Circuits and Expressions with Non-Associative Gates (Extended Abstract)*

Joshua Berman[1], Arthur Drisko[2], François Lemieux[3], Cristopher Moore[4], and Denis Thérien[3]

[1] State University of New York at Binghamton
[2] National Security Agency
[3] McGill University
[4] The Santa Fe Institute

Abstract. We consider circuits and expressions whose gates carry out multiplication in a non-associative groupoid such as loop. We define a class we call the *polyabelian* groupoids, formed by iterated quasidirect products of Abelian groups. We show that a loop can express arbitrary Boolean functions if and only if it is not polyabelian, in which case its Expression Evaluation and Circuit Value problems are $\mathbf{NC}^1$-complete and $\mathbf{P}$-complete respectively. This is not true for groupoids in general, and we give a counter-example. We show that Expression Evaluation is also $\mathbf{NC}^1$-complete if the groupoid has a non-solvable multiplication semigroup, but is in $\mathbf{TC}^0$ if the groupoid is both polyabelian and has a solvable multiplication semigroup. Thus, in the non-associative case, earlier results about the role of solvability in circuit complexity generalize in several different ways.

1 Introduction

Boolean expressions (or formulas) and circuits are well-known constructs in logic and computer science. A Boolean expression ϕ is either a variable x_i, the constant TRUE or FALSE, or is formed from shorter expressions by using the constructs $(\phi_1 \wedge \phi_2)$, $(\phi_1 \vee \phi_2)$ or $\neg\phi_1$. If all the x_i are assigned truth values TRUE or FALSE, then ϕ's truth value is determined by interpreting $\wedge$, $\vee$ and $\neg$ as AND, OR and NOT respectively.

A Boolean circuit is an acyclic directed graph with source nodes (inputs) labeled x_i, TRUE or FALSE, a single sink node (output), and three kinds of intermediate nodes: AND and OR gates with two inputs, and NOT gates with one input. (Expressions are simply circuits whose graph is a tree.) Then the truth value of the output node is defined in the obvious way.

Given an expression or circuit, and the truth values of its variables or inputs, determining the truth value of its output is called the Expression Evaluation or Circuit Value problem respectively. These problems are deeply related to two important complexity classes, $\mathbf{NC}^1$ and $\mathbf{P}$. Indeed Expression Evaluation is $\mathbf{NC}^1$-complete [10], while Circuit Value is $\mathbf{P}$-complete [16].

In this paper we will study the natural algebraic counterparts of these two questions; that is, we will consider circuits and expressions over some finite algebraic system $(A, \cdot)$, where $\cdot$ is a (not necessarily associative) binary operation on A, rather than over the Boolean semiring. Thus an expression ϕ over A is either a variable x_i, a constant in A, or is formed from shorter expressions using the construct $(\phi_1 \cdot \phi_2)$. [5] Similarly, in a circuit over A, input nodes are labeled by variables or by constants in A, and all other nodes have binary fan-in and compute the operation in A.

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

[5] Note that our algebraic expressions are fully parenthesized: the similar question when parenthesis are not given is known as the WordProblem; there is of course no difference between the two cases when the operation is associative. In the non-associative setting, the WordProblem has been taken to mean the following: given an unbracketed sequence of elements of A, does there exist a parenthesization that evaluates to some target element; this question has been extensively studied [17, 7, 5] and will not be pursued here.

Our goal is to investigate how the algebraic properties of $(A, \cdot)$, such as associativity, commutativity, solvability and so on, affect the computational complexity of the problems Expression Evaluation and Circuit Value. Our work is the direct continuation of a series of results dealing with the associative case, including for example [2, 4, 6, 20]: the following table summarizes what is known in that setting.

	Expression Evaluation	Circuit Value
non-solvable	$\mathbf{NC}^1$-complete	$\mathbf{P}$-complete
solvable	$\mathbf{ACC}^0$	$\mathbf{ACC}^1 \cap \mathbf{DET}$

We assume that the reader is familiar with standard complexity classes; in case of emergency, one can consult [15]. The above table shows that, for semigroups, solvability has a dramatic impact on the difficulty of solving the two problems considered (under the customary complexity-theoretic assumptions that $\mathbf{ACC}^0 \subsetneq \mathbf{NC}^1$ and $\mathbf{ACC}^1 \cap \mathbf{DET} \subsetneq \mathbf{P}$).

Our main contribution is to extend these results to non-associative operations. In particular, in the case of loops, we are able to characterize exactly when Expression Evaluation is $\mathbf{NC}^1$-complete and when Circuit Value is $\mathbf{P}$-complete. Partial results for groupoids in general are also obtained. It turns out that the idea of solvability, which is the dividing line in the associative world, must be refined in two important ways in the non-associative case: *polyabelianness*, the property of being an iterated quasidirect product of Abelian groups; and $\mathcal{M}$*-solvability*, the property of having a solvable multiplication semigroup.

2 Algebraic preliminaries

We will use the following terms. For the theory of quasigroups and loops, we recommend [22] as an introduction.

A *groupoid* (also called an *algebra* or *magma*) $(G, \cdot)$ is a set G with a binary operation $G \times G \to G$ called the product. The product of two elements a and b in G is written $a \cdot b$. When there is no ambiguity, $(G, \cdot)$ is simply denoted by G and the product is denoted using concatenation. The *order* of a groupoid is the number of elements in G, written $|G|$. Throughout the paper, we will assume that our groupoids are finite.

A (finite) *quasigroup* is a groupoid whose multiplication table is a *Latin square*, in which each symbol occurs once in each row and each column. A *loop* is a quasigroup with an identity. In a loop, each element a has a *left (right) inverse* which is the unique element a^λ (resp. a^ρ) such that $a^\lambda \cdot a = 1$ (resp. $a \cdot a^\rho = 1$). If they are the same, we will refer to them both as the two-sided inverse a^{-1}.

A *semigroup* is an associative groupoid, and a *monoid* is a semigroup with an identity. A *group* is an associative quasigroup; groups have two-sided inverses and an identity. We refer to the *cyclic group* $Z_p = \{0, 1, \ldots, p-1\}$ with addition mod p, and the groups S_n and A_n of permutations and even permutations respectively on n elements.

A *subgroupoid* (*subquasigroup, subloop*, etc.) of 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 $\langle S \rangle$.

The *left (right) cosets* of a subloop $H \subseteq G$ are the sets $aH = \{ah \mid h \in H\}$ $(Ha = \{ha \mid h \in H\})$ 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 set of cosets of H is the *quotient loop* or *factor* G/H; it has identity $1H = H$, and is the image of G under the homomorphism $\phi(a) = aH$.

A subloop of G is *proper* if it is neither $\{1\}$ nor all of G. A *simple* loop is one with no proper normal subloops. A *minimal* normal subloop of G is one which does not properly contain any proper normal subloops of G. The intersection of two normal subloops is normal.

The *commutator* of two elements a, b of a loop G is the unique element $[a, b] \in G$ which satisfies the equation $ab = (ba)[a, b]$. and the *associator* of three elements $a, b, c \in G$ is the unique element

$[a, b, c]$ which satisfies the equation $a(bc) = ((ab)c)[a, b, c]$. The subloop generated by all possible commutators and associators in a loop G is called the *commutator-associator subloop* or *derived subloop* and is denoted G'. It is normal, and the quotient G/G' is an Abelian group.

A loop G is *solvable* if either it is an abelian group or G/G' is solvable. Any non-solvable loop contains a simple loop H for which $H' = H$. More generally, we call a groupoid solvable if it contains no non-solvable loops.

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.

A loop G is *nilpotent* if either it is an abelian group or its center is non-trivial and $G/Z(G)$ is nilpotent. Nilpotent loops form a proper subclass of the solvable ones.

In a quasigroup Q, we can define left and right multiplication as functions $L_a(b) = a \cdot b$ and $R_a(b) = b \cdot a$. These are permutations on Q (the rows and the columns of the multiplication table), and the *multiplication group* $\mathcal{M}(Q)$ is the group of permutations they generate. More generally, any groupoid has a *multiplication semigroup* generated by the L_a's and R_a's, which are not necessarily one-to-one functions on Q.

Finally, an *homomorphism* from $(G, \cdot)$ to $(H, \star)$ is a function f from G to H satisfying $f(a \cdot b) = f(a) \star f(b)$. If G and H are the same groupoid, f is called an *endomorphism*, or an *automorphism* if it is also a bijection.

3 The polynomial closure of a groupoid

Let us define the set of functions $A^n \to A$ that can be expressed as circuits whose gates carry out multiplication in a groupoid A, and whose inputs can be variables or constant elements of A. This is equivalent to the set of (non-associative) polynomial expressions definable in A with constants and variables, such as $\phi(x_1, x_2, x_3) = (x_1(ax_2))((bx_3)(bx_3))$:

Definition. The *polynomial closure* of a groupoid A is the smallest set $\mathcal{P}(A)$ of functions ϕ on an arbitrary number of variables $x_1, \ldots, x_k$ containing the constants a (for all $a \in A$), the variables x_i (for all $1 \leq i \leq k$), and the products $(\phi_1 \cdot \phi_2)$ (for all $\phi_1, \phi_2 \in \mathcal{P}(A)$).

We will refer to the set of functions on k variables as $\mathcal{P}^k(A)$: for instance, $\mathcal{P}^1(A)$ contains the multiplication semigroup $\mathcal{M}(A)$, as well as functions like $\phi(x) = x^2$.

In [17] it is proved that if G is a simple non-abelian loop then $\mathcal{P}^n(G)$ contains all functions $f : G^n \to G$. This property is called *functional completeness.* Not every groupoid is functionally complete, however. For example, Straubing has shown [23] that if G is a solvable group and if FALSE and TRUE denote two distinct elements of G, then $\mathcal{P}(G)$ does not contain $x \wedge y$.

Let G be a loop. In [8], the *inner mapping group* $\mathcal{J} \subseteq \mathcal{M}(G)$ is defined as the set of permutations in $\mathcal{M}(G)$ that fix the identity. We generalize this definition as follows. Let U be the set of functions ϕ in $\mathcal{P}^1(G)$ such that $\phi(1) = 1$. Hence, for an element $x \in G$, $U(x) = \{\phi(x) \mid \phi \in U\}$ is the set of y's that we can send x to, while fixing 1. This set has the following remarkable property.

Lemma 1. *Let G be a loop, $x \in G$, and let N be a subloop of G. Then, N is normal if and only if $U(N) = N$, and $U(x)$ is the smallest normal subloop of G that contains x.*

Proof. We first observe that, if N is normal, then for any $\phi \in U$ and any $n \in N$, there exists $n' \in N$ such that $\phi(n) = n'\phi(1) = n'$. This implies that $U(N) \subseteq N$. It is proved in [8] that N is normal if and only if $\mathcal{J}(N) = N$. Clearly, $\mathcal{J}$ is included in U, and $\mathcal{J}(N) \subseteq U(N) \subseteq N$. This shows the first part of the lemma.

In [9] it is shown that $K = \langle \mathcal{J}(x) \rangle$ is the smallest normal subloop that contains x. We observe that $K \subseteq U(x)$ and, K being normal, $U(K) = K$. Finally, since, $x \in K$, $U(x) \subseteq U(K) = K$. □

A property of U that we will exploit is described in the following lemma.

Lemma 2. *If G is a loop, for any $x, y \neq 1$ such that $y \in U(x)$, there is a function $\pi_{x \to y}$ in $\mathcal{P}^1(G)$ that sends x to y and keeps the identity fixed.*

Proof. Immediate from the definition of $U(x)$. $\square$

As a corollary, we have:

Corollary 3. *If G is a simple loop, then for any $x, y \neq 1$, $\pi_{x \to y}$ is in $\mathcal{P}^1(G)$*

Proof. If G is simple, then for any $x \neq 1 \in G$, $U(x) = G$. $\square$

4 Boolean-complete groupoids

We are interested in whether a groupoid can express arbitrary Boolean functions.

Definition. Let G be a groupoid, and let $T \subset G$ and $F \subset G$ be two disjoint subsets of G. Define the mapping $\beta : T \cup F \to \{\text{FALSE}, \text{TRUE}\}$ with $\beta(g) = \text{TRUE}$ if $g \in T$ and $\beta(g) = \text{FALSE}$ is $g \in F$. We say that a Boolean function $f(x_1, \ldots, x_n)$ is *represented* [6] by an expression $\phi \in \mathcal{P}^n(G)$, if for any $g_1, \ldots, g_n \in T \cup F$, we have $f(\beta(g_1), \ldots, \beta(g_n)) = \beta(\phi(g_1, \ldots, g_n))$.

Definition. A groupoid G is *Boolean-complete* if there exist two disjoint subsets $T \in G$ and $F \in G$ such that any Boolean function can be represented by some $\phi \in \mathcal{P}(G)$ using elements in T to express TRUE and elements in F to express FALSE.

Note that, in order to have Boolean-completeness, it suffices to be able to represent AND and NOT. As a next observation we have the following lemma.

Lemma 4. *If a groupoid is Boolean-complete, then its* EXPRESSION EVALUATION *and* CIRCUIT VALUE *problems are* $\mathbf{NC}^1$*-complete and* $\mathbf{P}$*-complete respectively.*

Proof. Boolean EXPRESSION EVALUATION and CIRCUIT VALUE are reducible to their algebraic counterparts, since a local transformation can replace AND, OR and NOT gates with complexes of algebraic gates. $\square$

In [11] (see also [17]) it is shown that any non-solvable groupoid is Boolean-complete. Since the circuit value problem for solvable semigroups is in **NC** [6, 20], non-solvability is both necessary and sufficient for Boolean-completeness in the case of semigroups (assuming $\mathbf{NC} \subsetneq \mathbf{P}$). However, a loop can be solvable and still be Boolean-complete.

Let $(G, \cdot)$ be

$\cdot$	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2	3	4	1	6	7	8	5
3	3	4	1	2	7	8	5	6
4	4	1	2	3	8	5	6	7
5	5	6	7	8	1	3	2	4
6	6	7	8	5	3	2	4	1
7	7	8	5	6	2	4	1	3
8	8	5	6	7	4	1	3	2

Here G' is the normal subloop $\{1,2,3,4\} \cong Z_4$. The lower right-hand block is the multiplication table of a Boolean-complete quasigroup $Q = (\{1,2,3,4\}, \star)$. To see this, let FALSE $= 1$ and TRUE $= 2$ and write $a \wedge b = (a \star b)^2$ and $\neg a = 3 \star (1 \star a)$. Since the product in Q can be expressed in $\mathcal{P}(G)$ as $a \star b = (5 \cdot a) \cdot (5 \cdot b)$, it follows that G is Boolean-complete. We also note that this loop has a solvable multiplication group.

As can be seen in this example, solvability does not constrain the lower right-hand block in any way. In the next section we will show that for loops a property slightly more subtle than solvability is the dividing line between Boolean-completeness and -incompleteness.

[6] Other types of representations are discussed in the literature (e.g. in [23]): in the strong representation, T and F are restricted to be singletons, and then it is not clear that functions representable on a homomorphic image of a groupoid G are also representable on G; in the weak representation, composition of expressions is not possible. Our notion avoids these two pitfalls.

5 Polyabelian groupoids

The *direct product* of two groupoids $A \times B$ is the set of pairs (a, b) with pairwise multiplication, $(a_1, b_1) \cdot (a_2, b_2) = (a_1 a_2, b_1 b_2)$. Consider the following generalization:

Definition. A *quasidirect product* (called loop extension in [1]) of two groupoids A and B is the set of pairs (a, b), under the operation

$$(a_1, b_1) \cdot (a_2, b_2) = (a_1 a_2,\ b_1 \odot_{a_1,a_2} b_2)$$

where each a_1, a_2 defines a *local operation* $\odot_{a_1,a_2}$ on B. We will denote such a product $A \otimes B$.

If the local operations are of the form

$$b_1 \odot_{a_1,a_2} b_2 = f_{a_1,a_2}(b_1) \cdot g_{a_1,a_2}(b_2)$$

we will call them *separable*. Furthermore, if B is an Abelian group and the $\odot$'s are of the form

$$b_1 \odot_{a_1,a_2} b_2 = f_{a_1,a_2}(b_1) + g_{a_1,a_2}(b_2) + h_{a_1,a_2}$$

where f and g are endomorphisms on B and h is an element of B, depending arbitrarily on a_1 and a_2, then we will call them *affine*. We will call a quasidirect product $A \otimes B$ separable or affine on B if all its local operations are.

Lemma 5. *If $A \otimes B$ is a quasigroup, then A and B are quasigroups and all the $\odot$'s are quasigroup operations on B. If $A \otimes B$ is a quasigroup and is affine on B, then all the f's and g's are automorphisms on B. If $A \otimes B$ is a loop affine on $(B, +)$, then A is a loop, and for all $a \in A$ we have $f_{a,0} = g_{0,a} = \mathbf{1}$ and $h_{a,0} = h_{0,a} = 0$; thus $b_1 \odot_{0,0} b_2 = b_1 + b_2$ for $b_1, b_2 \in B$, and B is a subloop of A on which $+$ and $\cdot$ coincide.*

Proof. Easy; left to the reader. □

The quasidirect product is a rather general way of extending to a loop from a normal subloop:

Lemma 6. *If a loop G has a normal subloop N, then G is isomorphic to a quasidirect product $(G/N) \otimes N$. Furthermore, all the local operations, and (if N is Abelian and G is affine on N) the f's, g's and h's, are expressible in $\mathcal{P}(G)$.*

Proof. Choose a set T with one element in each coset of N, and write $\tau(a)$ for the element of T in aN. Then $T \cong G/N$ if we define multiplication on T as $t_1 \bullet t_2 = \tau(t_1 \cdot t_2)$.

Every element of G can be uniquely written $g = tn$ where $t \in T$ and $n \in N$. Then

$$(t_1 n_1) \cdot (t_2 n_2) = (t_1 \bullet t_2) \cdot (n_1 \odot_{t_1,t_2} n_2)$$

where

$$n_1 \odot_{t_1,t_2} n_2 = L^{-1}_{(t_1 \bullet t_2)}((t_1 n_1) \cdot (t_2 n_2))$$

If N is an Abelian group and G is affine on N, then

$$\begin{aligned} f_{a_1,a_2}(n) &= (n \odot_{a_1,a_2} 0) - (0 \odot_{a_1,a_2} 0) \\ g_{a_1,a_2}(n) &= (0 \odot_{a_1,a_2} n) - (0 \odot_{a_1,a_2} 0) \\ h_{a_1,a_2} &= (0 \odot_{a_1,a_2} 0) \end{aligned}$$

where we abuse notation by writing $+$ and 0, instead of $\cdot$ and 1, for products in N. Finally, we note that the $+$ operation on N is in $\mathcal{P}(G)$, since $a \odot_{0,0} b = a + b$ by lemma 5. □

Next define the following class of groupoids:

Definition. A groupoid is *polyabelian* if it is an iterated quasidirect product of Abelian groups A_i:

$$((A_0 \otimes A_1) \otimes A_2) \otimes \cdots \otimes A_k$$

where all the products are affine.

It can be shown that the class of polyabelian groupoids is a pseudovariety, i.e. that it is closed under quotients, taking subgroupoids and forming finite direct products. The next few lemmas show inclusions between the polyabelian loops and some common classes of groups and loops:

Lemma 7. *Polyabelian loops are solvable.*

Proof. Let $H_i = (A_i \otimes A_{i+1}) \otimes \cdots \otimes A_k$. Then the reader can show that all the H_i are normal subloops of G, and $H_i/H_{i+1} = A_i$ is Abelian. In particular, $G_i \subseteq H_i$ in the derived series where $G_{i+1} = G_i'$. □

The converse is not true for loops in general (for instance, the solvable Boolean-complete loop described in section 4, since the local operations in its lower right-hand block are not affine) but it is true for groups:

Lemma 8. *Solvable groups are polyabelian.*

Proof. Any solvable group G has a normal subgroup N which is Abelian, namely the last non-trivial group in its derived series such that $N' = \{1\}$. Then the local operation in $G/N \otimes N$ (from lemma 6) can be written

$$\begin{aligned} n_1 \odot_{t_1,t_2} n_2 &= (t_1 \bullet t_2)^{-1} t_1 n_1 t_2 n_2 \\ &= ((t_1 \bullet t_2)^{-1} t_1 t_2) + (t_2^{-1} n_1 t_2) + n_2 \end{aligned}$$

where we use $+$ for products within N. Thus G is affine on N where

$$\begin{aligned} f_{t_1,t_2}(n) &= t_2^{-1} n t_2 \\ g_{t_1,t_2}(n) &= n \\ h_{t_1,t_2} &= (t_1 \bullet t_2)^{-1} t_1 t_2 \end{aligned}$$

Then G/N has an Abelian normal subgroup, and so on; by induction G is polyabelian. □

Lemma 9. *Nilpotent loops are polyabelian.*

Proof. Let G be a nilpotent loop with center $Z(G)$. Then the local operation in $G/Z(G) \otimes Z(G)$ is

$$n_1 \odot_{t_1,t_2} n_2 = L^{-1}_{(t_1 \bullet t_2)}(t_1 t_2) + n_1 + n_2$$

since n_1 and n_2 associate and commute with everything. So G is affine on $Z(G)$ with f and g being the identity mapping and $h = L^{-1}_{(t_1 \bullet t_2)}(t_1 t_2)$. $G/Z(G)$ also has a non-trivial center, and so on; by induction G is polyabelian. □

Thus polyabelianness coincides with solvability for groups, and lies properly between nilpotence and solvability for loops.

We wish to show that, for purposes of Boolean-completeness, polyabelianness is the correct generalization of solvability in the non-associative case: that is, a groupoid is Boolean-complete if and only if it is not polyabelian. This will turn out to be true for loops, but not for groupoids in general. In the appendix we give an example of a non-polyabelian groupoid which is not Boolean-complete.

6 Polyabelian groupoids aren't Boolean-complete

In one direction, we can prove this for all groupoids. In [20] it was shown that CIRCUIT VALUE for polyabelian groupoids is in $\mathbf{ACC}^1$, and a simple modification of the proof for solvable semigroups in [6] shows that it is also in **DET**. We now show directly that polyabelian groupoids cannot express the AND function.

Theorem 10. *A polyabelian groupoid cannot be Boolean-complete.*

Proof. Let $G = ((A_0 \otimes A_1) \otimes A_2) \otimes \cdots A_k$ be a polyabelian groupoid. It suffices to show that G cannot represent the AND of n variables when n is large enough.

Assume all abelian groups in the decomposition of G have exponent dividing q, i.e. they all satisfy the equation $qx = 0$ for any element x, and suppose $w(x_1, \cdots, x_n)$ is an expression over G representing AND, with T and F being the true and false elements respectively.

Consider the projection of w into A_0; for an Abelian group, any function in $\mathcal{P}(A)$ has the form $g + c_1x_1 + \cdots + c_nx_n$, where g is in A_0 and the c_i's are non-negative integers. If $n \geq q^2$ there must be q variables having the same coefficient mod q; without loss of generality, let them be $x_1, \cdots, x_q$. Take any t in T, f in F: then $w(t^n) = t_1 \in T$ and $w(f^q t^{n-q}) = f_1 \in F$ have the same A_0-component. Note that it is also true that $w(t_1^n)$ and $w(f_1^q t_1^{n-q})$ have the same A_0-component.

For the inductive step, consider $G = K \otimes A$ where A is an abelian group whose exponent divides q, and assume that there is some t in T and f in F which agree on their K-component. Note that this certainly implies that $w(t^n)$ and $w(f^q t^{n-q})$ also agree on their K-component. If we restrict the domain of the expression w to $\{t, f\}^n$, we can replace any occurrence of x_i by (k, y_i), where k is the common K-component of t and f, i.e. only the A-component remains a variable.

Consider now the parse tree of w. Label each node with the element $(k, a) \in G$ resulting from the evaluation of the corresponding subexpression. Also, if N is an inner node whose left (right) child N_l (N_r) is labeled with (k_l, a_l) ((k_r, a_r)), then we label the edge from N to N_l (from N to N_r) with the endomorphism f_{k_l,k_r} (g_{k_l,k_r}).

The projection of w into the A-component will be a sum (in A) of the following terms: for each path π from the root to a leaf, there will be a term $t_\pi(a)$, where t_π is an endomorphism of A and a in $A \cup \{y_1, \ldots, y_n\}$ is the A-component of the symbol labeling that leaf; also, for each path π from the root to an inner node, there will be a term $t_\pi(h_{k_1,k_2})$, where t_π is an endomorphism of A and h_{k_1,k_2} is a constant in A that depends on the K-evaluation k_1 of the left subtree of that inner node and on the K-evaluation k_2 of its right subtree. In each case, the endomorphism t_π is obtained by composing the labels of the edges along the path.

Let $\theta_1, \ldots, \theta_t$ be the endomorphisms of A. Using the commutativity of A, we can group similar terms and the above sum becomes $g + (c_{11}\theta_1 + \cdots + c_{1t}\theta_t)(y_1) + \cdots + (c_{n1}\theta_1 + \cdots + c_{nt}\theta_t)(y_n)$, where g is in A and the c_{ij}'s are non-negative integers. Note that the c_{ij}'s will depend on which common value is being used to replace the K-components of the variables.

If $n \geq q^{t+1}$, we can then find q variables having the same coefficients; let's assume these are $y_1, \ldots, y_q$. Then we get that $w(t^n) = t_1$ and $w(f^q t^{n-q}) = f_1$ agree in their A-component; since they also agree on their K-component, they are equal in $K \otimes A$. Continuing in this way, we obtain elements $t, f \in G$ that agree on their A_i-component for all i, and so T and F are not disjoint, a contradiction. □

7 Non-polyabelian loops are Boolean-complete

First we need the following lemma:

Lemma 11. *If G/N is Boolean-complete, then G is.*

Proof. Let the required subsets of G/N be $T = \{Nt_1, \ldots, Nt_r\}$ and $F = \{Nf_1, \ldots, Nf_s\}$; then clearly the subsets $T' = \{nt_i; n \in N, 1 \leq i \leq r\}$ and $F' = \{nf_i; n \in N, 1 \leq i \leq s\}$ of G satisfy the requirements for Boolean-completeness. □

We can now show:

Theorem 12. *Non-polyabelian loops are Boolean-complete.*

Proof. For non-solvable loops, the result has been proved in [11]. Assume now that G is solvable, and let H be the smallest non-polyabelian factor of G. Then H has a normal subloop K which is an Abelian group, namely the last non-trivial subloop in its derived series with $K' = \{1\}$. Let N be a minimal normal subloop of H contained in K; then N is also Abelian. H is not affine on N, otherwise H/N would be a smaller non-polyabelian factor.

Recall the definition of $U(x)$ from section 3. Since N is minimal, $U(n) \supset N$ for any $n \in N$; otherwise $U(n) \cap N$ would be a smaller normal subloop. So, by lemma 2, for any $n_1, n_2 \in N$, there exists a function $\pi_{n_1 \to n_2}$ that sends n_1 to n_2 and preserves the identity.

Since H is not affine on N, some local operation $\odot$ is either not separable or not affine. Define the *separator*

$$K_\odot(n_1, n_2) = (n_1 \odot n_2) - (n_1 \odot 0) - (0 \odot n_2) + (0 \odot 0)$$

where we use $+$ and $-$ for products in N. If $\odot$ is not separable, then $K_\odot(n_1, n_2) = k \neq 0$ for some n_1, n_2; however, $K_\odot(0, n) = K_\odot(n, 0) = 0$ for any n. But this gives us our AND gate: let FALSE $= 0$ as before and choose TRUE $= t \in N$, and let

$$a \wedge b = \pi_{k \to t}\big(K_\odot(\pi_{t \to n_1}(a), \pi_{t \to n_2}(b))\big)$$

If all the local operations are separable, then one must not be affine: that is, some f or g is not a endomorphism of N. Let $f(n) = (n \odot 0) - (0 \odot 0)$ as in lemma 7, and define the *affinator*

$$L_f(n_1, n_2) = f(n_1 + n_2) - f(n_1) - f(n_2)$$

If f is not a endomorphism, then $L_f(n_1, n_2) = k \neq 0$ for some n_1, n_2; but $L_f(n, 0) = L_f(0, n) = 0$ for all n, so

$$a \wedge b = \pi_{k \to t}\big(L_f(\pi_{t \to n_1}(a), \pi_{t \to n_2}(b))\big)$$

Similarly if some g is not a endomorphism.

So any nonlinearity in the local operations can be used to construct an AND gate, and we can express negation as $\neg a = ta^\rho$. Thus H is Boolean-complete with the singletons $T = \{t\}$ and $F = \{0\}$, and G is Boolean-complete by lemma 11. □

Combining theorems 10 and 12, we get:

Theorem 13. *A loop is Boolean-complete iff it is not polyabelian.*

8 $\mathcal{M}$-solvability and expression evaluation

Since non-polyabelian groupoids can represent arbitrary Boolean functions, their EXPRESSION EVALUATION problem is $\mathbf{NC}^1$-complete. This condition is not necessary however, even for loops. In order to get an exact characterization, we now consider the relationship between expressions in a groupoid and words in the multiplication semigroup of that groupoid.

Definition. A groupoid is *$\mathcal{M}$-solvable* if its multiplication semigroup is solvable.

For groups, solvability and $\mathcal{M}$-solvability coincide since $\mathcal{M}(G) \cong (G \times G)/Z(G)$ is solvable iff G is. For loops, $\mathcal{M}$-solvability implies solvability by a recent result of Vesanen [24], and nilpotent loops are $\mathcal{M}$-solvable by a theorem of Bruck [8]. Thus, like polyabelianness, $\mathcal{M}$-solvability is a generalization of solvability which lies properly between nilpotence and solvability for loops.

What is the exact relationship between $\mathcal{M}$-solvability and polyabelianness? Indeed, there exist loops that are both $\mathcal{M}$-solvable and polyabelian, such as all Abelian groups. A loop can also be $\mathcal{M}$-solvable without being polyabelian: we have seen in section 4 an example of an $\mathcal{M}$-solvable loop that is Boolean-complete. In the appendix we show how to construct a polyabelian loop that is not $\mathcal{M}$-solvable. As a consequence, we have the following theorem.

Theorem 14. *$\mathcal{M}$-solvable loops and polyabelian loops form incomparable classes.*

One consequence of this theorem is that a loop can be non-Boolean-complete while its multiplication group is non-solvable. This is quite surprising in view of the following observation.

Lemma 15. *For any groupoid G, the* Expression Evaluation *problem of $\mathcal{M}(G)$ is reducible that of G.*

Proof. Suppose we have a word in $\mathcal{M}(G)$ such as $L_a R_b L_c$. If G has n elements, the product is determined by its action on each element $1, \ldots, n$. But this just means evaluating n expressions in G, namely $a((c1)b), a((c2)b), \ldots, a((cn)b)$. □

Then we immediately have

Theorem 16. *The* Expression Evaluation *problem for non-$\mathcal{M}$-solvable groupoids is $\mathbf{NC}^1$-complete.*

Proof. By lemma 15 and the fact that Expression Evaluation for non-solvable semigroups is $\mathbf{NC}^1$-complete [2]. □

As mentioned above, Expression Evaluation is in $\mathbf{ACC}^0$ for solvable groups [4]. For this problem, the correct notion to replace solvability in the non-associative case consists of being *both* polyabelian and $\mathcal{M}$-solvable, but the need to parse the expression makes the problem somewhat harder. Before showing this, we state a lemma that is proved in the appendix.

Lemma 17. *For an affine quasidirect product $G = A \otimes B$, define $\langle fg(B) \rangle$ as the subgroup of $Aut(B)$ generated by all the f's and g's in the local operations. Then G is $\mathcal{M}$-solvable if and only if $\langle fg(B) \rangle$ is solvable.*

Theorem 18. *For a groupoid A which is both polyabelian and $\mathcal{M}$-solvable,* Expression Evaluation *is in $\mathbf{TC}^0$, or in $\mathbf{ACC}^0$ if the expression's parse tree is already known.*

Proof. Consider a polyabelian groupoid $G = (A_0 \otimes A_1) \otimes \cdots \otimes A_k$. An expression ϕ can be inductively calculated using the ideas in the proof of theorem 10 (see also the algorithm in [20] for Circuit Value): more precisely, for each node in the parse tree of ϕ, we compute in parallel the value of the corresponding subexpression, first in A_0, then in $A_0 \otimes A_1$, etc...

The A_0-computations are over an Abelian group, hence certainly realizable in $\mathbf{ACC}^0$. At the $(i+1)^{st}$ iteration, we get the A_i-component of each subexpression by summing (in A_i) terms of the form $t_\pi(a)$ and $t_\pi(h_{k_1,k_2})$. Each value a is available from the original input, each value h_{k_1,k_2} is available from the previous iteration of the algorithm, and each endomorphism t_π is a composition of polynomially-many endomorphisms of A_i, each of which is also obtained from the previous iteration; this composition takes place in $\langle fg(A_i) \rangle$; by lemma 17 this is solvable if G is $\mathcal{M}$-solvable, so each composition can be evaluated in $\mathbf{ACC}^0$, as can the sum. We can calculate all of ϕ's components with k induction steps, and we're done.

However, this computation needs access to the parse tree of the input expression. But the set of well-formed expressions is a *structured context-free language* [14], and such expressions can be parsed in $\mathbf{TC}^0$ as shown in [3]. Thus we can compute ϕ by first parsing it with a $\mathbf{TC}^0$ circuit, and then calculating the value with $k+1$ levels of $\mathbf{ACC}^0$ circuits.

(Since the expression only has to be parsed once, the *majority-depth* of the circuit, defined as the maximum number of threshold gates that any path traverses, is constant for all groupoids. Thus this problem is in $\widehat{\mathbf{TC}}^0_d$ for some small d as defined in [18].) □

Cases where the parse tree is already known could include uniform families of expressions, one of each length. For instance, the problem of predicting a cellular automaton amounts to evaluating a uniform family of circuits, one of each size. This uniformity can significantly simplify the Circuit Value problem, as in [20, 21].

9 Conclusion and directions for further work

We have shown that the very precise relationship between solvability and circuit complexity that holds for associative algebras, has an elegant counterpart in the non-associative case, provided that we refine appropriately the notion of algebraic decomposition: solvability becomes polyabelianness for Boolean-completeness and CIRCUIT VALUE, and a combination of polyabelianness and $\mathcal{M}$-solvability for EXPRESSION EVALUATION. The table given in the Introduction becomes the following in the case of loops:

	EXPRESSION EVALUATION	CIRCUIT VALUE
non-polyabelian	$\mathbf{NC}^1$-complete	**P**-complete
polyabelian but not $\mathcal{M}$-solvable	$\mathbf{NC}^1$-complete	$\mathbf{ACC}^1 \cap \mathbf{DET}$
polyabelian and $\mathcal{M}$-solvable	$\mathbf{TC}^0$	$\mathbf{ACC}^1 \cap \mathbf{DET}$

For groupoids in general, non-polyabelianness needs to be replaced with some other structural property as the necessary and sufficient condition for Boolean-completeness, because of the first example given in the appendix. However, the second and third rows of this table hold for all groupoids.

These results also have a language-theoretic interpretation. A regular language can be characterized by its *syntactic monoid*, the semigroup of allowed transitions of its finite-state machine. Thus the regular languages that are $\mathbf{NC}^1$-complete are exactly those whose syntactic monoid is non-solvable (assuming $\mathbf{ACC}^0$ is properly contained in $\mathbf{NC}^1$). Similarly, the set of expressions in a non-associative groupoid that evaluate to a particular element is a structured context-free language, generated by the productions $a \to (bc)$ for all b, c such that $b \cdot c = a$. Thus it seems that we may be close to showing exactly which (structured) context-free languages are $\mathbf{NC}^1$-complete.

Finally, we emphasize that our work gives further indication that, contrary to what one may think, non-associative algebras can be rigorously analysed and that their structural properties have very deep computational meaning.

Acknowledgements. J.B. assisted with this work as an intern at the Santa Fe Institute, funded by the Research Experience for Undergraduates program of the National Science Foundation. A.D. is grateful to the Santa Fe Institute for an enjoyable visit. C.M. is grateful to McGill University for their hospitality, and to Claire Riley for spicing the visit up a bit.

References

1. A.A. Albert, "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 James Corbett, "On the relative complexity of some languages in $\mathbf{NC}^1$." *Info. Proc. Lett.* **32** (1989) 251–256.
4. D. Barrington and D. Thérien, "*Finite Monoids and the Fine Structure of NC^1*", JACM **35**4(1988)941–952
5. M. Beaudry and P.McKenzie, "Circuits, Matrices, and Nonassociative Computation", Proc. of the 7th IEEE Structure in Complexity Theory, (1992) 94–106
6. M. Beaudry, P. McKenzie, P. Péladeau, and D. Thérien, "Circuits with monoidal gates." Springer LNCS, *Proc. STACS* (1993) 555–565.
7. F. Bédard, F. Lemieux and P.McKenzie, "Extensions to Barrington's M-program model", Theoretical Computer Science **107**,199331–61
8. R.H. Bruck, "Contributions to the theory of loops." *Trans. Amer. Math. Soc.* **60** (1946) 245–354.
9. R.H. Bruck, "*A Survey of Binary Systems*", Springer-Verlag, 1966.
10. S.R. Buss, "The Boolean formula value problem is in ALOGTIME." In *Proc. 18th ACM Symp. on the Theory of Computing* (1987) 123–131.
11. H. Caussinus and F. Lemieux, "*The Complexity of Computing over Quasigroups*", In the Proceedings of the 14th annual FST&TCS Conference, pp.36-47, 1994.

12. O. Chein, H.O. Pflugfelder, and J.D.H. Smith (eds.), *Quasigroups and Loops: Theory and Applications.* Heldermann Verlag, 1990.
13. M. Hall, *The Theory of Groups.* Chelsea, 1976.
14. O.H. Ibarra, T. Jiang and B. Ravikumar, "Some subclasses of context-free languages in $\mathbf{NC}^1$." *Info. Proc. Lett.* **29** (1988) 111-117.
15. J. van Leewen (editor), "Hanbook of Computer Science, Vol. A" Elsevier/MIT Press, 1990.
16. R.E. Ladner "The circuit value problem is log-space complete for P." *SIGACT News* **7:1** (1975) 18–20.
17. F. Lemieux, *Finite Groupoids and their Applications to Computational Complexity.* Ph. D. Thesis, School of Computer Science, McGill University, Montréal (1996).
18. A. Maciel and D. Thérien, "Threshold circuits of small majority-depth." Technical Report SOCS–95.5, School of Computer Science, McGill University, Montréal (1995).
19. W.D. Maurer and J.L. Rhodes, "A property of finite simple non-Abelian groups." *Proc. Am. Math. Soc.* **16** (1965) 552–554.
20. C. Moore, "Non-Abelian cellular automata." Santa Fe Institute Working Paper 95-09-81.
21. C. Moore, "Quasi-linear cellular automata." to appear in *Physica* **D** (1996).
22. H.O. Pflugfelder, *Quasigroups and Loops: An Introduction.* Heldermann Verlag, 1990.
23. H. Straubing, "Representing Functions by Words over Finite Semigroups", Université de Montréal, Technical Report #838, 1992.
24. A. Vesanen, "Solvable groups and loops." *Journal of Algebra* **180** (1996) 862–876.
25. M. Weinstein, *Examples of Groups.* Polygonal Publishing House, Passaic, New Jersey, 1977.

Appendix

Example of a non-polyabelian groupoid which is not Boolean-complete

It would be very nice if non-polyabelianness were the criterion for Boolean-completeness for groupoids in general (presumably with 'Abelian group' replaced by 'Abelian semigroup' in the definition). However, we can give a counterexample. Let $(A, \star)$ be

$$\begin{array}{c|cc|cc} \star & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 2 & 3 \\ 1 & 1 & 0 & 2 & 3 \\ \hline 2 & 2 & 2 & 0 & 0 \\ 3 & 3 & 3 & 0 & 1 \end{array}$$

Theorem 19. *$(A, \star)$ is non-polyabelian, but cannot express the* AND *function.*

Proof. The equivalence classes $\{0, 1\}$ and $\{2, 3\}$ form the only proper normal congruence $\sim$ of A. There is no Abelian semigroup on which the lower-right and upper-left local operations are both affine, so A is not polyabelian.

We could not have TRUE and FALSE in different equivalence classes, since $(A/{\sim}) \cong \mathrm{Z}_2$. So we will assume first that TRUE, FALSE $\in \{0, 1\}$.

The lower right-hand block looks like an AND gate with TRUE $= 1$ and FALSE $= 0$, in which case we could use the upper left-hand block to say $\neg a = 1 \star a$. But there is no way to map, say, 0 to 2 and 1 to 3, since $a \star b = b \star a = a$ whenever $a \in \{2, 3\}$ and $b \in \{0, 1\}$; that is, 2 and 3 dominate 0 and 1.

Formally, consider a polynomial expression ϕ in $\mathcal{P}(A)$. The following rules preserve the value of ϕ on variables in $\{0, 1\}$:

- Replace $a \star \phi$ and $\phi \star a$ with a if $a \in \{2, 3\}$ and ϕ contains only elements of $\{0, 1\}$, and
- Replace $a \star b$ with the appropriate constant in $\{0, 1\}$ if $a, b \in \{2, 3\}$.

Applying these rules repeatedly will leave us either with a constant in $\{2, 3\}$ or with variables and constants in $\{0, 1\}$, on which $\star$ is a sub groupoid isomorphic to Z_2. So any function in $\mathcal{P}(A)$ is constant or affine when restricted to variables in $\{0, 1\}$, and so cannot express an AND.

Finally, assume that TRUE, FALSE $\in \{2, 3\}$. Any node whose output is 2 or 3 is equal to either its left or its right input (whichever one is 2 or 3), which in turn is equal to one of its inputs, and so on back up to a single constant or variable. Since $\sim$ is normal, for inputs in $\{2, 3\}$ the same subexpressions always have values in $\{2, 3\}$, and this path always leads back to the same input or constant. So any function in $\mathcal{P}(A)$ with an output in $\{2, 3\}$ is either constant or equal to one of its inputs when restricted to variables in $\{2, 3\}$, and cannot be an AND. $\square$

Note that we have not shown that this groupoid's EXPRESSION EVALUATION or CIRCUIT VALUE problem is in **NC**, simply that it is not Boolean-complete. Perhaps the reader can find some more subtle analog of polyabelianness that works for all groupoids.

Example of a polyabelian loop which is not $\mathcal{M}$-solvable

Definition. For an affine quasidirect product $A \otimes B$, define $\langle fg(B) \rangle$ as the subgroup of $\mathrm{Aut}(B)$ generated by all the f's and g's in the local operations, and define $\langle fgh(B) \rangle$ as the group of affine operations on B generated by all the rows and columns of the local operations.

We need the following definition from [13]:

Definition. The *wreath product* of B by A, written $A \wr B$, is a particular kind of semidirect product $A \otimes B^A$ where B^A is the set of functions β from A to B, with multiplication defined as

$$(a_1, \beta_1) \cdot (a_2, \beta_2) = (a_1 a_2, \beta') \quad \text{where} \quad \beta'(a) = \beta_1(a_2 a) \cdot \beta_2(a)$$

In other words, each element of $A \wr B$ consists of an element of A and a vector β of $|A|$ elements of B; the β's are multiplied componentwise, but with the components of β_1 permuted by a_2.

If A and B are groups of permutations on sets S and T respectively, then $A \wr B$ is the group of permutations on $S \times T$ such that the elements of each block $\{a\} \times T$ are internally permuted by $\beta(a)$, and the blocks are permuted with each other by multiplication in A.

Then we have

Lemma 20. *If $G = A \otimes B$ is a quasigroup, then $\mathcal{M}(B, \odot)$ for every local operation $\odot$ is contained in a factor of a subgroup of $\mathcal{M}(G)$. Furthermore, if G is affine on B, the following are equivalent:*

- *G is $\mathcal{M}$-solvable*
- *$\langle fg(B) \rangle$ is solvable*
- *$\langle fgh(B) \rangle$ is solvable.*

Proof. Let $(G, \cdot) = A \otimes B$. Let H be the subgroup of $\mathcal{M}(G)$ that preserves all the blocks, and let N be the subgroup of H that fixes the elements of a particular block $\{a\} \times B$. It is easy to see that N is normal in H, and that H/N is isomorphic to the set of permutations on B that can be carried out with multiplications in G.

Using the notation of lemma 7, let one of the local operations be $b_1 \odot b_2 = a \setminus ((a_1 b_1)(a_2 b_2))$ where $a \setminus (a_1 a_2) \in B$. Multiplications in $\odot$ are compositions of multiplications in G: for instance, $R_b^{\odot} = L_a^{-1} R_{(a_2 b)} L_{a_1}$. So $\mathcal{M}(B, \odot) \subseteq H/N$.

Now suppose G is affine on B. Since the rows and columns of all the $\odot$'s generate exactly $\langle fgh(B) \rangle$, we have $\langle fgh(B) \rangle = H/N$. The subgroup of $\langle fgh(B) \rangle$ that preserves the identity of B is simply $\langle fg(B) \rangle$. Since subgroups and factors of solvable groups are solvable, both these groups are solvable if $\mathcal{M}(G)$ is.

Conversely, since elements of $\mathcal{M}(G)$ permute the blocks with each other while permuting them internally by the rows and columns of the $\odot$'s, $\mathcal{M}(G)$ is a subgroup of the wreath product $\mathcal{M}(A) \wr \langle fgh(B) \rangle$, and the wreath product of two solvable groups is solvable.

Finally, since two affine functions $\phi_1(b) = f_1(b) + h_1$ and $\phi_2(b) = f_2(b) + h_2$ compose as

$$(\phi_1 \circ \phi_2)(b) = (f_1 \circ f_2)(b) + h_1 + \phi_1(h_2)$$

we see that $\langle fgh(B) \rangle$ is a semidirect product $\langle fg(B) \rangle \otimes B$. This is solvable if $\langle fg(B) \rangle$ and B are; but B is an Abelian group, so we're done. $\square$

We can now show how to to construct a polyabelian loop which is not $\mathcal{M}$-solvable.

Let B be an Abelian group with a non-solvable automorphism group: for instance, $\mathrm{Aut}(\mathrm{Z}_2^3)$ is the simple group of order 168, and is generated by two elements [25]

$$f = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \quad \text{and} \quad g = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

If we let $G = \mathrm{Z}_2 \otimes \mathrm{Z}_2^3$ where $b_1 \odot_{1,1} b_2 = f(b_1) + g(b_2)$, then $\langle fg(B) \rangle = \mathrm{Aut}(B)$ is non-solvable and G is not $\mathcal{M}$-solvable by lemma 20. (This produces a loop of order 16; since all smaller Abelian groups have solvable automorphism groups, we believe this is the smallest possible.)

This article was processed using the LaTeX macro package with LLNCS style