrossing"; and, as done previously we rename the parameters to $V, W, B, \mathcal{X}, \mu, \alpha, \sigma$.

Above, when we follow $X \mapsto bX$ with $b \in R$, then automatically $\overline{X}$ is replaced with $\overline{X}\,\overline{b}$, and $\overline{b} \in L$. We also have $\{X, \overline{X}\} \subseteq \mathcal{L}$ if and only if $\sigma(X) \in bB^*a$ for some $ab \in LR$. In that case we actually substituted X by bXa and $\overline{X}$ by $\overline{a}\overline{X}\,\overline{b}$. Recall that we have at most $4n$ variables in W. Thus, at this stage we have:

$$|W| \leq 104n + 6\,|W_{\text{init}}| + 8n = 112n + |W_{\text{init}}|\,. \tag{3.11}$$

The second phase begins with creating a list $\mathcal{P} = \{ab \in LR \,|\, \overline{a} \neq b\}$. After that we run the following while-loop.

while $\mathcal{P} \neq \emptyset$ **do**

(1) Define

$$B' = A \cup \{a \in B \mid |W|_a \geq 1 \vee \exists\, X \in \mathcal{X} : \sigma(X) \in aB^*\}\,.$$

If $B' \neq B$, then follow a substitution transition $V \xrightarrow{\varepsilon} (W, B', \mathcal{X}, \emptyset, \mu)$ where the label $\varepsilon = \mathrm{id}_{C^*}$ yields the inclusion of $M(B', \emptyset, \emptyset, \mu)$ into $M(B, \emptyset, \emptyset, \mu)$. Rename the parameters to $V, W, B, \mathcal{X}, \mu, \alpha, \sigma$.

(2) Select and remove some pair ab in $\mathcal{P}$. If ab does not occur as a factor in W, then do nothing, else perform the next steps.

(3) Choose a fresh letter $c = c_{ab} \in C \setminus B$ with $\mu(c) = \mu(ab)$ and let $B'' = B \cup \{c, \overline{c}\}$. Define an $\mathcal{X}$-morphism

$$h : M(B'', \mathcal{X}, \emptyset, \mu') \to M(B, \mathcal{X}, \emptyset, \mu)$$

by $h(c) = ab$.

(4) Replace in W all factors ab by c and all factors $\overline{b}\overline{a}$ by $\overline{c}$. Let $W' \in (B' \cup \mathcal{X})^*$ be the new word and $V' = (W', B'', \mathcal{X}, \emptyset, \mu')$ be the new state. We have $W = h(W')$; and hence there is a compression transition

$$V \xrightarrow{h} V'.$$

(5) Follow the compression transition $V \xrightarrow{h} V'$; and rename the parameters to $V, W, B, \mathcal{X}, \mu, \alpha, \sigma$.

endwhile

Lemma 3.33. *During the while-loop for pair compression the following properties hold.*

(1) *After the first step, where the new alphabet B' is created* (*and then renamed as B*) *we have $|B| \leq |W| + 2$.*

(2) *No factor $ab \in LR$ ever becomes crossing.*

(3) *At each step where we move from state V to V' we have $\|V\| > \|V'\|$.*

(4) *Each transition satisfies the forward property.*

Proof. (1) In the first step inside the loop, when the new alphabet B' is created, we have $|B'| \leq |W|$. Therefore, after the first renaming, we have $|B| \leq |W|$. When we define B'', we add two new letters. Hence, we obtain $|B''| \leq |W| + 2$, which yields, after renaming, $|B| \leq |W| + 2$. This property persists during subsequent loops.

(2) We have to show that no factor $ab \in LR$ ever becomes crossing. To see this, consider the alphabet reduction by following the transition $V \xrightarrow{\varepsilon} (W, B', \mathcal{X}, \emptyset, \mu)$ with $B' \neq B$. It involves replacing every letter $a \in B \setminus B'$ by $\alpha(a)$ according to Lemma 3.26. The potential problem is that we might have $a \in L$, but $\alpha(a)$ starts with a letter in R, so we might create new LR factors. However as B' contains all letters a where $\sigma(X) \in aB^*$ for some X, we never introduce any new crossing pairs.

(3) The assertion $\|V\| > \|V'\|$ is trivial.

(4) The transition $V \xrightarrow{\varepsilon} (W, B', \mathcal{X}, \emptyset, \mu)$ with $B' \neq B$ satisfies the forward property by Lemma 3.26. In order to see that $V \xrightarrow{h} V'$ satisfies the forward property when we have $h(c) = ab$ we proceed as follows. As done for W, also replace in $\sigma(W)$ all factors ab by c and all factors $\overline{b}\overline{a}$ by $\overline{c}$. Since ab is not crossing, we find a B'-morphism

$$\sigma' : M(B', \mathcal{X}, \emptyset, \mu')^* \to M(B', \emptyset, \emptyset, \mu')$$

such that $\sigma(X) = h\sigma'(X)$ for all variables X. Thus, we obtain $(\alpha h, \sigma')$ as a solution at V' □

Lemma 3.34. *Let $V_p = (W, B, \mathcal{X}, \emptyset, \mu)$ be a state in $\mathcal{A}$ with a solution (α, σ) where $96n + 6\,|W_{\text{init}}| < |W| \leq 104n + 6\,|W_{\text{init}}|$ such that W does not contain any factor d^2 for $\# \neq d \in B$. Let (L, R) be the partition with $B \setminus \{\#\} = L \cup R$ according to the choice made in Remark* 3.32. *Then pair compression on V_p leads to a state $V_q = (W'', B', \mathcal{X}, \emptyset, \mu'')$ with $|W''| \leq 96n + 6\,|W_{\text{init}}|$, that is, the state V_q is small. Moreover, the intermediate steps of the pair compression algorithm are performed within $\mathcal{A}$.*

Proof. Recall that the NFA $\mathcal{A}$ is trim. Hence, there is a path

$$V_0 \xrightarrow{h_1} \cdots V_{p-1} \xrightarrow{h_p} V_p$$

from an initial state with the appropriate μ to V_p. Let $V_i = (W_i, B_i, \mathcal{X}_i, \theta_i, \mu_i)$. We perform the following *marking* process. The idea is that we wish to mark all constants in the W_i which could possibly give rise to a factor $a\overline{a}$ in W. These factors can arise in exactly two ways: the initial equation may be unreduced to start with, or from a substitution (for example, we may have aX or YZ factors of the initial equation and we pop $X \to aX$ or $Y \to Ya$, $Z \to \overline{a}Z$).

(1) In $W_0 = W_{\text{init}}$ we mark all letters (both constants and variables).
(2) If $V_{i-1} \xrightarrow{\varepsilon} V_i$ is a substitution transition, $W_i = \tau(W_{i-1})$ and the positions with constants in W_{i-1} are mapped to positions with constants in W_i. We mark constants in W_i that come from marked constants in W_{i-1}, and if $\tau(X) \in a\Gamma^*$ and X is marked in W_{i-1}, we mark the newly added a on the left of the variable X in W_i, and leave X unmarked. If $\tau(Y) = Y$ and Y is marked in W_{i-1}, we leave Y marked in W_i. Note that in this way each marked variable gives rise to exactly one marked letter.
(3) If $V_{i-1} \xrightarrow{h} V_i$ is a compression transition, then we have $h(W_i) = W_{i-1}$. Mark a constant c in W_i if it is mapped by h to an occurrence of a factor containing a marked position in W_{i-1}.

Note that since the pair compression procedure is always preceded by the preprocessing step above, we can assume that every variable X in W_{init} has been replaced by aX where a is marked, so in V_p the word W contains at most $|W_{\text{init}}|$ marked constants and no marked variables.

When we run the pair compression procedure on W we cannot compress pairs $a\overline{a}$, or pairs containing variables. If we now mark all variables present in W, then we are allowed to compress any pairs of letters in W that are unmarked. After marking the variables we have at most $2\,|W_{\text{init}}|$ marked letters in W.

Let us factor the word $W \in (B \cup \mathcal{X})^*$ as $W = x_0u_1x_1 \cdots u_\ell x_\ell$, where ℓ is chosen to be maximal that for all $1 \leq i \leq \ell$ we have:

(1) $x_i \in (B \cup \mathcal{X})^*$.
(2) $u_i \in (B\backslash\{\#\})^*$ and u_i does not contain any marked position.
(3) The length of each u_i is exactly 3.

The factorization enjoys the following properties.

- Since all #'s are marked, we have $x_0 \neq 1 \neq x_\ell$. Some other x_i can be empty.
- Since we require $|u_i| = 3$ it may be that x_i contains for each marked position also two unmarked position. The exception is the first position in x_0. Hence, we obtain
$$\sum_{0 \leq i \leq \ell} |x_i| \leq 3(2\,|W_{\text{init}}|) - 2 \leq 6\,|W_{\text{init}}|\,.$$
- Since $|W| - 6\,|W_{\text{init}}| > 96n$, the previous line yields
$$\ell > 32n.$$

Consider the word W' which was obtained via the substitution transitions, but before the compression of factors $ab \in LR$ into single letters. The increase in length, which is $|W'| - |W|$, comes from the substitution transitions $X \mapsto bX, \overline{X} \mapsto \overline{X}\,\overline{b}$ with $X \in \mathcal{L}$, so the length goes up by at most $8n$. Note that the u_i factors do not change, only the x_i factors do. Hence W' has the factorization $W' = y_0u_1y_1 \cdots u_\ell y_\ell$ with $y_i \in (B \cup \mathcal{X})^*$ and
$$|y_0 \cdots y_\ell| \leq |x_0 \cdots x_\ell| + 8n. \tag{3.12}$$

Finally, let W'' be the word obtained after pair compression has been performed. The word W'' is the compression of some word $y_0v_1y_1 \cdots v_my_m$ where each v_i is the result of the compression restricted to u_i.

Each u_i can be written as $u_i = abc$ with $a, b, c \in B$. Since W did not contain any proper factor d^2 with $d \in B$ by hypotheses (and as we have performed block compression first), we know $a \neq b \neq c$. Moreover, we cannot have $\overline{a} = b$ or $\overline{c} = b$ because in every occurrence of $b\overline{b}$ in W at least one position is marked.

Assume for a moment that membership to L or R was defined uniformly at random. That is for each $\# \neq a \in B$ the probability for $a\overline{a} \in LR$ is $\frac{1}{2}$ and independent of the other events "$b\overline{b} \in LR$" for $a \neq b$.

There are two possibilities: either $b \in L$ or $b \in R$. In the first case, either $c \in R$ or $c \in L$, and in the second case either $a \in L$ or $a \in R$. Each event

$bc \in LR, bc \in LL, ab \in LR, ab \in RR$ has probability $\frac{1}{4}$, so with probability $\frac{1}{2}$ one pair in the factor u_i is compressed: thus the expected length of a factor v_i is $\mathrm{E}\left[|v_i|\right] = \frac{3}{2} + \frac{2}{2} = \frac{5}{2}$. By linearity of expectation, we obtain

$$\mathrm{E}\left[|v_1 \cdots v_\ell|\right] = \frac{5}{2}\ell. \tag{3.13}$$

Thus if the partition (L, R) were chosen at random, we expect the length of the word $u_1 \cdots u_\ell$ to decrease from 3ℓ to $\frac{5}{2}\ell$ or less, that is, we expect at least $\frac{1}{6}\ell$ factors u_i are compressed (each v_i has length either 2 or 3). But in Remark 3.32 we made the best choice of compressing a maximal number of pairs in W'. This means at least $\frac{1}{6}\ell$ factors of W' are compressed. Hence, for the actual pair compression, we may estimate the length of W'' as follows.

$$\begin{aligned}
|W''| &\leq |x_0 \cdots x_\ell| + 8n + \frac{5}{2}\ell && \text{since } \frac{1}{6}\ell \text{ factors are compressed} \\
&= |W| + 8n - \frac{\ell}{2} && \text{since } |W| = |x_0 \cdots x_\ell| + 3\ell \\
&\leq |W| - 8n && \text{since } \ell > 32n \\
&\leq 96n + 6\,|W_{\mathrm{init}}| && \text{since } |W| \leq 104n + 6\,|W_{\mathrm{init}}|.
\end{aligned}$$

Since $|W''| \leq 96n + 6\,|W_{\mathrm{init}}|$, the last state $V_q = (W'', B', \mathcal{X}, \emptyset, \mu'')$ is small.

□

A linear bound on the size of C is evident from the proofs above and an explicit bound is given next. Thus, we have shown Lemma 3.29.

3.10.4. *The size of the extended alphabet C: the choice of κ*

The longest equation W we needed to establish completeness occurs during block compression, where we found that $|W| \leq 168n + 6\,|W_{\mathrm{init}}|$ (3.10). Combining this with $|W_{\mathrm{init}}| \leq 6n$ (3.2) we obtain

$$|W| \leq 168n + 36n = 204n. \tag{3.14}$$

The largest alphabet we ever needed during block and pair compression was less than

$$3 \cdot (|A_+| + |W|) \leq 3 \cdot (n + 204n) = 3 \cdot 205n = 615n.$$

Thus, we can choose κ such that

$$|C| = \kappa \cdot n = 615n. \tag{3.15}$$

3.10.5. *Finishing the proof of Theorem* 2.1 *in the monoid case*

Lemma 3.29 implies Proposition 3.28 by the reduction in Sec. 3.10.2. This in turn proves (2.3) in Theorem 2.1 in the monoid case $\mathbb{M}(A) = A^*$. Clearly, $\{(h(c_1), \ldots, h(c_m)) \in C^* \times \cdots \times C^* \mid h \in L(\mathcal{A})\}$ is empty if and only if $L(\mathcal{A}) = \emptyset$. It remains to show that $\mathcal{A}$ contains a directed cycle if and only if (U, V) has infinitely many solutions. If there is no cycle, then $L(\mathcal{A})$ is finite and (U, V) can have only finitely many solutions. The converse has been shown in Corollary 3.21.

4. Proof of Theorem 2.1 in the Group Case: $\mathbb{M}(A) = \mathbf{F}(A_+)$

The proof is a reduction to the monoid case. Recall that $A = A_\pm \cup \{\#\}$, $\mathbb{F}$ is the subset of reduced words in $A_\pm^*$, and $\pi : A^* \to \mathrm{F}(A_+)$ is the canonical projection.

We start with an equation (U, V) in the free group $\mathrm{F}(A_+)$, where $U, V \in (A \cup \mathcal{X})^*$, $\mathcal{X} = \left\{X_1, \overline{X_1}, \ldots, X_m, \overline{X_m}\right\}$, and solutions are A-morphisms $\sigma : (A \cup \mathcal{X})^* \to \mathbb{F}$ such that $\pi\sigma(U) = \pi\sigma(V)$. In a first phase we transform the equation (U, V) into a system of triangular equations, where *triangular* means $1 \leq |UV| \leq 3$. We may assume $UV \neq 1$. If $|UV| \leq 3$, then the equation is already triangular. Hence, let us assume $|UV| \geq 4$. Since we are in the group case we may also assume $|V| = 1$. Write $U = x_1 \cdots x_p$ with $x_i \in A \cup \mathcal{X}$ and $p \geq 3$. Next, we introduce a new variable X and replace $x_1 \cdots x_p = V$ by the system

$$x_1 \cdots x_{p-1} = X \wedge X x_p = V.$$

We iterate until the system is triangular. The procedure introduces more variables, but it does not change the set of solutions. More formally, if $\{(U_i, V_i) \,|\, 1 \leq i \leq t\}$ is the system of triangular equations we obtained above, then

$$\begin{aligned}&\{(\sigma(X_1), \ldots, \sigma(X_m)) \in \mathbb{F} \times \cdots \times \mathbb{F} \mid \pi\sigma(U) = \pi\sigma(V)\}\\ &\quad = \{(\sigma(X_1), \ldots, \sigma(X_m)) \in \mathbb{F} \times \cdots \times \mathbb{F} \,|\, \forall\, 1 \leq i \leq t : \pi\sigma(U_i) = \pi\sigma(V_i)\}.\end{aligned}$$

The crucial step in our reduction is to switch from solutions over free groups to solutions over free monoids with involution. We do this using the following lemma, whose geometric interpretation is simply that the Cayley graph of a free group (over standard generators) is a tree.

Lemma 4.1. *Let x, y, z be reduced words in $A_\pm^*$. Then $xy = z$ holds in the group $F(A_+)$ (i.e. $\pi(xy) = \pi(z)$) if and only if there are reduced words P, Q, R in $A_\pm^*$ such that $x = PR, y = \overline{R}Q$, and $z = PQ$ holds in the free monoid $A_\pm^*$.*

Proof. The direction from right to left is trivial, whether or not P, Q, R are reduced. For the other direction there are two cases. First, xy is a reduced word. Then we can choose $P = x$, $R = 1$, $Q = y$, and we are done. Second, we have $x = x'a$ and $y = \overline{a}y'$ for some letter $a \in A_\pm$, so $x'y' = z'$ holds in the group $\mathrm{F}(A_+)$. By induction, there are reduced words P, Q, R' with $x' = PR', y' = \overline{R'}Q, z = PQ$ in $A_\pm^*$. We can define $R = R'a$, which is reduced due to the equation $x = x'a = PR'a$ and the fact that x is reduced. The result is now immediate. □

The consequence of Lemma 4.1 is that with the help of fresh variables P, Q, R we can substitute every equation $xy = z$ with $x, y, z \in \{1\} \cup A_\pm \cup \Omega$ in $\mathrm{F}(A_+)$ by

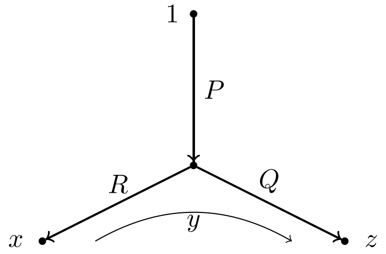


Fig. 3. Paths corresponding to geodesic words for x, y, z with $xy = z$ in the Cayley graph of $\mathrm{F}(A_+)$ with standard generators, as in Lemma 4.1. The geodesics to vertices x and z split after an initial path labeled by P.

the following three word equations to be solved over a free monoid with involution:

$$x = PR, \quad y = \overline{R}Q, \quad z = PQ. \tag{4.1}$$

More precisely, in the third phase of the transformation we replace each $U_i = V_i$, where $U_i = x_i y_i$ and $V_i = z_i$, by the three equations

$$x_i = P_i R_i, \quad y_i = \overline{R_i} Q_i, \quad z_i = P_i Q_i. \tag{4.2}$$

Thus, for $s = 3t \leq 3\,|UV|$ we obtain a new system of triangular word equations $\{(U_i', V_i') \,|\, 1 \leq i \leq s\}$ such that

$$\begin{aligned} &\{(\sigma(X_1), \ldots, \sigma(X_m)) \in \mathbb{F} \times \cdots \times \mathbb{F} \mid \pi\sigma(U) = \pi\sigma(V)\} && (4.3)\\ &\quad = \{(\sigma(X_1), \ldots, \sigma(X_m)) \in \mathbb{F} \times \cdots \times \mathbb{F} \mid \forall\, 1 \leq i \leq s : \sigma(U_i') = \sigma(V_i')\}. && (4.4)\end{aligned}$$

Note that the morphism π is not present in (4.4), since (4.4) refers to a system of equations over a free monoid with involution.

The final step is to encode the system $\{(U_i', V_i') \,|\, 1 \leq i \leq s\}$ into a single word equation (U'', V'') over the free monoid A^*, by defining

$$\begin{aligned} U'' &= U_1' \# \cdots \# U_s', \\ V'' &= V_1' \# \cdots \# V_s'. \end{aligned}$$

Thus we have deterministically reduced the equation (U, V) to the equation (U'', V''), where

$$|U''V''| \leq 15\,|UV|$$

since each $U_i' V_i'$ has length at most 3 and we have inserted $2s - 2$ copies of the letter #. This finishes the proof of Theorem 2.1 for the group case.

Remark 4.2. Since the length of the word equation obtained from a free group equation of length n is at most $15n$, an upper bound for the size of the alphabet C in the statement of Theorem 2.1 in the free group case is $615 \cdot 15n = 9225n$.

5. Example of Preprocessing, Block and Pair Compression Procedures

We conclude with a demonstration of the procedures described in Sec. 3.10 with a simple example. Suppose we have a single equation (U, V) in a free monoid with involution with

$$U = XaYbaXP \quad \text{and} \quad V = bYb^3ZQ.$$

For simplicity we have chosen an equation with no involuted letters. Suppose also that we know a solution

$$\sigma(X) = b^5, \quad \sigma(Y) = b^4a, \quad \sigma(Z) = bab, \quad \sigma(P) = ab^3a, \quad \sigma(Q) = ab^5ab^3a.$$

We depict the situation as follows:

$$\rlap{\overbrace{}^{X}}b\ \underbrace{b\ b\ b\ b\ a}_{Y}\ \overbrace{b\ b\ b\ \rlap{\underbrace{}_{Z}}b\ a}^{Y}\ b\ \underbrace{a\ \overbrace{b\ b\ b\ b\ b}^{X}\ \overbrace{a\ b\ b\ b\ a}^{P}}_{Q}.$$

For simplicity, we will ignore the rest of the word W_{init}, and focus just on the factor $U\#V$.

We first follow the preprocessing step as explained in Sec. 3.10.2. In this case we pop the first and last letter of each variable, to obtain:

$$b\ \rlap{\overbrace{}^{X}}b\ \underbrace{b\ b\ b}_{Y}\ a\ b\ \overbrace{b\ b\ b}^{Y}\ \underbrace{a}_{Z}\ b\ a\ \underbrace{b\ \overbrace{b\ b\ b}^{X}\ b\ a\ \overbrace{b\ b\ b}^{P}}_{Q}\ a.$$

Next we enter block compression. In step (1) we compute $\Lambda_a = \emptyset, \Lambda_b = \{4, 5\}$. Note that $3 \notin \Lambda_b$ since the factor b^3 is completely inside P and Q so is not visible. The block compression process will not touch this factor. We also compute $\mathcal{X}_a = \emptyset$ and $\mathcal{X}_b = \{X, Y\}$. Note that $P \notin \mathcal{X}_b$ since it is preceded by a in W.

Step (2) introduces the fresh letters $c_b, c_{4,b}, c_{5,b}$, and renames the letters b that are part of a visible block of length at least 2 as c_b:

$$c_b\ \rlap{\overbrace{}^{X}}c_b\ \underbrace{c_b\ c_b\ c_b}_{Y}\ a\ c_b\ \overbrace{c_b\ c_b\ c_b}^{Y}\ \underbrace{a}_{Z}\ b\ a\ \underbrace{c_b\ \overbrace{c_b\ c_b\ c_b}^{X}\ c_b\ a\ \overbrace{b^3}^{P}}_{Q}\ a.$$

In step (3) we split the variables $X \to X'X, Y \to Y'Y$, then remove X, Y since $\sigma(X) = 1 = \sigma(Y)$:

$$c_b\ \rlap{\overbrace{}^{X'}}c_b\ \underbrace{c_b\ c_b\ c_b}_{Y'}\ a\ c_b\ \overbrace{c_b\ c_b\ c_b}^{Y'}\ \underbrace{a}_{Z}\ b\ a\ \underbrace{c_b\ \overbrace{c_b\ c_b\ c_b}^{X'}\ c_b\ a\ \overbrace{b^3}^{P}}_{Q}\ a.$$

Note that Q does not belong to $\mathcal{X}_b$, so it does not split even though $\sigma(Q)$ starts with c_b.

Step (4) renames one of the c_b in each block in both W and $\sigma(W)$:

$$c_{5,b}\ \rlap{$\overbrace{}^{X'}$}c_b\ \underbrace{c_b\ c_b\ c_b}_{Y'}\ a\ c_{4,b}\ \overbrace{c_b\ c_b\ c_b}^{Y'}\ \underbrace{a}_{Z}\ b\ a\ \underbrace{c_{5,b}\ \overbrace{c_b\ c_b\ c_b}^{X'}\ c_b\ a\ \overbrace{b^3}^{P}}_{Q}\ a.$$

We now enter the loop in step (5). We write $c = c_b, c_\lambda = c_{\lambda,b}$:

$$c_5\ \rlap{$\overbrace{}^{X'}$}c\ \underbrace{c\ c\ c}_{Y'}\ a\ c_4\ \overbrace{c\ c\ c}^{Y'}\ \underbrace{a}_{Z}\ b\ a\ \underbrace{c_5\ \overbrace{c\ c\ c}^{X'}\ c\ a\ \overbrace{b^3}^{P}}_{Q}\ a.$$

Since $\theta(X') = \theta(Y') = c$ we pop each to make the number of c letters in each $\sigma(X)$ even:

$$c_5\ \overbrace{c\ c}^{X'}\ \underbrace{c\ c}_{Y'}\ a\ c_4\ c\ \overbrace{c\ c}^{Y'}\ \underbrace{a}_{Z}\ b\ a\ \underbrace{c_5\ \overbrace{c\ c}^{X'}\ c\ c\ a\ \overbrace{b^3}^{P}}_{Q}\ a.$$

Note that we have used the fact that X', Y' commute with c in the partially commutative monoid.

We are now at part (d) of step (6). Since c_4c^3 is a factor where the number of c letters is odd, we follow the compression transition $h(c_4) = c_4c$ to obtain:

$$c_5\ \overbrace{c\ c}^{X'}\ \underbrace{c\ c}_{Y'}\ a\ c_4\ \overbrace{c\ c}^{Y'}\ \underbrace{a}_{Z}\ b\ a\ \underbrace{c_5\ \overbrace{c\ c}^{X'}\ c\ c\ a\ \overbrace{b^3}^{P}}_{Q}\ a.$$

We now have all blocks of c inside variables and in W of even length, so we can finally follow the block compression transition $h(c) = cc$ to reduce the number of c letters by half:

$$c_5\ \overbrace{c}^{X'}\ \underbrace{c}_{Y'}\ a\ c_4\ \overbrace{c}^{Y'}\ \underbrace{a}_{Z}\ b\ a\ \underbrace{c_5\ \overbrace{c}^{X'}\ c\ a\ \overbrace{b^3}^{P}}_{Q}\ a.$$

Since there are still c letters remaining in $\sigma(W)$ we repeat the loop, and after two more iterations of the loop we obtain:

$$c_5\ a\ c_4\ \underbrace{a}_{Z}\ b\ a\ \underbrace{c_5\ a\ \overbrace{b^3}^{P}}_{Q}\ a.$$

At this point we have removed all letters c_b so the loop terminates. We reduce the alphabet by removing c_b, and remove the types. Note that we keep each $c_{\lambda,b}$

since each letter represents a different length block of b's, and therefore they are all different. Let us rename $c_{5,b} = d$ and $c_{4,b} = e$. So the equation is now:

$$d\ \ a\ \ e\ \ \underbrace{a}_{Z}\ \ b\ \ a\ \ \underbrace{d\ \ a\ \ \overbrace{b\ \ b\ \ b}^{P}}_{Q}\ \ a.$$

As promised, W contains no proper factors b^2 for any $b \in B$, so we can start pair compression.

Suppose we choose a partition of $B\backslash\{\#\}$ as $B_+ = \{\overline{a}, b, d, e\}$ and $B_- = \{a, \overline{b}, \overline{d}, \overline{e}\}$ (we suppose this choice is maximal according to Remark 3.32). In step (1) of pair compression we introduce fresh letters c_{ba}, c_{da}, c_{ea}, then in step (2) we create the list $\mathcal{L} = \{Z, \overline{P}, \overline{Q}\}$. (We will continue to ignore involutions, and focus just on a factor of W containing no involuted letters or variables). We perform uncrossing by popping a from Z and removing Z, and since we follow $\overline{P} \to \overline{bP}$ then we also follow $P \to Pb$, and similarly for Q, leading to:

$$d\ \ a\ \ e\ \ a\ \ b\ \ a\ \ \underbrace{d\ \ a\ \ \overbrace{b\ \ b}^{P}}_{Q}\ \ b\ \ a.$$

In step (3) we follow compression transitions $h(c_{ba}) = ba, h(c_{da}) = da, h(c_{ea}) = ea$ to obtain:

$$c_{da}\ \ c_{ea}\ \ c_{ba}\ \ \underbrace{c_{da}\ \ \overbrace{b\ \ b}^{P}}_{Q}\ \ c_{ba}.$$

This completes one round of the process. We then return to the preprocessing step, which gives:

$$c_{da}\ \ c_{ea}\ \ c_{ba}\ \ c_{da}\ \ \underbrace{b}_{Q}\ \ b\ \ c_{ba},$$

and then block compression would produce:

$$c_{da}\ \ c_{ea}\ \ c_{ba}\ \ c_{da}\ \ c_{2,b}\ \ c_{ba}.$$

Acknowledgments

We wish to thank Monserrat Casals-Ruiz, Artur Jeż, Ilya Kazachkov, Markus Lohrey, Alexei Miasnikov, Nicholas Touikan for helpful discussions. We are particularly indebted to the referee for numerous suggestions which greatly improved the presentation. This research was supported by the Australian Research Council (Future Fellowship FT110100178), the Swiss National Science Foundation (Professorship FN PP00P2-144681/1), and a Université de Neuchâtel Overhead Grant.

References

[1] A. V. Aho, Indexed grammars — an extension of context-free grammars, *J. ACM* **15** (1968) 647–671.

[2] P. R. Asveld, Controlled iteration grammars and full hyper-AFL's, *Inform. Control* **34**(3) (1977) 248–269.

[3] L. Ciobanu, V. Diekert and M. Elder, Solution sets for equations over free groups are EDT0L languages, in *Automata, Languages and Programming*, eds. M. Halldórsson, K. Iwama, N. Kobayashi and B. Speckmann, Lecture Notes in Computer Science, Vol. 9135 (Springer, 2015), pp. 134–145.

[4] L. Ciobanu, V. Diekert and M. Elder, Solution sets for equations over free groups are EDT0L languages, preprint (2015), arXiv:abs/1502.03426.

[5] M. Clerbout and M. Latteux, Partial commutations and faithful rational transductions, *Theor. Comput. Sci.* **34** (1984) 241–254.

[6] V. Diekert, A. Jeż and W. Plandowski, Finding all solutions of equations in free groups and monoids with involution, in *Computer Science Symp. in Russia 2014* (CSR 2014), eds. E. A. Hirsch, S. O. Kuznetsov, J. Pin and N. K. Vereshchagin, Moscow, Russia, June 7–11, 2014. *Proc.*, *Lecture Notes in Computer Science*, Vol. 8476 (Springer, 2014), pp. 1–15.

[7] V. Diekert and G. Rozenberg (eds.), *The Book of Traces* (World Scientific, Singapore, 1995).

[8] A. Ehrenfeucht and G. Rozenberg, On some context free languages that are not deterministic ET0L languages, *RAIRO Theor. Inform. Appl.* **11** (1977) 273–291.

[9] S. Eilenberg, *Automata, Languages, and Machines*, Vol. A (Academic Press, New York, London, 1974).

[10] J. Ferté, N. Marin and G. Sénizergues, Word-mappings of level 2, *Theory Comput. Syst.* **54** (2014) 111–148.

[11] S. Ginsburg and G. Rozenberg, T0L schemes and control sets, *Inform. Control* **27** (1975) 109–125.

[12] A. Jeż, Recompression: A simple and powerful technique for word equations, *J. ACM* **63**(1) (2016) 4:1–4:51.

[13] R. M. Keller, Parallel program schemata and maximal parallelism I. Fundamental results, *J. ACM* **20**(3) (1973) 514–537.

[14] O. Kharlampovich and A. Myasnikov, Elementary theory of free non-abelian groups, *J. Algebra* **302** (2006) 451–552.

[15] A. Mazurkiewicz, Concurrent program schemes and their interpretations, DAIMI Rep. PB 78, Aarhus University, Aarhus (1977).

[16] J. Messner, Pattern matching in trace monoids, in *Proc. 14th Annual Symp. on Theoretical Aspects of Computer Science* (*STACS'97*), Lübeck (Germany), 1997, ed. R. Reischuk, Lecture Notes in Computer Science, Vol. 1200 (Springer, Verlag, Heidelberg, 1997), pp. 571–582.

[17] Ch. H. Papadimitriou, *Computational Complexity* (Addison Wesley, 1994).

[18] J.-É. Pin, *Varieties of Formal Languages* (North Oxford Academic, London, 1986).

[19] W. Plandowski, An efficient algorithm for solving word equations, in *Proc. Annual Symp. Theory of Computing*, ed. J. M. Kleinberg (ACM, 2006), pp. 467–476.

[20] A. A. Razborov, On systems of equations in free groups, PhD thesis, Steklov Institute of Mathematics (1987), in Russian.

[21] A. A. Razborov, On systems of equations in free groups, in *Combinatorial and Geometric Group Theory* (Cambridge University Press, 1994), pp. 269–283.

[22] G. Rozenberg and A. Salomaa, *The Book of L* (Springer, 1986).

[23] G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages*, Vol. 1 (Springer, 1997).

[24] Z. Sela, Diophantine geometry over groups VIII: Stability, *Ann. of Math.* **177** (2013) 787–868.

[25] N. W. M. Touikan, The equation $w(x, y) = u$ over free groups: An algebraic approach, *J. Group Theory* **12**(4) (2009) 611–634.

[26] D. Wise, *From Riches to Raags*: *3-Manifolds*, *Right-Angled Artin Groups*, *and Cubical Geometry* (American Mathematical Society, 2012).