Subtree-Counting Loops

François Lemieux[1], Cristopher Moore[2], and Denis Thérien[3]

[1] Université du Québec à Chicoutimi
[2] University of New Mexico, Albuquerque, and the Santa Fe Institute
[3] McGill University, Montréal, Québec

Abstract. An important objective of the algebraic theory of languages is to determine the combinatorial properties of the languages recognized by finite groups and semigroups. In [20], finite nilpotent groups are characterized as those groups that have the ability to count subwords. In this paper, we attempt to generalize this result to finite loops. We introduce the notion of *subtree-counting* and define subtree-counting loops. We prove a number of algebraic results. In particular, we show that all subtree-counting loops and their multiplication groups are nilpotent. We conclude with several small examples and a number of open questions related to computational complexity.
Keywords: Finite loops, nilpotency, formal languages, trees and subtrees.
Subject Classification: 20N05, 68Q70

1 Introduction

A body of recent work focuses on the computational complexity of various problems involving algebraic structures, such as evaluating circuits and expressions [2–4], predicting cellular automata [15, 16], solving equations [11], and communication complexity [19]. While these algebraic problems are interesting in their own right, they also offer elegant characterizations of some low-lying complexity classes, and may even help us prove new separations between them.

Most of this work has dealt with associative structures, namely groups, semigroups and monoids, largely because the idea of a syntactic monoid is familiar from the theory of finite-state automata. However, some progress has been made in the non-associative case as well [5, 6, 13, 10, 17]. Here, concepts such as solvability generalize in several competing ways, and finding the appropriate one for a given problem can be difficult. For instance, the complexity of circuit evaluation and expression evaluation over loops is determined by two different generalizations of solvability, which we call polyabelianness (being a certain kind of product of Abelian groups) and $\mathcal{M}$-solvability (having a solvable multiplication group) [17].

In this paper, we attempt to generalize the concept of nilpotence, by building on Thérien's result that nilpotent groups are characterized by counting subwords up to some constant size [20]. In the non-associative case, we expect subwords to become subtrees, and so we explore loops which count subtrees of constant size. We find that many of the properties of nilpotent groups hold true for these loops as well.

The paper is structured as follows. After defining some terms in algebra, we review the properties of nilpotent groups and their ability to count subwords. We then introduce the notion of subtree counting and define subtree-counting loops. We prove a number of algebraic results relating these to nilpotent and $\mathcal{M}$-nilpotent loops. We conclude with several small examples and a number of open questions related to computational complexity.

2 Algebraic definitions

For the theory of quasigroups and loops, we refer the reader to [1, 7, 8, 18]. We will use the following terms, and additional definitions are given in the text.

A *groupoid* $(G, \cdot)$ is a binary operation $f : G \times G \to G$, written $f(a, b) = a \cdot b$ or simply ab. The *order* of a groupoid is the number of elements in G, written $|G|$.

A *quasigroup* is a groupoid whose multiplication table is a *Latin square*, in which each symbol occurs once in each row and each column. Equivalently, for every a, b there are unique elements a/b and $a \backslash b$ such that $(a/b) \cdot b = a$ and $a \cdot (a \backslash b) = b$; thus the left (right) *cancellation property* holds, that $bc = bd$ (resp. $cd = bd$) implies $c = d$.

An *identity* is an element 1 such that $1 \cdot a = a \cdot 1 = a$ for all a. A *loop* is a quasigroup with an identity.

A groupoid is *associative* if $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all a, b, c. A *semigroup* is an associative groupoid, and a *monoid* is a semigroup with identity. A *group* is an associative quasigroup; groups have inverses and an identity. In a group, the *order* of an element a is the smallest $p > 0$ such that $a^p = 1$ (or $pa = 0$ in an Abelian group).

Two elements a, b *commute* if $a \cdot b = b \cdot a$. A groupoid is *commutative* if all pairs of elements commute. Commutative groups are also called *Abelian*. We will use $+$ instead of $\cdot$ for products in an Abelian group, and call the identity 0 instead of 1. The simplest Abelian group is the *cyclic group* $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, the set of integers with addition mod p.

A *subgroupoid* (*subquasigroup*, *subloop*, etc.) of a groupoid G is a subset $H \subseteq G$ such that $b_1 \cdot b_2 \in H$ for all $b_1, b_2 \in H$. The subgroupoid generated by a set S, consisting of all possible products of elements in S, is written $\langle S \rangle$.

A *homomorphism* is a function ϕ from one groupoid $(A, \cdot)$ to another $(B, \star)$ such that $\phi(a \cdot b) = \phi(a) \star \phi(b)$. An *isomorphism* is a one-to-one and onto homomorphism; we will write $A \cong B$ if A and B are isomorphic.

An *equivalence relation* is a relation $\sim$ that is reflexive and transitive. Its *index* is the number of equivalence classes. An equivalence relation is a *congruence* with respect to G if $a \sim b$ and $c \sim d$ implies that $ac \sim bd$. (For infinite quasigroups, we also demand that $a/c \sim b/d$ and $a \backslash c \sim b \backslash d$.) We can then define a groupoid $G/\sim$ whose elements are $\sim$'s equivalence classes, and the obvious map from G to $G/\sim$ is a homomorphism. Conversely, for any homomorphism ϕ we can define a congruence $a \sim b$ if $\phi(a) = \phi(b)$. The groupoid $G/\sim$ is called a *quotient* or *factor* of G. A groupoid is *simple* if it has no factors other than $\{1\}$ and itself.

A *divisor* is a factor of a subgroupoid. Since a sub of a factor is the factor of a sub, the divisor relation is the transitive closure of the sub and factor relations.

The *left (right) cosets* of a subloop $H \subseteq G$ are the sets $aH = \{ah \mid h \in H\}$ and $Ha = \{ha \mid h \in H\}$ for each $a \in G$. A subloop H is *normal* if the following hold for all $a, b \in G$:

$$aH = Ha, \quad a(bH) = (ab)H, \quad \text{and} \quad (aH)b = a(Hb)$$

Then the relation

$$a \sim b \text{ if } a = bh \text{ for some } h \in H$$

is a congruence, the left and right cosets coincide, and the cosets form a quotient subloop G/H.

The *commutator* of two elements in a loop is $[a,b] = ab \,/\, ba$, i.e. the unique element such that $ab = [a,b](ba)$. The *associator* of three elements is $[a,b,c] = (ab)c \,/\, a(bc)$, i.e. the unique element such that $(ab)c = [a,b,c](a(bc))$. The subloop $\langle [G,G],[G,G,G] \rangle$ generated by all possible commutators and associators in a loop G is called the *commutator-associator subloop* or *derived subloop* G'. It is normal, and it is the smallest subloop such that the quotient G/G' is an Abelian group.

The *derived series* of a loop G is the series of normal subloops $G = G_0 \supset G_1 \supset \cdots$ where $G_{i+1} = G_i'$. A loop is *solvable of degree* k if $G_k = \{1\}$.

The *center* of a loop is the set of elements that associate and commute with everything, $Z(G) = \{c \mid cx = xc,\, c(xy) = (xc)y = x(yc) \text{ for all } x, y \in G\}$. It is a normal subloop of G, and is always an Abelian group.

The *upper central series* of a loop is $\{1\} = Z_0 \subset Z_1 \subset \cdots$ where Z_{i+1}/Z_i is the center of G/Z_i. The *lower central series* is $G = \Gamma_0 \supset \Gamma_1 \supset \cdots$ where $\Gamma_{i+1} = \langle [G,\Gamma_i],[G,G,\Gamma_i] \rangle$ is generated by the commutators and associators of Γ_i with elements of G. A loop is *nilpotent of class* k if $Z_k = G$ or if $\Gamma_k = \{1\}$; these turn out to be equivalent and k is the same in either case.

In a groupoid G, we can define left and right multiplication as functions $L_a(b) = a \cdot b$ and $R_a(b) = b \cdot a$, namely the rows and columns of the multiplication table. The *left (right) multiplication semigroup* $\mathcal{M}_L(G)$ (resp. $\mathcal{M}_R(G)$) is the set of functions on G generated by the L_a (resp. the R_a), and the *multiplication semigroup* is the set of functions generated by both. If G is a quasigroup, the L_a and R_a are permutations, so $\mathcal{M}_L(G)$, $\mathcal{M}_R(G)$ and $\mathcal{M}(G)$ are groups.

A *pseudovariety* is a class of groupoids V such that subgroups, factors, and finite direct products of groupoids in V are also in V. The solvable and nilpotent loops both form pseudovarieties.

For a given alphabet A, we define the groupoid $A^{(+)}$ as the smallest set that includes A and such that whenever f and g belong to $A^{(+)}$ then their formal product (fg) also belongs to $A^{(+)}$. It is isomorphic to the set of non-empty binary trees whose leaves are labelled with elements of A, or the set of parenthesized words generated by the grammar $S \to (S \cdot S)$, $S \to A$. The *free groupoid over* A is the set $A^{(*)} = A^{(+)} \cup \{1\}$, where 1 is the empty tree. The *free monoid over* A

is the associative version of this, namely the set A^* of finite words over A, where the product is by concatenation and 1 is the empty word.

The *yield* of a labelled tree is the word formed by reading its leaves from left to right, which is clearly a homomorphism from $A^{(*)}$ to A^*. Thus free monoids are factors of free groupoids under the congruence that identifies trees with the same yield, and so removes the non-associativity of the groupoid.

Moreover, every finite groupoid (monoid) is a divisor of the free groupoid (monoid) over some finite alphabet, i.e. it can be derived from a free object by imposing some congruence with a finite index.

The length of a word w, or the number of leaves in a tree w, is denoted $|w|$. With the exception of the free algebras, all loops used in this paper are finite.

3 Nilpotence and subword counting

Counting subwords is a well-known operation in combinatorial algebra (e.g. Ch. 6 of [14]). If x and w are two words over some alphabet A, then $|x|_w$ is the number of ways that x can be written

$$x = y_0 w_1 y_1 w_2 \cdots w_k y_k$$

where $w_1 w_2 \cdots w_k = w$ and $y_i \in A^*$. For instance, $|abab|_{ab} = 3$. (Note that many authors write $\binom{x}{w}$ instead of $|x|_w$.) If x and y are both words over A, we can define the subword counts of their product recursively:

$$|xy|_w = \sum_{\substack{u,\, v \,\in\, A^* \\ uv \,=\, w}} |x|_u |y|_v \tag{1}$$

summing over all the ways to break w into a pair of words.

In a group or monoid, we can define two words as equivalent if they evaluate to the same element. It is interesting to ask what exactly about a long word makes it evaluate to one element or another; different groups are 'sensitive' to different properties of the word. To make this precise, we say that a language $L \subset A^*$ is *recognized* by a monoid M if there is a homomorphism h from A^* to M, and a subset $K \subset M$, such that $L = h^{-1}(K)$. In other words, h maps each symbol of A to an element of M, and L is the set of words where the resulting string evaluates to an accepting element of M.

In the associative case, Thérien [20] showed that nilpotent groups are exactly those that are sensitive to counting subwords up to a certain fixed length. Specifically, define an equivalence class $\sim_k^p$ that counts, mod p, subwords of length up to k:

$$x \sim_k^p y \text{ if } |x|_u \equiv |y|_u \bmod p \text{ for all } |u| \leq k$$

It is easy to show from Equation 1 that this is a congruence. Then we have

Theorem 1 (Thérien). *If a group G has order p and is generated by m elements, it is nilpotent of class k if and only if it is a divisor of $A^*/\sim_k^p$ where A is the free monoid on m symbols. Therefore, any language recognized by G is a union of equivalence classes of $\sim_k^p$.*

Thus nilpotent groups can be characterized completely by the combinatorics of subwords.

For instance, if we take the free group with two generators a and b and count the subwords a, b, ab and ba mod 2, we get an 8-element group

$$\{1, a, b, ab, ba, aba, bab, abab\}$$

Note, for instance, that $abab = baba$ since (mod 2) both have zero a's, zero b's, one ab, and one ba. Furthermore, $abab$ commutes with every element. The reader can check that this is isomorphic to the dihedral group D_4, the symmetries of the square, where a and b correspond to reflections around axes 45° apart, and $abab$ is a 180° rotation.

Similarly, if we have two generators i and j and we count i's and j's mod 2, but combine the counts of ii, jj and ji by adding them together mod 2, we get the quaternion group $Q_8 = \{\pm 1, \pm i, \pm j, \pm k\}$. The combined count of ii, jj and ji gives the sign if $ij = k$ is defined as positive, which makes sense since $i^2 = j^2 = -1$ and $ji = -k$.

To put it differently, $\{a, b\}^* / \sim_2^2$ is a 32-element group, of which both D_4 and Q_8 are factors. (Since there are six subwords of length ≤ 2, namely a, b, aa, ab, ba and bb, in principle this group could have 64 elements. However, subword counts are not independent of each other.)

4 Subtree-counting loops

In the non-associative case, subwords presumably become subtrees. Counting subtrees is actually simpler than counting subwords, since there is only one way to divide a binary tree into smaller ones. The intuitive definition seems to be the following, where x, y, u, v are elements of the free groupoid $A^{(*)}$ and $a, b \in A$ are generators:

$$\begin{aligned}
|1|_a &= 0 \\
|a|_1 &= 0 \\
|a|_a &= 1 \\
|a|_b &= 0 \\
|(xy)|_a &= |x|_a + |y|_a \\
|(xy)|_{(uv)} &= |x|_{(uv)} + |y|_{(uv)} + |x|_u |y|_v
\end{aligned}$$

Given $p \geq 2$ and $k \geq 0$, define $x \sim_k^p y$ iff $|x|_u \equiv |y|_u$ (mod p) for all $u \in A^{(*)}$ of size at most k. The following lemma follows directly from the definition:

Lemma 1. *The relation $\sim_k^p$ is a congruence of finite index.*

Define $D_k^p = \{x \in A^{(*)} \mid x \sim_k^p 1\}$

Lemma 2. *$A^{(*)} / \sim_k^p$ is a finite loop with identity D_k^p*

Proof. We want to prove that $(xy) \sim_k^p (xz)$ implies $y \sim_k^p z$ (the proof of left cancellation is symmetric). It suffices to show that for all $s \in A^{(*)}$ of size less than k, $|(xy)|_s = |(xz)|_s$ implies that $|y|_s = |z|_s$.

The proof is by mathematical induction on $|s|$. This is clear when $|s| \leq 1$. Otherwise $s = (uv)$ and

$$\begin{aligned} |(xy)|_s &\equiv |x|_s + |y|_s + |x|_u |y|_v \\ &\equiv |x|_s + |z|_s + |x|_u |z|_v \\ &\equiv |(xz)|_s \pmod p \end{aligned}$$

Hence, there exists some constant $c = |x|_u$ such that

$$|y|_s + c|y|_v \equiv |z|_s + c|z|_v \pmod p$$

By the inductive hypothesis, we have that $|y|_v \equiv |z|_v$. So, $|y|_{(uv)} \equiv |z|_{(uv)}$ and $y \sim_k^p z$. □

Definition 1. *Loops that divide* $A^{(*)}/\sim_k^p$ *are called* subtree-counting loops of class k.

When a subtree-counting loop is express as an algebra $(G, \cdot, \backslash, /)$, we can deduce how to count subtrees in quotients x/y and $y\backslash x$:

Lemma 3. *If*

$$|x/y|_a \equiv |y\backslash x|_a \equiv |x|_a - |y|_a \text{ if } a \in A$$

$$|x/y|_{uv} \equiv |x|_{uv} - |y|_{uv} - |x/y|_u \, |y|_v$$

$$|y\backslash x|_{uv} \equiv |x|_{uv} - |y|_{uv} - |y|_u \, |y\backslash x|_v$$

Proof.

$$|x|_a \equiv |(x/y)y|_a \equiv |x/y|_a + |y|_a$$

$$|x|_{uv} \equiv |(x/y)y|_{uv} \equiv |x/y|_{uv} + |y|_{uv} + |x/y|_u \, |y|_v$$

and similarly for $y\backslash x$. □

The definition of $A^{(*)}/\sim_k^p$ implies that for any $s, t \in A^{(+)}$ satisfying $s \sim_k^p t$, we have that s and t evaluate to the same element in $A^{(*)}/\sim_k^p$. This property is generalized in the following lemma.

Lemma 4. *A loop G is subtree-counting if and only if there exists two positive integers k and p such that for any $s, t \in G^{(+)}$ satisfying $s \sim_k^p t$, we have that s and t evaluate to the same element in G.*

Proof. Let G be a subtree counting loop and consider first the special case where $G = A^{(*)}/\sim_k^p$. Without loss of generality, we can assume that $A \subseteq G$, which means that A is a basis for G. By definition, we know that for all $x, y \in A^{(+)}$, if $x \sim_k^p y$, then x and y evaluate to the same element in G.

Let $h : G \to A^{(+)}$ be any mapping such that $h(g)$ evaluates to g, for all $g \in G$ and such that $h(a) = a$ for all $a \in A$. We can extend h in the natural way to a groupoid morphism $h : G^{(+)} \to A^{(+)}$. Thus, for any $t \in G^{(+)}$, we have that t and $h(t)$ evaluate to the same element in G.

Let $u \in G^{(+)}$ be such that $|u| \leq k$, let $X(u)$ be the set of all leaves of u that are in A, and let $Y(u)$ be the set of all other leaves. Given $v \in A^{(+)}$, we are interested by the occurrences of v in $h(u)$ that contains all the leaves from $X(u)$ and at least one leaf from $h(g)$, for each $g \in Y(u)$. We denote the number of such occurrences with $\|h(u)\|_v$. We have

$$|h(s)|_v = \sum_{\substack{u \,\in\, G^{(+)} \\ |u| \,\leq\, |v|}} |s|_u \cdot \|h(u)\|_v$$

Hence, $s \sim_k^p t$ implies that $h(s) \sim_k^p h(t)$ and that s and t evaluate to the same element in G.

Consider now the general case where G divides $A^{(*)}/\sim_k^p$. Let S be a subloop of G and let $h : S \to G$ be a loop morphism. Define the function $h^{-1} : G \to S$ by choosing $h^{-1}(g)$ to be any element in S such that $h(h^{-1}(g)) = g$ and extend it in the natural way to a groupoid morphism $h^{-1} : G^{(+)} \to (h^{-1}(G))^{(+)}$.

Let $x, y \in G^{(+)}$ be such that $x \sim_k^p y$. Then, we must have $h^{-1}(x) \sim_k^p h^{-1}(y)$ and, since $h^{-1}(x)$ and $h^{-1}(y)$ belong to $A^{(*)}/\sim_k^p$, they both evaluate to the same element in S. Since h is a morphism, x and y must evaluate to the same element in G.

The other direction of the proof is immediate. $\square$

5 Properties of subtree-counting loops

Let $G = A^{(*)}/\sim_k^p$ where $p \geq 2$ and $k \geq 1$. Let $\eta : A^{(*)} \to A^{(*)}/\sim_k^p$ be the natural morphism. For $0 \leq i \leq k$, let $D_i^p = \{x \in A^{(*)} \mid x \sim_i^p 1\}$ and define $\Delta_i^p = \eta(D_i^p)$, a normal subloop of G. We can then define the following descending series of normal subloops, which we call the *subtree series:*

$$G = \Delta_0^p \supseteq \Delta_1^p \supseteq \cdots \supseteq \Delta_k^p = \{1\}$$

This series can still be defined if G is a proper divisor of $A^{(*)}/\sim_k^p$. In this case, there exists a subloop $S \subseteq A^{(*)}/\sim_k^p$ and a loop morphism $h : S \to G$. Hence, it suffices to define $\Delta_i^p = h(\eta(D_i^p) \cap S)$.

Commutators and associators are contained in various Δ_j^p because their counts of small subtrees cancel out, as the next two lemmas show.

Lemma 5. *If $x \in \Delta_k^p$ and $y \in \Delta_l^p$, then their commutator $[x, y] \in \Delta_{k+l+1}^p$.*

Proof. Let S be defined as above. We observe that any commutator y of G is the morphic image of a commutator x in S. Hence, if $x \in \eta(D_i^p)$ then $y \in \Delta_i^p$.

Consequently, it suffices to consider the case where $G = A^{(*)}/\sim_k^p$. The reader can show that $|[x,y]|_a = 0 \pmod p$ for all $a \in A$, and that

$$|[x,y]|_{uv} \equiv |x|_u\,|y|_v - |y|_u\,|x|_v - |[x,y]|_u\,|yx|_v \pmod p$$

If $|[x,y]|_w \equiv 0$ for all words shorter than uv, then the last term is zero, and the first two terms are also zero unless $|u| > k$ and $|v| > l$ or vice versa. Thus the shortest subword with nonzero count in $[x,y]$ has length at least $k+l+2$, so $[x,y] \in \Delta_{k+l+1}$. □

Lemma 6. *If $x \in \Delta_k^p$, $y \in \Delta_l^p$, and $z \in \Delta_m^p$, then their associator $[x,y,z] \in \Delta_{k+l+m+2}^p$.*

Proof. Again, it is sufficient to consider the case where $G = A^{(*)}/\sim_k^p$. As in the previous lemma, $|[x,y,z]|_a \equiv 0$ for all $a \in A$. If $u = rs$ and $v = tw$, then

$$|[x,y,z]|_{uv} \equiv |x|_r\,|y|_s\,|z|_{tw} - |x|_{rs}\,|y|_t\,|z|_w - |[x,y,z]|_u\,|(xy)z|_v \pmod p$$

(the first and second terms, respectively, disappear if $|u| = 1$ or $|v| = 1$). So the shortest word with nonzero count must be the product of three words of length greater than k, l and m respectively; its length is then at least $k+l+m+3$, so $[x,y,z] \in \Delta_{k+l+m+2}$. □

Thérien's result [20] shows that in the associative case, subtree-counting and nilpotence are the thing. In the non-associative case, this is still true in one direction:

Theorem 2. *If a loop G is subtree-counting of class k, it is nilpotent of class k.*

Proof. Assume G divides $A^{(*)}/\sim_k^p$ for some p and k. Recall the definition of the lower central series Γ_i. We have $\Delta_0^p = \Gamma_0 = G$, and $\Gamma_i \subset \Delta_i^p$ follows by induction from lemmas 3 and 4 for all $i \geq 0$. Therefore, if $\Delta_k^p = \{1\}$, then $\Gamma_k = \{1\}$. □

If the loop is commutative, we can make this stronger:

Theorem 3. *If a loop G is commutative and subtree-counting of class k, it is nilpotent of class $\lceil k/2 \rceil$.*

Proof. $\Delta_0^p = \Gamma_0 = G$, and if all commutators are the identity, then $\Gamma_i \subset \Delta_{2i}^p$ follows by induction from lemma 4 for all $i \geq 0$. Therefore, if $\Delta_k^p = \{1\}$, then $\Gamma_j = \{1\}$ where $2j \geq i$. □

Nilpotence implies solvability [7], but we can show that a loop's solvability degree is exponentially smaller than its subtree-counting class:

Theorem 4. *If a loop G is subtree-counting of class k, it is solvable of degree $\lceil \log_2(k+1) \rceil$. If it is also commutative, it is solvable of degree $\lceil \log_3(k+1) \rceil$.*

Proof. Recall the definition of the derived series G_i. We have $\Delta_0^p = G_0 = G$, and $G_j \subset \Delta_i^p$ implies $G_{j+1} \subset \Delta_{2i+1}^p$ by lemmas 3 and 4. Therefore, $G_i \subset \Delta_{2^i-1}^p$ for all $i \geq 0$, so if $\Delta_k^p = \{1\}$ then $\Gamma_j = \{1\}$ where $2^j \geq k+1$. If all commutators are the identity, then $G_j \subset \Delta_i^p$ implies $G_{j+1} \subset \Delta_{3i+2}^p$ by lemma 4. Therefore $G_i \subset \Delta_{3^i-1}^p$, so if $\Delta_k^p = \{1\}$ then $\Gamma_j = \{1\}$ where $3^j \geq k+1$. □

We close this section with a characterization of the first few classes of subtree-counting loops. Recall that the *center* of a loop is the set of elements that commute and associate with all other elements. We also say that a loop is *associator-distributive* if $[wx, y, z] = [w, y, z]\,[x, y, z]$ and similarly on the other two variables. Then:

Theorem 5. *Suppose a loop is subtree-counting of class k. If $k = 1$, it is an Abelian group. If $k = 2$, it is a group and nilpotent of class 2. If $k = 3$, it is associator-distributive and its associators are in its center.*

Proof. If $k = 1$, all commutators and associators are the identity by lemmas 3 and 4. If $k = 2$, all associators are the identity by lemma 4, so it is a group and is subword-counting of class 2. If $k = 3$, we can check that an associator $[u, v, w]$ commutes with any element x by counting subtrees. If $|s| \leq 3$,

$$|x\,[u, v, w]|_s \equiv |x|_s + |[u, v, w]|_s \equiv |[u, v, w]\,x|_s \pmod p$$

since $[u, v, w]$ contains no subtrees of size 1 or 2 by Lemma 6. A similar argument shows that an associator associates with any pair of elements. To show associator-distributivity, since $[wx, y, z]$ contains no subtrees of size 1 or 2, we just have to count subtrees of size 3. If $a, b, c \in A$, then

$$\begin{aligned} |[wx, y, z]|_{(ab)c} &\equiv |wx|_a\,|y|_b\,|z|_c \pmod p \\ &\equiv (|w|_a + |x|_a)\,|y|_b\,|z|_c \pmod p \\ &\equiv |[w, y, z]|_{(ab)c} + |[x, y, z]|_{(ab)c} \pmod p \\ &\equiv |[w, y, z][x, y, z]|_{(ab)c} \pmod p \end{aligned}$$

and similarly for $a(bc)$. □

6 $\mathcal{M}$-nilpotence and nilpotence

If we think of left and right multiplication as functions $L_a(b) = ab$ and $R_a(b) = ba$, the L_a and R_a are permutations given by the rows and columns of the multiplication table. Recall that the left (right) multiplication group of a loop G is the group generated by the L_a (resp. R_a), and the multiplication group $\mathcal{M}(G)$ is generated by both.

In [17], we used the idea of $\mathcal{M}$-solvability, the property of having a solvable multiplication group, to address the complexity of expression evaluation in loops. Here, we will say that a loop is *$\mathcal{M}$-nilpotent of class k* if its multiplication group is nilpotent of class k, and *left (right) $\mathcal{M}$-nilpotent* if its left (right) multiplication group is.

The following inclusions are known [7, 21]:

$\mathcal{M}$-nilpotent $\Rightarrow$ nilpotent $\Rightarrow$ $\mathcal{M}$-solvable $\Rightarrow$ solvable

For groups, $\mathcal{M}(G)$ is in the variety generated by G, so $\mathcal{M}$-nilpotence and nilpotence coincide. In the non-associative case, however, the $\mathcal{M}$-nilpotent loops are a proper subclass of the nilpotent ones. For instance, the following loop is nilpotent of class 2:

$$\begin{matrix} 1&2&3&4&5&6\\ 2&1&4&3&6&5\\ 3&4&5&6&2&1\\ 4&3&6&5&1&2\\ 5&6&1&2&3&4\\ 6&5&2&1&4&3 \end{matrix}$$

Its derived subloop $\{1, 2\}$ is also its center. However, its left, right and full multiplication groups are all equal to a 24-element group which is solvable of degree 2 but not nilpotent.

Then we can show that subtree-counting loops are $\mathcal{M}$-nilpotent:

Theorem 6. *If a loop G is subtree-counting of class k, then it is $\mathcal{M}$-nilpotent of class k.*

Proof. Define a *spine* as a tree where every node has at most one child which is not a leaf. An element of $\mathcal{M}(G)$ is characterized by its action on the elements of G. Since the multiplications L_a and R_a add leaves on the left and right, an element of $\mathcal{M}(G)$ corresponds to $|G|$ spines of the same shape, one for each element. For instance, $L_aR_bL_c$ corresponds to the spines $a((cx)b)$ for each $x \in G$ as shown in figure 1.

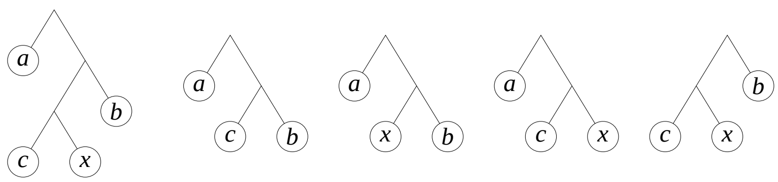

Fig. 1. An element m of the multiplication group corresponds to a set of spines, one for each element $x \in G$. The subtrees of these spines correspond to subwords of m of the same size or one smaller, depending on whether or not they include x. Here we show a spine corresponding to $m = L_aR_bL_c$ and its subtrees of size 3.

Let $m \in \mathcal{M}(G)$, and call these spines $m(x)$ for each $x \in G$. For each x, the spines $m(x)$ have two kinds of subtrees, namely those that don't include x and those that do. If a subtree of $m(x)$ of size k doesn't include x, it corresponds to a subword of m of size k. If it does include x, it corresponds to a subword of m of size $k - 1$. In either case, the subtrees of $m(x)$ are dictated by the subwords of m of the same size or smaller.

Therefore, if $m_1, m_2 \in \mathcal{M}(G)$ have the same subword counts of size k or less, then for all x their spines $m_1(x)$ and $m_2(x)$ have the same subtree counts of size k or less. Since G is subtree-counting of class k, $m_1(x) = m_2(x)$ for all $x \in G$, but this means that $m_1 = m_2$. Thus $\mathcal{M}(G)$ is subword-counting of class k, and by theorem 1 it is nilpotent of class k. $\square$

We can also obtain a partial converse to the last part of theorem 5, with a purely algebraic corollary. Recall the notion of associator-distributivity from the previous section. Then:

Theorem 7. *If a loop G has the following properties:*

- *G is associator-distributive, and*
- *all of G's associators are in its center, and*
- *there is a set of generators A for G such that the subgroup of $\mathcal{M}_R(G)$ generated by their right multiplications, $\langle\{R_a \mid a \in A\}\rangle$ is nilpotent of class k,*

then it is subtree-counting of class $\max(3, k)$*. Therefore, G is $\mathcal{M}$-nilpotent of class* $\max(3, k)$.

Proof. If we are given a tree in $G^{(*)}$, we start by rewriting it as a tree in $A^{(*)}$ by replacing elements of G with products of elements of A. Now define a *(left) comb* in $A^{(*)}$ as a tree where every node's right child, if it has one, is a leaf. Inductively, the empty tree is a comb, and ca is a comb if c is a comb and $a \in A$. Since the parenthesization of a comb is fixed, we can denote it without ambiguity by its yield, e.g. $((ab)c)d$ is simply denoted $abcd$.

Then we start by converting an arbitrary tree to a comb with the same yield which is equivalent with respect to G, keeping track of the associators as we do so. We do this inductively, first transforming subtrees of depth 2, then subtrees of depth 3, and so on. Suppose that at some point in this process we are about to transform a subtree t. If t is already a comb, there is nothing to do. Otherwise, $t = ba$ where $b = b_1 \cdots b_k$ and $a = a_1 \cdots a_l$ are two combs of size $k \geq 1$ and $l \geq 2$, where $b_j, a_i \in A$ for all j, i. To apply the transformation, we use the following:

$$\begin{aligned} ba &= b(a_1 \cdots a_l) \\ &= (b(a_1 \cdots a_{l-1}))\, a_l\, [b, a_1 \cdots a_{l-1}, a_l] \\ &\;\vdots \\ &= (b_1 \cdots b_k a_1 \cdots a_l)\, \textstyle\prod_{i=2}^{l} [b, a_1 \cdots a_{i-1}, a_i] \end{aligned}$$

Since associators are in the center of G, each one can be moved to the side of the expression as it is created.

Now since G is associator-distributive, we can write this product of associators as

$$\prod_{j=1}^{k} \prod_{i=2}^{l} \prod_{h=1}^{i-1} [b_j, a_h, a_i]$$

There is a bijection between the associators $[b_j, a_h, a_i]$ in this product and the subtrees $b_j(a_h a_i)$ of size 3 rooted at the node where b and a meet. By induction, the transformation of a tree into a left comb creates precisely one associator $[a, b, c]$ for each subtree $a(bc)$ where $a, b, c \in A$.

Thus we can convert a tree into an equivalent comb, and the product of associators it takes to do this is a function only of subtrees of size 3. Since a left comb in $A^{(*)}$ is formed by composing a series of right multiplications R_a for $a \in A$, and since these generate a nilpotent group of class k, we can evaluate the comb by counting subcombs of size k. Since the comb has the same yield as the original tree, this is the same as counting subtrees of size k in the original tree and combining subtrees of the same yield.

Thus the value of the tree is determined by counting subtrees of size $\max(3, k)$, so G is subtree-counting of this class. Finally, G is $\mathcal{M}$-nilpotent of class $\max(3, k)$ by theorem 6. □

Obviously, the third condition of theorem 7 is satisfied whenever G is right-$\mathcal{M}$-nilpotent of class k. For instance, consider the octonion loop O_{16}, which consists of 16 elements $\{\pm 1, \pm i, \pm j, \pm k, \pm E, \pm I, \pm J, \pm K\}$. Its multiplication table is

$$\begin{array}{rrrrrrrr}
1 & i & j & k & E & I & J & K \\
i & -1 & k & -j & I & -E & -K & J \\
j & -k & -1 & i & J & K & -E & -I \\
k & j & -i & -1 & K & -J & I & -E \\
E & -I & -J & -K & -1 & i & j & k \\
I & E & -K & J & -i & -1 & -k & j \\
J & K & E & -I & -j & k & -1 & -i \\
K & -J & I & E & -k & -j & i & -1
\end{array}$$

which we extend to elements with minus signs in the obvious way. Just as the quaternions are commutative up to a sign, the octonions are associative up to a sign. Since all commutators and associators are in the center $\{\pm 1\}$, O_{16} is nilpotent of class 2. Moreover, the reader can check that it is associator-distributive and its right multiplication group (which has 128 elements) is nilpotent of class 2. Therefore, it is subtree-counting of class 3, and its full multiplication group (which has 1024 elements) is nilpotent of class 3.

The reader might hope that all nilpotent loops of class 2 are associator-distributive. This is not the case, as we will show below.

7 Examples

If we take the free groupoid on one generator $\{1, a, aa, a(aa), (aa)a, \ldots\}$ and consider equivalence classes that count subtrees up to size 3 mod 2, then $\{a\}^{(*)}/\sim_3^2$ is a subtree-counting loop of class 3 with 8 elements. It is an extension of $\mathbb{Z}_2$ by $\mathbb{Z}_4$, and its multiplication table is

$$
\begin{array}{cccccccc}
1&2&3&4&5&6&7&8\\
2&1&4&3&6&5&8&7\\
3&4&1&2&7&8&5&6\\
4&3&2&1&8&7&6&5\\
5&6&8&7&3&4&2&1\\
6&5&7&8&4&3&1&2\\
7&8&6&5&1&2&4&3\\
8&7&5&6&2&1&3&4
\end{array}
\tag{2}
$$

The eight elements can be represented by $1, 2 = a((aa)a), 3 = aa, 4 = a(a(a((aa)a)))$, $5 = a$, $6 = a(a((aa)a))$, $7 = (aa)a$, and $8 = a(aa)$. In fact, there are no non-associative subtree-counting loop with fewer than 8 elements, since the smallest non-associative nilpotent loops have 6 elements, and these all have a multiplication group $\mathbb{Z}_2 \wr \mathbb{Z}_3$ of order 24 that is solvable but not nilpotent (here $\wr$ is the wreath product [12]).

Counting subtrees of size 3 mod p for larger p gives class 3 loops of size cp^2 where c appears to depend only on p mod 6:

$$
c = \begin{cases} 1 \text{ if } p \bmod 6 = 1 \text{ or } 5 \\ 2 \text{ if } p \bmod 6 = 2 \text{ or } 4 \\ 3 \text{ if } p \bmod 6 = 3 \\ 6 \text{ if } p \bmod 6 = 0 \end{cases}
$$

We have checked this for $p \le 25$, and we conjecture it is true for all p. Counting subtrees up to size 4 mod 2 gives a subtree-counting loop of class 4 with $128 = 2^7$ elements, and counting mod 3 gives $729 = 3^6$ elements.

All of these loops are generated by a single element, like the free groupoid of which they are factors. For an example with two generators, if we take the free groupoid on two generators a, b and impose the relations $a^2 = b^2 = 1$ and $xy = yx$ for all x, y, we get a subtree-counting loop of class 3 with 16 elements.

Its multiplication table is

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	1	4	3	6	5	8	7	10	9	12	11	14	13	16	15
3	4	1	2	7	8	5	6	11	12	9	10	15	16	13	14
4	3	2	1	8	7	6	5	12	11	10	9	16	15	14	13
5	6	7	8	1	2	3	4	13	14	15	16	10	9	12	11
6	5	8	7	2	1	4	3	14	13	16	15	9	10	11	12
7	8	5	6	3	4	1	2	15	16	13	14	12	11	10	9
8	7	6	5	4	3	2	1	16	15	14	13	11	12	9	10
9	10	11	12	13	14	15	16	1	2	3	4	7	8	5	6
10	9	12	11	14	13	16	15	2	1	4	3	8	7	6	5
11	12	9	10	15	16	13	14	3	4	1	2	5	6	7	8
12	11	10	9	16	15	14	13	4	3	2	1	6	5	8	7
13	14	15	16	10	9	12	11	7	8	5	6	1	2	3	4
14	13	16	15	9	10	11	12	8	7	6	5	2	1	4	3
15	16	13	14	12	11	10	9	5	6	7	8	3	4	1	2
16	15	14	13	11	12	9	10	6	5	8	7	4	3	2	1

where the generators are (say) $5 = a$ and $9 = b$. Counting mod 3, mod 4, and mod 5 gives loops of 81, 256, and 625 elements respectively.

Going back to a one-symbol alphabet and counting (mod 2) the five subtrees of *depth* 2 or less, a, aa, $(aa)a$, $a(aa)$ and $(aa)(aa)$, gives a 16-element loop

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	1	4	3	6	5	8	7	10	9	12	11	14	13	16	15
3	4	5	6	9	10	11	12	1	2	14	13	15	16	8	7
4	3	6	5	10	9	12	11	2	1	13	14	16	15	7	8
5	6	7	8	2	1	4	3	13	14	16	15	10	9	11	12
6	5	8	7	1	2	3	4	14	13	15	16	9	10	12	11
7	8	1	2	13	14	15	16	6	5	9	10	12	11	3	4
8	7	2	1	14	13	16	15	5	6	10	9	11	12	4	3
9	10	11	12	4	3	6	5	15	16	7	8	2	1	14	13
10	9	12	11	3	4	5	6	16	15	8	7	1	2	13	14
11	12	14	13	16	15	9	10	7	8	1	2	4	3	6	5
12	11	13	14	15	16	10	9	8	7	2	1	3	4	5	6
13	14	15	16	8	7	2	1	12	11	4	3	5	6	9	10
14	13	16	15	7	8	1	2	11	12	3	4	6	5	10	9
15	16	10	9	11	12	14	13	3	4	6	5	7	8	2	1
16	15	9	10	12	11	13	14	4	3	5	6	8	7	1	2

Here $\{1, 2\}$ is a normal subloop, and dividing it out gives the 8-element loop (2) above.

If we count just the balanced trees a, aa and $(aa)(aa)$ up to depth 2, we get another 8-element loop,

$$\begin{array}{cccccccc}
1&2&3&4&5&6&7&8\\
2&3&4&1&6&7&8&5\\
3&4&5&6&7&8&1&2\\
4&1&6&7&8&5&2&3\\
5&6&7&8&1&2&3&4\\
6&7&8&5&2&3&4&1\\
7&8&1&2&3&4&5&6\\
8&5&2&3&4&1&6&7
\end{array}$$

where the generator is (say) $2 = a$. This loop is not isomorphic to (2) since only two elements give the identity when squared. It is commutative but not associative, since $(22)3 = 5$ but $2(23) = 1$. However, like (2) it is an extension of $\mathbb{Z}_2$ by $\mathbb{Z}_4$.

In fact, all loop extensions of $\mathbb{Z}_2$ by $\mathbb{Z}_4$ are nilpotent and $\mathcal{M}$-nilpotent, since $\mathbb{Z}_2 \wr \mathbb{Z}_4$ is nilpotent of class 4. Similarly, all loop extensions of $\mathbb{Z}_2$ by $\mathbb{Z}_2^2$ are $\mathcal{M}$-nilpotent, since $\mathbb{Z}_2 \wr Z_2^2$ is nilpotent of class 3. We do not know if all of these are subtree-counting.

This loop also shows that, unlike the derived series and the central lower series, the subtree series can halt for a while and then continue downward. $\Delta_1 = \{1,3,5,7\}$ is generated by $3 = a^2$ and is isomorphic to $\mathbb{Z}_4$, while Δ_2 and Δ_3 coincide and are both $\{1, 5 = (aa)(aa)\}$. Finally, $\Delta_4 = \{1\}$. Thus $\Delta_0 \supset \Delta_1 \supset \Delta_2 = \Delta_3 \supset \Delta_4$. In general, counting (mod 2) balanced trees with one generator up to depth k gives a subtree-counting loop of class 2^k and size 2^{k+1}. Thus, in the non-associative case, a loop of size n can have a subtree-counting degree linear in n, whereas the nilpotence degree of a loop can be at most logarithmic in n. This suggests that determining when a given loop is *not* subtree-counting may require exponentially more computation than telling when a loop is not nilpotent.

As these examples show, we can choose to count some subset S of the set of trees of size less than or equal to k, instead of all of them. This will be a congruence, and so will give a well-defined loop, if and only if S is closed under subtrees, i.e. $uv \in S$ implies $u \in S$ and $v \in S$. For instance, we can choose to count subtrees up to a certain depth rather than a certain size; balanced subtrees up to a certain depth; left or right combs of a certain depth; and so on.

If we define loops as *(balanced) subtree-counting of depth k* in the obvious way, we have

Lemma 7. *If a loop is subtree-counting of class k, then it is (balanced) subtree-counting of depth k. If it is (balanced) subtree-counting of depth d, then it is subtree-counting of class 2^d.*

Proof. A subtree of size k is contained in a balanced subtree of depth at most k, and a subtree of depth d has size at most 2^d. □

However, a tree which is not a comb is not contained in a comb of any size, so the *subcomb-counting* loops might be a proper subclass of the subtree-counting ones.

8 Open questions

We have introduced the class of subtree-counting loops and show that it is a subclass of the M-nilpotent loops. However, we still don't know if this inclusion is strict. If so, it would be interesting to have some examples and investigate their combinatorial properties.

A more basic problem is that we have no decision algorithm to determine if a finite loop G of order g is subtree-counting. This is equivalent to determining if there exist p and k such that G divides $A^{(*)}/\sim_k^p$ for some alphabet A. If G is subtree-counting, then we can take $p = g$ and $A = G$ since it must be a morphic image of $H = G^{(*)}/\sim_k^p$. Since G/Δ_1^p is an abelian group divided by $\mathbb{Z}_p$, then g must be a multiple of p. This implies that G divides $G^{(*)}/\sim_k^g$. Finding k seems to be more difficult. However, we observe that in order to compute the number of subtrees of depth $d > 1$, it seems necessary to have some information about the number of subtrees of depth $d-1$. This suggests that the number of elements in a subtree-counting loop G of class k must be at least $\log k$ and that G must divide $G^{(*)}/\sim_{2^g}^g$. We conjecture that this is true, in which case a decision algorithm would exist.

Another set of open questions come from the theory of computational complexity, especially low-level parallel complexity classes. For instance, expressions and circuits over solvable groups can be evaluated in the classes $\mathbf{ACC}^0$ and $\mathbf{ACC}^1$, while over non-solvable groups these problems are $\mathbf{NC}^1$-complete and $\mathbf{P}$-complete respectively (see [3, 4, 15] for definitions of these classes and proofs of these results). Similarly, equations over nilpotent groups can be solved in polynomial time, while for non-solvable groups this problem is **NP**-complete [11] and for solvable groups quasipolynomial time is believed to suffice. Finally, languages defined over groups have constant multiplayer communication complexity if and only if they are nilpotent [19].

Subtree-counting loops can be shown to have many of the same complexity properties as nilpotent groups, suggesting that subtree-counting may play the same role for loops that nilpotence does for groups. However, we have not yet been able to prove the converse computational hardness results for non-subtree-counting loops. In particular, we would like to know if any expressions or programs over non-subtree-counting loops can always express the logical AND of an arbitrary number of variables. We hope that techniques from loop theory can be applied to this and other complexity-theoretic questions.

Acknowledgements. F.L and D.T. where supported by grants from FCAR (Québec) and NSERC (Canada). D.T where also supported by a grant from the von Humbolt Foundation. C.M. is grateful to McGill University for a delightful visit to Montréal, and to Molly Rose and Spootie the Cat for their support. We also thank William C. Waterhouse and Michael Kinyon for helpful conversations.

References

1. A.A. Albert, "Quasigroups I." *Trans. Amer. Math. Soc.* **54** (1943) 507–519, and "Quasigroups II." *Trans. Amer. Math. Soc.* **55** (1944) 401–419.
2. D. A. Barrington, "Bounded-width polynomial-size branching programs recognize exactly those languages in $\mathbf{NC}^1$." *J. Comput. System Sci.* **38** (1989) 150–164.
3. D.A. Mix Barrington and D. Thérien, "Finite monoids and the fine structure of $\mathbf{NC}^1$." *Journal of the ACM* **35** (1988) 941–952.
4. M. Beaudry, P. McKenzie, P. Péladeau, and D. Thérien, "Circuits with monoidal gates." *Proc. STACS* (1993) 555–565.
5. M. Beaudry and P. McKenzie, "Circuits, matrices, and nonassociative computation." *J. Comput. System Sci.* **50** (1995) 441–455.
6. M. Beaudry, F. Lemieux, and D. Thérien, "Finite loops recognize exactly the regular open languages." *Proc. 24th International Colloquium on Automata, Languages and Programming.* Lecture Notes in Computer Science 1256, Springer-Verlag, 1997, 110–120.
7. R.H. Bruck, "Contributions to the theory of loops." *Trans. Amer. Math. Soc.* **60** (1946) 245–354.
8. R.H. Bruck, *A Survey of Binary Systems.* Springer-Verlag, 1966.
9. S.R. Buss, "The Boolean formula value problem is in ALOGTIME." *Proc. 18th ACM Symp. on the Theory of Computing* (1987) 123–131.
10. H. Caussinus and F. Lemieux, "The Complexity of Computing over Quasigroups." *Proc. 14th annual FST&TCS* (1994) 36–47.
11. M. Goldman and A. Russell, "The complexity of solving equations over finite groups." *Proc. 14th Annual IEEE Conference on Computational Complexity* (1999).
12. P. Hall, *The Theory of Groups*, Macmillan, (1959).
13. F. Lemieux, *Finite Groupoids and their Applications to Computational Complexity.* Ph. D. Thesis, School of Computer Science, McGill University, Montréal (1996).
14. M. Lothaire, *Combinatorics on Words.* Encyclopedia of Mathematics and its Applications, G.-C. Rota, Ed. Addison-Wesley, 1983.
15. C. Moore, "Predicting Non-linear Cellular Automata Quickly by Decomposing Them into Linear Ones." *Physica D* **111** (1998) 27–41.
16. C. Moore, "Quasi-linear Cellular Automata." *Proceedings of the International Workshop on Lattice Dynamics, Physica D* **103** (1997) 100–132.
17. C. Moore, D. Thérien, F. Lemieux, J. Berman, and A. Drisko, "Circuits and Expressions with Non-Associative Gates." To appear in *J. Comput. System Sci.*
18. H.O. Pflugfelder, *Quasigroups and Loops: An Introduction.* Heldermann Verlag, 1990.
19. J.-F. Raymond, P. Tesson and D. Thérien, "An algebraic approach to communication complexity." *Proc. 25th International Colloquium on Automata, Languages and Programming.* Lecture Notes in Computer Science 1443, Springer-Verlag, 1998, 29–40.
20. D. Thérien, "Subword counting and nilpotent groups." In *Combinatorics on Words, Progress and Perspectives,* Larry Cummings, Ed. Academic Press, 1983, 297–306.
21. A. Vesanen, "Solvable groups and loops." *Journal of Algebra* **180** (1996) 862–876.

Addresses of the authors

François Lemieux (**Corresponding author**), Département d'informatique et de mathématique, Université du Québec à Chicoutimi, 555 boulevard de l'Université, Chicoutimi (Qc) Canada G7H 2B1. *flemieux@uqac.ca*

Cristopher Moore, Computer Science Department, The University of New Mexico, Farris Engineering Center, Room 157, Albuquerque (NM) USA 87131. *moore@cs.unm.edu*

Denis Thérien, School of Computer Science, McGill University, 3480 University Street, McConnell Engineering Building Room 318, Montréal (Qc) Canada H3A 2A7. *denis@cs.mcgill.ca*