Finite loops recognize exactly the regular open languages*

Martin Beaudry† François Lemieux‡ Denis Thérien§

Abstract

In this paper, we characterize exactly the class of languages that are recognizable by finite loops, i.e. by cancellative binary algebras with an identity. This turns out to be the well-studied class of regular open languages. Our proof technique is interesting in itself: we generalize the operation of block product of monoids, which is so useful in the associative case, to the situation where the left factor in the product is non-associative.

1 Introduction

The algebraic approach in the study of regular languages, based on considering finite monoids as language recognizers, certainly is the most powerful tool available for understanding computations realized by finite-state automata. It has developed into a rich and coherent framework to relate combinatorial descriptions of regular languages and algebraic properties of their recognizers [10, 16]. An early example of such relationship is the famous theorem of Schützenberger [24]: a subset of A^* is star-free (i.e. can be obtained from finite sets using Boolean operations and concatenation) iff it can be recognized by a group-free monoid (i.e. in which no subset forms a non-trivial group).

In much the same way that monoids can be used to recognize languages, one may also consider other types of algebras and study their computational power. For example, non-associative binary algebras, usually called groupoids, are exactly the recognizers needed for context-free languages: this relationship has been well-known in theory of tree automata (see [12]) and can be traced back to the paper of Mezei and Wright [15]. It has also been used in complexity theoretic work (e.g. see [26, 5]). In view of this connection, it is natural to try to characterize the languages recognized by various specific subclasses of finite groupoids. One such class, that has been extensively studied in the past

*Work supported by FCAR (Québec) and CRSNG (Canada)

†Département de mathématiques et d'informatique, Université de Sherbrooke, Sherbrooke (Qc) Canada, J1K 2R1,beaudry@dmi.usherb.ca

‡School of Computer Science, McGill University, 3480 rue University, Montréal (Qc) Canada, H3A 2A7, lemieux@cs.mcgill.ca

§**Corresponding author**: School of Computer Science, McGill University, 3480 rue University, Montréal (Qc) Canada, H3A 2A7, denis@cs.mcgill.ca

[7, 8, 1, 6], consists of *loops*, i.e. groupoids with an identity and for which every row and every column of the multiplication table contains every element. In [9] it was proved that any language recognized by a finite loop must be regular. The main result of our paper gives an exact characterization of which languages can be recognized by loops.

The answer is surprizing and elegant: a language $L \subseteq A^*$ can be recognized by a finite loop iff L is regular and open in the group topology on A^*. This topology, introduced by [21, 22], is the smallest one such that every morphism from A^* onto a finite group is continuous; investigations of its properties were motivated early on by several deep connections with important questions about finite monoids [17, 18]. Our result thus adds on a new perspective to a class of languages which already has a significant history.

The paper is organized as follows: in section 2, we introduce most of the relevant definitions that will be needed. In section 3, we present some tools that are useful in constructing loops to recognize languages. In section 4, we show that loops recognize only regular open languages. In section 5, we prove that every regular open language is recognizable by a finite loop. We derive some consequences of this theorem in the last section where we also present some ideas for further applications of our techniques.

2 Preliminaries

In this section, we introduce our notation and review some elementary facts about monoids and groupoids.

Let A be a finite set: we write A^* for the *free monoid* generated by A, i.e. for the set of all words of finite length over the alphabet A, concatenation being the associative operation. The *length* of a word x is denoted by $|x|$ and ϵ stands for the unique word of length 0, which is the identity element of the free monoid. A *congruence* on A^* is an equivalence relation α that is compatible with concatenation, i.e. $x_1 \alpha y_1$ and $x_2 \alpha y_2$ imply $x_1x_2 \alpha y_1y_2$. The quotient A^*/α is then an A-generated monoid, and every A-generated monoid is of this form. A language $L \subseteq A^*$ is *recognized* by the monoid M iff there exist a morphism $\phi : A^* \to M$ and a subset F of M such that $L = \{x \in A^* : \phi(x) \in F\}$; equivalently we can view the morphism as going from A^* to M^*, transforming a word x into a string of monoid elements which is then evaluated in M; in this point of view, only alphabetical morphisms need to be considered, that is morphisms mapping letters to letters. We observe that when L is recognized by M, $\phi(A^*)$ is a submonoid of M isomorphic to A^*/α for some congruence α and L is a union of α-classes. It is well-known that a language is regular iff it can be recognized by a finite monoid, i.e. iff it is a union of α-classes for some congruence α of finite index. We will say that $L \subseteq A^*$ is a group language iff L can be recognized by a finite group.

The notions above have natural counterparts in the non-associative world. A *groupoid* is given by a set and a binary operation: we will assume here that every groupoid contains a 2-sided identity element. The *free groupoid* generated by A will be denoted by $A^{(*)}$. It will be convenient to think of an element in $A^{(*)}$ as a pair (t, x) where x is a word in the free monoid and t is a rooted binary tree with $|x|$ leaves; in particular, the identity of the free groupoid is then $(\emptyset, \epsilon)$. The product of (t_1, x_1) with (t_2, x_2) is then the pair (t_1t_2, x_1x_2) where $t_1t_2 = t_1$ if t_2 is empty, $t_1t_2 = t_2$ if t_1 is empty, and otherwise $t_1t_2 = t$ is the tree obtained by joining the root of t_1 and the root of t_2 to a new node, which becomes the root of t. If $g \in A^{(*)}$ is identified in this way with a pair (t, x) we define $\mathrm{Tree}\,(g) = t$ and $\mathrm{Yield}\,(g) = x$. We say that $g_1, g_2 \in A^{(*)}$ are *yield-equivalent* if $\mathrm{Yield}\,(g_1) = \mathrm{Yield}\,(g_2)$. We can view each row and each column in the multiplication table of the groupoid G as defining a mapping from G to G. The closure of these mappings under the operation of composition is called the *multiplicative monoid* of G, denoted by $\mathcal{M}(G)$.

Congruences on $A^{(*)}$ are defined in the same way as in the associative case. If α is a congruence on $A^{(*)}$, the quotient $A^{(*)}/\alpha$ is an A-generated groupoid, and every A-generated groupoid arises in this way. A *loop* G is a groupoid whose multiplication table contains every element in each row and in each column; clearly, this will be the case iff $\mathcal{M}(G)$ is a group. Equivalently, a loop is a groupoid that is left and right cancellative, i.e. $ab = ac$ implies $b = c$ and $ba = ca$ implies $b = c$, for any $a, b, c \in G$.

We now wish to use groupoids to recognize subsets of A^*; note that if G is not associative, the notion of a morphism from A^* to G is not well-defined. We say that the language $L \subseteq A^*$ is recognized by the groupoid G if there exist an alphabetic morphism $\phi : A^* \to G^*$ and a subset F of G such that $L = \{x \in A^* : G(\phi(x)) \cap F \neq \emptyset\}$, where $G(\phi(x))$ is the set of elements of G obtained by evaluating the string $\phi(x)$ of G^* in all possible ways. Note that if G is associative, there is only one way of evaluating $\phi(x)$ and we are back to the definition given for monoids. We will say that $L \subseteq A^*$ is a *loop language* iff L can be recognized by a loop. In terms of congruences, the groupoid G recognizes the language L iff there exist an A-generated subgroupoid of G isomorphic to $A^{(*)}/\alpha$ and a subset F of this subgroupoid such that $x \in L$ iff there is some tree t such that $[(t, x)]_\alpha$ is in F. One pleasant feature of this notion of language recognition is

Lemma 2.1 *[5] L is recognizable by a finite groupoid iff L is context-free.*

The *finite group topology* on A^* is the smallest topology such that every morphism from A^* onto a finite group is continuous. It is equivalent to say that the group languages form a basis for this topology. It was first introduced by Hall [13] for the free group, and by Reutenauer for the free monoid [21, 22]. Connections were soon discovered between some classical problems about finite monoids and computing the closure of a given regular language for this topology [17, 18]; it thus became an important question to characterize which regular languages are open or closed. A sequence of deep results [2, 3] finally led to the following combinatorial characterization for the regular open sets [19].

Lemma 2.2 *A regular language is open iff it is a finite union of languages of the form $L_0a_1L_1 \ldots a_kL_k$ where the a_i's are letters and the L_i's are group languages.*

3 Recognizing languages with loops

The aim of this section is to prove that any language of the form $B^*a_1B^* \cdots B^*a_kB^*$, where B is a finite alphabet and $a_i \in B$, can be recognized by a finite loop. This result will be of great help in proving Theorem 5.3.

In general, it is not an easy task to construct directly a loop B that recognizes a given language L. What can be done instead, is to construct a partially defined loop G that recognizes L, and then, embed G into a loop B. This motivates the following definition.

A groupoid G with an absorbing element, denoted 0, is *weakly cancellative* if for any $a, x, y \in G$, the two properties $(ax = ay \neq 0) \Rightarrow (x = y)$ and $(xa = ya \neq 0) \Rightarrow (x = y)$ are satisfied.

The Cayley table of a weakly cancellative groupoid is such that in each row and each column no nonzero element appears twice. Hence, the nonzero elements of such a groupoid form a partially defined groupoid which we call an *incomplete loop*. This terminology is justified by the following lemma.

Lemma 3.1 ([11]) *An incomplete loop containing n elements can be embedded in a loop containing t elements, for any $t \geq 2n$.* □

Recall that if Q is a loop and $w \in Q^+$ then $Q(w)$ is the set of elements that result from evaluating w using all possible bracketings.

Lemma 3.2 *Let Q be a loop and let $u, v, w \in Q^+$. Then, the cardinality of $Q(uwv)$ is at least as large as that of $Q(w)$.*

Weakly cancellative groupoids will be useful to prove that a language can be recognized by a loop. This is a consequence of the following lemma.

Lemma 3.3 *Any language recognized by a weakly cancellative groupoid, with 0 in the accepting set, is also recognized by a loop.*

Proof. Let G be a weakly cancellative groupoid, and let $L \subseteq G^*$ be a language recognized by G. Assume that 0 belongs to the accepting set. Let $B = G - \{0\}$, let $B^{(*)}$ be the free groupoid over the set B, and let β be the cardinality of B. We also denote by B the incomplete loop induced by the elements of B in G.

We will define a sequence of incomplete loops B_i, for $i \geq 0$. Let $B_0 = B$ and define B_{i+1} from B_i as follows. All products defined in B_i are defined identically in B_{i+1}. Moreover, for any undefined product $a \cdot b$ in B_i, we define $a \cdot b = (ab)$ in B_{i+1}

Remark. *Observe that for any $a, b \in B$, if the product ab is not defined in B_k, then $c = ab$ is a new element in B_{k+1}. Moreover, for any $d \in B_{k+1}$, the products cd and dc are not defined in B_{k+1}. Those products generate two new elements in B_{k+2}, and so on. This and Lemma 3.2 imply that for any $u, v \in B^*$ such that $k = |u| + |v|$, $B_{k+1}(uabv)$ contains at least k elements.*

Let $k = \beta + 2$ and let B_k be embedded in a finite loop H. We will argue that L is recognized by H with the accepting set containing all nonzero elements of the accepting set of G plus all elements not in B.

If $w \in B^*$ can be evaluated to a nonzero element in G, then w can be evaluated to the same element in H using the same parenthesization. This shows that if $w \in B^*$ cannot be evaluated to 0 in G, then w is accepted by G if and only if it is accepted by H.

Suppose that w can be evaluated to 0 in G. Then, there exists a segment u of w of minimal length that can be evaluated to 0, i.e. $w = sut$, $0 \in G(u)$ and for any strict segment v of u, $0 \notin G(v)$. So, there exist $u_1, u_2 \in B^+$ and $a, b \in B$ such that $u = u_1u_2$, $a \in G(u_1)$, $b \in G(u_2)$ and $ab = 0$ in G, but $a \neq 0$ and $b \neq 0$. This implies that w can be partially evaluated to $sabt$ both in G and in H. Now, there are two possibilities. First, if $|s| + |t| < k$, then $s(ab)t$ can only be evaluated, in H, to an element in $B^{(*)} - B$: in this case H accepts w. Otherwise, by the above remark, $H(w)$ contains at least $\beta + 1$ different elements, and so, at least one of them is not in B. Thus, H accepts w if and only if G accepts w. □

As an example of application of Lemma 3.3, we can show that any cofinite language is recognized by a finite loop. Since it is easy to see that no finite language can be recognized by a finite loop, the class of loop languages is not closed under complement.

Loops can also recognize languages that are not cofinite and are not recognized by a group. A simple example is the set OR $\subseteq \{0,1\}^*$, composed of all words that contain at least one 1. This language is recognized by U_1, the monoid defined by $00 = 0$ and $01 = 10 = 11 = 1$. Here, 0 is an identity and 1 is absorbing. Since U_1 is a weakly cancellative groupoid, the language OR can be recognized by a finite loop. It is easy to verify that the complement of OR cannot be recognized by any finite loop.

We close this section with a lemma that will play an important role in the proof of Theorem 5.3.

Lemma 3.4 *Let A be a finite alphabet and let $a_1, \ldots, a_k$ be elements of A ($k > 0$). Then $L_k = A^*a_1A^* \cdots A^*a_kA^*$ is recognized by a finite loop.*

Proof. The proof is by induction on k. Observe first that $L_1 = A^*a_1A^*$ can be recognized by the weakly cancellative groupoid U_1 discussed above. By Lemma 3.3, L_1 can also be recognized by a finite loop.

Let $k = i+j$, where $i, j > 0$. Then, there exists a finite loop Q_i that recognizes $L_i = A^*a_1A^* \cdots A^*a_iA^*$ with the accepting set $F_i \subseteq Q_i$ and there exists a finite loop Q_j that recognizes $L_j = A^*a_{i+1}A^* \cdots A^*a_kA^*$ with the accepting set $F_j \subseteq Q_j$.

Consider the weakly cancellative groupoid Q defined as the loop $Q_i \times Q_j$ except that $(a, b)(c, d) = 0$ whenever $a \in F_i$ and $d \in F_j$. Then, Q recognizes the language $L_k = A^*a_1A^* \cdots A^*a_kA^*$ with 0 as the accepting element. By Lemma 3.3, L is also recognized by a finite loop. □

4 Finite loops recognize only open regular languages

In [9], it is shown that finite loops only recognize regular languages. In this section we refine this result by showing that only open regular languages can be recognized by such algebras. The following can be observed.

Lemma 4.1 *Any language $L \subseteq A^*$ of the form $L_0 \cdots L_k$, where L_i is recognized by a finite group, is open.*

To prove the next theorem, we will use the following definition. Let A be an alphabet and S a set of variables. A special tree t over A with variables in S is a binary tree where each element of S appears exactly once as a label of a leaf. We will use special trees in two particular situations: when S contains a single variable X; and when each leaf of t is labeled with a variable in S. In this last case we say that $t = t(x_1, \ldots, x_n)$ is a special tree with n leaves.

Let t be a special tree over A with the variable X and let t' be any tree. We denote by $t \cdot t'$ the tree obtained when the leaf in t labeled with X is replaced by t'. Observe that when t' is also a special tree with variable X, the result is still a special tree with variable X.

Observe also that $\cdot$ is an associative operation. Hence, for any special trees $t_1, \ldots, t_k$ over Q with variable X, the expression $t_1 \cdot t_2 \cdot \cdots \cdot t_k$ defines the same special tree no matter which parenthesization is used. This will be denoted by $\prod_{i=1}^{k} t_i$.

Similarly, if $s(x_1, \ldots, x_n)$ is a special tree with n leaves and $t_1, \ldots t_n$ are arbitrary trees, then $s(t_1, \ldots, t_n)$ is the tree obtained by substituting the tree t_i for the leaf labeled with x_i, for all i.

Theorem 4.2 *Finite loops recognize only open regular languages.*

Proof. We will use the technique of [9].

Let Q be a finite loop. We define a *comb* over Q recursively as follows. Any $a \in Q \cup \{\epsilon\}$ is a comb. If $a \in Q$ and $u \in Q^{(*)}$ is a comb then $w = (au)$ is also a comb. No other element of $Q^{(*)}$ is a comb. Hence, a comb $c \in Q^{(*)}$ corresponds to the left-to-right bracketing of Yield (c).

Any $t \in Q^{(*)}$ can be decomposed into $t = s(t_1, \ldots, t_n)$, where $n \geq 1$, $s(x_1, \ldots, x_n)$ is a special tree with n leaves, and t_i is a comb over Q. Let $\mathrm{comb}(t)$ be the smallest n for which such a decomposition exists.

We will show that, for any tree $t \in Q^{(*)}$, there exists a yield-equivalent tree $s \in Q^{(*)}$ evaluating to the same element and such that $\mathrm{comb}(s)$ is bounded by a constant. By Lemma 4.1, this will prove the theorem because the set of words in Q^* that left-to-right evaluate to a given element forms a language recognized by the multiplication group of Q.

More precisely, we will show that for any tree $t \in Q^{(*)}$ such that $\mathrm{comb}(t) > 8^q$, where q is the order of Q, we can find a yield-equivalent tree $t' \in Q^{(*)}$ evaluating to the same element as t and such that $\mathrm{comb}(t') < \mathrm{comb}(t)$.

Suppose that $t \in Q^{(*)}$ is such that $\mathrm{comb}(t) = n > 8^q$, and let $t = s(t_1, \dots, t_n)$ be decomposed as explained above. Since s has more than 8^q leaves, it must possess a path of length $k > 3q$. Let the nodes on this path be $d_0, d_1, \dots, d_k$, where d_0 is the root of s and d_{i+1} is a child of d_i.

For $0 \le i \le q$, let s_i be the tree rooted at d_{3i}. Moreover, for $0 \le i < q$, let v_i be the special tree constructed from s_i by substituting the variable X for s_{i+1}. Hence, for each v_i there exist four indices $1 \le \alpha_i \le \beta_i < \gamma_i \le \delta_i \le n$ such that the leaves at the left of X in v_i are labeled with $x_{\alpha_i}, \dots, x_{\beta_i}$ and those on its right are labeled with $x_{\gamma_i}, \dots, x_{\delta_i}$. Moreover, the leaves of s_q are labeled with $x_{\alpha_q}, \dots, x_{\beta_q}$ for some $1 \le \alpha_q \le \beta_q \le n$. We have $s = (\prod_{i=0}^{q-1} v_i) \cdot s_q$, where $v_i = v_i P(x_{\alpha_i}, \dots, x_{\beta_i}, X, x_{\gamma_i}, \dots, x_{\delta_i})$, and $s_q = s_q(x_{\alpha_q}, \dots, x_{\beta_q})$.

We can thus write: $t = (\prod_{i=0}^{q-1} z_i) \cdot z_q$, where $z_i = v_i(t_{\alpha_i}, \dots, t_{\beta_i}, X, t_{\gamma_i}, \dots, t_{\delta_i})$ and $[z_q = s_q(t_{\alpha_q}, \dots, t_{\beta_q})$.

Let $w_i = \mathrm{Yield}\,(t_i)$ and define l_i to be the comb whose yield is $w_{\alpha_i} \cdots w_{\beta_i}$, and r_i the comb whose yield is $w_{\gamma_i} \cdots w_{\delta_i}$. Then $\bar{z}_i = ((l_i X) r_i)$ is yield equivalent to z_i. Using the fact that our loop is both cancellative and finite, it is easily verified (Lemma 7 of [9]) that there exist two integers a and b such that t and t' evaluate to the same element, where t' is defined as $t' = (\prod_{i=0}^{a-1} z_i) \cdot (\prod_{i=a}^{b} \bar{z}_i) \cdot (\prod_{i=b+1}^{q-1} z_i) \cdot z_q$. We observe that $\mathrm{comb}(z_i) \ge 3$ while $\mathrm{comb}(\bar{z}_i) \le 2$. This implies that $\mathrm{comb}(t') < \mathrm{comb}(t)$, proving the theorem.

$\square$

5 Every regular open language is recognized by a finite loop

In this section, we will conclude the proof of our main result by establishing the converse of Theorem 4.2. In order to do so, we will introduce the *block product* of a monoid with a groupoid. If the second operand is also a monoid, our construction reduces to the known notion of block product applied to associative structures [23, 25], which have proved itself to be extremely useful as a decomposition tool for finite monoids. Note that the block product is a two-sided version of the classical notion of wreath product: our construction below can be trivially modified to define the wreath product of a monoid with a groupoid. Actually, the wreath product is sufficient to prove our main result. We choose to give the more general construction as the potential for future applications seems more important.

Let α be a finite-index congruence on A^*, let B be the finite alphabet $A^*/\alpha \times A \times A^*/\alpha$ and let β be a finite-index congruence on the free groupoid $B^{(*)}$. For any u, v in A^*, we define the mapping $\theta_{u,v} : A^* \to B^*$ by $\theta_{u,v}(\epsilon) = \epsilon$ and $\theta_{u,v}(a_1 \dots a_n) = b_1 \dots b_n$, where $b_i = ([ua_1 \dots a_{i-1}]_\alpha, a_i, [a_{i+1} \dots a_n v]_\alpha)$.

We now define a binary relation on $A^{(*)}$;

$$(t,x)\,\beta\,\Box\,\alpha\,(s,y) \quad \text{iff} \quad \begin{array}{l} 1)\ x\,\alpha\,y \ \text{ and} \\ 2)\ (t,\theta_{u,v}(x))\,\beta\,(s,\theta_{u,v}(y)) \ \text{ for all } u,v\in A^* \end{array}$$

Lemma 5.1 *$\beta\,\Box\,\alpha$ is a congruence of finite index on $A^{(*)}$.*

Proof. That it is an equivalence of finite index is easily checked. Suppose now that $(t_1,x_1)\,\beta\,\Box\,\alpha\,(s_1,y_1)$ and $(t_2,x_2)\,\beta\,\Box\,\alpha\,(s_2,y_2)$; we want to show that $(t_1t_2,x_1x_2)\,\beta\,\Box\,\alpha\,(s_1s_2,y_1y_2)$.

Since α is a congruence, we have that $x_1x_2\,\alpha\,y_1y_2$. Fix now u and v arbitrarily in A^*; we have

$$\begin{array}{rcl} (t_1t_2,\theta_{u,v}(x_1x_2)) & = & (t_1,\theta_{u,x_2v}(x_1))(t_2,\theta_{ux_1,v}(x_2)) \\ & = & (t_1,\theta_{u,y_2v}(x_1))(t_2,\theta_{uy_1,v}(x_2)) \\ & \beta & (s_1,\theta_{u,y_2v}(y_1))(s_2,\theta_{uy_1,v}(y_2)) \\ & = & (s_1s_2,\theta_{u,v}(y_1y_2)). \end{array}$$

□

The next lemma says that the cancellation properties are preserved by the block product.

Lemma 5.2 *If A^*/α is a group and $B^{(*)}/\beta$ is a loop, then $A^{(*)}/\beta\,\Box\,\alpha$ is a loop.*

Proof. It is clear that the $\beta\,\Box\,\alpha$-class containing the identity of $A^{(*)}$ is an identity for the groupoid $A^{(*)}/\beta\,\Box\,\alpha$. We next show that $\beta\,\Box\,\alpha$ is a left-cancellative congruence, i.e. $(t,x)(s,y)\,\beta\,\Box\,\alpha\,(t,x)(q,z)$ implies $(s,y)\,\beta\,\Box\,\alpha\,(q,z)$.

The hypothesis says that $(ts,xy)\,\beta\,\Box\,\alpha\,(tq,xz)$; hence $xy\,\alpha\,xz$ and because α is a group congruence, hence left-cancellative, we deduce $y\,\alpha\,z$.

Consider some arbitrary u and v in A^*: we now need to show that
$(s,\theta_{u,v}(y))\beta(q,\theta_{u,v}(z))$. Choose $w\in A^*$ such that $wx\alpha u$ (such w exists since α is a group congruence). The hypothesis implies that $(ts,\theta_{w,v}(xy))\,\beta\,(tq,\theta_{w,v}(xz))$, hence $(t,\theta_{w,yv}(x))(s,\theta_{wx,v}(y))\,\beta\,(t,\theta_{w,zv}(x))(q,\theta_{wx,v}(z))$. Since y and z are α-congruent, we have $(t,\theta_{w,yv}(x))=(t,\theta_{w,zv}(x))$. Since β is a left cancellative congruence, we infer $(s,\theta_{wx,v}(y))\,\beta\,(q,\theta_{wx,v}(z))$, i.e. $(s,\theta_{u,v}(y))\,\beta\,(q,\theta_{u,v}(z))$. Hence $(s,y)\,\beta\,\Box\,\alpha\,(q,z)$.

By symmetry, we get that $\beta\,\Box\,\alpha$ is right cancellative as well, so that $A^{(*)}/\beta\,\Box\,\alpha$ is a loop. □

We are now ready to complete the proof of our main result.

Theorem 5.3 *If $L\subseteq A^*$ is a regular open language, then L can be recognized by a finite loop.*

Proof. Suppose that L is an open regular language; by Theorem 2.2, L is a finite union of languages of the form $L_0a_1L_1 \ldots a_kL_k$, where each L_i is recognized by a group. Using the classical construction for the associative case, it is readily verified that the class of loop languages is closed under union, so it suffices to prove that any language of the above form is recognized by a loop. If $k = 0$, the claim is clearly true.

Let now $k \geq 1$; without loss of generality we can assume that all L_i's are recognized by the same group G, e.g. by taking the direct product of the syntactic monoid of each language L_i. Let $G = A^*/\alpha$, so that each L_i is a union of α-classes; since concatenation distributes over union, and using once more closure under union, it suffices to consider the case where each L_i is a single class of the congruence, i.e. $L_i = [u_i]_\alpha$ for some u_i in A^*. Let $B = A^*/\alpha \times A \times A^*/\alpha$ and $H = B^{(*)}/\beta$ be the loop recognizing the language $B^*b_1B^* \ldots B^*b_kB^*$, as given by lemma 3.4, where $b_i = ([u_0a_1u_1 \ldots u_{i-1}]_\alpha, a_i, [u_ia_{i+1}u_{i+1} \ldots u_k]_\alpha)$. We claim that L is recognized by the loop $A^{(*)}/\beta \,\Box\, \alpha$; in fact we will show that $x \in L$ iff there is some tree t with $|x|$ leaves such that $[(t, \theta_{\epsilon,\epsilon}(x))]_\beta$ is an accepting element of H.

Let first x be in L; thus $x = x_0a_1x_1 \ldots a_kx_k$, with $x_i \;\alpha\; u_i$ for each i. Thus $\theta_{\epsilon,\epsilon}(x)$ is in $B^*b_1B^* \ldots b_kB^*$, hence for some tree t, $[(t, \theta_{\epsilon,\epsilon}(x))]_\beta$ is an accepting element of H.

Conversely, suppose $x \in A^*$ is such that, for some tree t, $[(t, \theta_{\epsilon,\epsilon}(x))]_\beta$ is an accepting element of H. Thus $(t, \theta_{\epsilon,\epsilon}(x)) = y_0b_1y_1 \ldots b_ky_k$, where $b_i = ([u_0a_1u_1 \ldots u_{i-1}]_\alpha, a_i, [u_ia_{i+1}u_{i+1} \ldots u_k]_\alpha)$. Therefore $x = x_0a_1x_1 \ldots a_kx_k$, where $x_0a_1x_1 \ldots x_{i-1} \;\alpha\; u_0a_1u_1 \ldots u_{k-1}$ for $i = 1, \ldots, k$, and also $x_k \;\alpha\; u_k$. Using the fact that α is a group congruence, we deduce $x_i \;\alpha\; u_i$ for each i, so that x is in L. □

6 Conclusion

Our characterization has a number of consequences, from the point of view of algebra, language theory and computational complexity.

First, we get a new combinatorial description of the regular open languages.

Corollary 6.1 *Any regular open language is a finite union of languages of the form $L_1 \ldots L_k$, where each L_i is a group language.*

Proof. By the proof of theorem 4.2, every loop language is of this form, hence by Theorem 5.3, this is also true for regular open languages. □

It is also appropriate to note the following structural representation that we get for loops. By the proof of theorem 4.2, we see that a loop G recognizes only regular open languages where the group languages that are needed are recognizable by the multiplication group of G. By the proof of theorem 5.3, any language of the form $L_0a_1L_1 \ldots a_kL_k$, where each L_i is recognized by the group

$\mathcal{M}(G)$, can be recognized by a loop of the form $A^{(*)}/\beta\Box\alpha$, where $A^*/\alpha \simeq \mathcal{M}(G)$ and β is the loop congruence induced by the construction of Lemma 3.4. Thus, in some sense, computing over the loop G is similar to computing over the group $\mathcal{M}(G)$, the non-associativity being taken care of by the very simple loops given in 3.4. It would be very interesting to see to what extent this phenomenon holds for groupoids in general.

Another consequence of this work is that the computational complexity of testing membership in a language recognized by a loop G can be infered from the algebraic structure of its multiplication group $\mathcal{M}(G)$. Any language L recognized by G is a finite union of languages of the form $L_1L_2\cdots L_k$ where the L_i's are recognized by $\mathcal{M}(G)$. When $\mathcal{M}(G)$ is solvable, each of these languages is in ACC^0 ([4]), where ACC^0 is the class of languages that are recognized by a family polynomial-size constant-depth Boolean circuits using NOT, AND, OR, and modular gates. In such a case, it is easy to see that L is also in ACC^0. This shows the following corollary.

Corollary 6.2 *Any language recognized by a loop whose multiplication group is solvable belongs to* ACC^0.

Note that when G is a group, it can be shown that $\mathcal{M}(G)$ is solvable precisely when G is solvable. Hence, the above result naturally fits in the structural complexity framework of [4].

It is remarkable that non-associative algebras such as loops could be related to such natural class of languages as the regular open languages. Our generalization of the block product yields a loop decomposition that shows that absence of associativity does not necessarily imply absence of structure. This is also confirmed by other recent works, such as [6]. We strongly believe that a better understanding of non-associative algebras, in particular finite groupoids, could have important consequences in language theory and computational complexity.

References

[1] A.A. Albert, *Quasigroups I*, *Trans. Amer. Math. Soc.*, **54** (1943) 507–519. *Quasigroups II*, *Trans. Amer. Math. Soc.*, **55** (1944) 401–419.

[2] C.J. Ash, *Inevitable sequences and a proof of the type II conjecture*, Proc. of the Monash Conf. on Semigroup Theory, World Scientific, Singapore (1991) 31–42.

[3] C.J. Ash, *Inevitable graphs: a proof of the type II conjecture and some related decision procedures*, Int. J. Alg. and Comp. **1** (1991) 127–146.

[4] D. Barrington and D. Thérien, "*Finite Monoids and the Fine Structure of NC^1* ", JACM **35**4(1988)941–952

[5] F. Bédard, F. Lemieux and P.McKenzie, *Extensions to Barrington's M-program model*, *TCS* **107** (1993), pp. 31-61.

[6] J. Berman, A. Drisko, F. Lemieux, C. Moore, and D. Thérien, *Circuits and Expressions with Non-Associative Gates*, Submitted to 12th Annual Conference on Computational Complexity (CCC'97)

[7] R.H. Bruck, *Contributions to the Theory of Loops*, Trans. AMS **60** (1946) 245–354.

[8] R.H. Bruck, *A Survey of Binary Systems*, Springer-Verlag, 1966.

[9] H. Caussinus and F. Lemieux, *The complexity of computing over quasigroups,* Proc. 14th annual FST&TCS, 1994, pp.36-47.

[10] S. Eilenberg, *Automata, Languages and Machines, vol. B*, Academic Press, New York, 1976.

[11] T. Evans, *Embedding Incomplete Latin Squares*, Amer. Math. Monthly, 67 pp.958-961, 1960.

[12] F. Gécseg and M. Steinby, *Tree Automata,* Akadémiai Kiadó, Budapest, 1984.

[13] M. Hall Jr., *A topology for free groups and related groups*, Ann. of Maths **52** (1950) 127–139.

[14] F. Lemieux, *Finite groupoids and their applications to computational complexity,* Ph.D. Thesis, McGill University, May 1996.

[15] J. Mezei and J.B. Wright, *Algebraic automata and context-free sets,* Inform. and Contr. **11** (1967) 3–29.

[16] J.-E. Pin, *Varieties of Formal Languages*, Plenum Press, New York, 1986.

[17] J.-E. Pin, *A topological approach to a conjecture of Rhodes*, Bulletin of the Australian Mathematical Society **38** (1988) 421–431.

[18] J.-E. Pin, *Topologies for the free monoid*, Journal of Algebra **137** (1991) 297–337.

[19] J.-E. Pin, *Polynomial closure of group languages and open sets of the Hall topology*, 21th ICALP, Springer-Verlag, LNCS 820, 1994, 424–435.

[20] J.-E. Pin, *$BG = PG$: A Success Story*, Proc. of Intern. Conf. on Groups, Semigroups, and Formal Languages, York 1993, Kluwer Publisher.

[21] C. Reutenauer, *Une topologie du monoïde libre*, Semigroup Forum **18** (1979), 33–49.

[22] C. Reutenauer, *Sur mon article "Une topologie du monoïde libre"*, Semigroup Forum **22** (1981), 93–95.

[23] J. Rhodes and B. Tilson, *The kernel of monoid morphisms*, J. Pure and Applied Algebra **62** (1989) 227–268.

[24] M.-P. Schützenberger *On finite monoids having only trivial subgroups*, Information and Control **8** (1965) 190–194.

[25] D. Thérien, *Two-sided wreath product of categories*, J. Pure and Applied Algebra **74** (1991) 307–315.

[26] L.G Valiant, *General context-free recognition in less than cubic time*, J. Comput. System Sci. **10** (2) (1975) 308–315.