Space Complexity of Reachability Testing in Labelled Graphs[*]

Vidhya Ramaswamy[†] Jayalal Sarma[†] K. S. Sunil[†]

December 15, 2016

Abstract

Fix an algebraic structure $(\mathcal{A}, *)$. Given a graph $G = (V, E)$ and the labelling function ϕ ($\phi : E \to \mathcal{A}$) for the edges, two nodes s, $t \in V$, and a subset $F \subseteq \mathcal{A}$, the $\mathcal{A}$-Reach problem asks if there is a path p (need not be simple) from s to t whose yield (result of the operation in the ordered set of the labels of the edges constituting the path) is in F. On the complexity frontier of this problem, we show the following results.

- When $\mathcal{A}$ is a group whose size is polynomially bounded in the size of the graph (hence equivalently presented as a multiplication table at the input), and the graph is undirected, the $\mathcal{A}$-Reach problem is in L. Building on this, using a decomposition in [4], we show that, when $\mathcal{A}$ is a fixed quasi-group, and the graph is undirected, the $\mathcal{A}$-Reach problem is in L. In a contrast, we show NL-hardness of the problem over bidirected graphs, when $\mathcal{A}$ is a matrix group over $\mathbb{Q}$, or an aperiodic monoid.
- As our main theorem, we prove a dichotomy for graphs labelled with fixed aperiodic monoids by showing that for every fixed aperiodic monoid $\mathcal{A}$, $\mathcal{A}$-Reach problem is either in L or is NL-complete.
- We show that there exists a monoid M, such that the reachability problem in general DAGs can be reduced to $\mathcal{A}$-Reach problem for planar non-bipartite DAGs labelled with M. In contrast, we show that if the planar DAGs that we obtain above are bipartite, the problem can be further reduced to reachability testing in planar DAGs and hence is in UL.

[*]A preliminary version of this paper has been accepted for presentation at the *11th International Conference on Language and Automata Theory and Applications* (LATA 2017).

[†]Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India. Email : `vidhya0794@gmail.com,jayalal@cse.iitm.ac.in,sunil@cse.iitm.ac.in`.

1 Introduction

The reachability problem on combinatorial structures has been fundamental and well studied in complexity theory. A most striking example of this is the graph reachability problem which asks, given a directed graph G and two special vertices s and t whether there is a path from s to t in G or not. The problem is known to be NL-complete for directed acyclic graphs. Deterministic logspace algorithms are known for restricted classes of graphs - when each component of the directed graph is Eulerian[19], or the graph has bounded treewidth [9]. Reachability for planar graphs is in unambiguous[1] logspace [6]. See [1] for a survey.

Word problems over algebraic structures also play a fundamental role in complexity theoretic characterizations. Fix an algebraic structure $\mathcal{A}$ with an associated binary operation, and a subset $F \subseteq \mathcal{A}$. Given a $w \in \mathcal{A}^*$, test if the sequences of elements and operations among them is in F or not. An important milestone in this direction is the dichotomy result due to Barrington and Thérein[3] classifying the complexity of the word problems over a fixed monoid structure: if the monoid M contains at least one non-solvable group, then the word problem can be shown to be complete for NC^1 (under AC^0 projections) and if all groups are solvable then it characterizes ACC^0. Chandra *et al* [8] showed that if there are no non-trivial groups then it characterizes the class AC^0. It is also known that word problems over groupoids characterize LogCFL[5].

Reachability on labelled graphs is a natural generalization of the graph reachability problem and the word problem on algebraic structures. A graph G is said to be *labelled* if the edges are assigned labels from an underlying set S. When this set also equipped a binary operation $* : S \times S \to S$, the reachability problem asks to test, given the graph G and two vertices s and t and an element $a \in S$, if there is a path (need not be simple) from s to t whose yield (result of the operation in the ordered set of the labels of the edges constituting the path) is a or not. A closely related problem is that of the L-Reach problem where a language L over the alphabet Σ, given a graph $G(V, E)$, two vertices s and t and a labelling function $\phi : E \to \Sigma$, test if there is a path from s to t whose yield (the concatenation of the labels in the ordered set of edges constituting the path) belongs to the language L. In [15], characterizations of the language reachability problem with respect to languages classes and graph classes were obtained.

In this work, we study the problem when the labels come from richer algebraic structures. When the structure $\mathcal{A}$ is a groupoid (equivalently a case of L-Reach problem when the language is restricted to be a context-free language) this problem has been used in inter-procedural slicing and inter-procedural data flow analysis [12, 20, 21]. On the complexity frontier, it is easy to observe that the $\mathcal{A}$-Reach problem is always harder than the word problem over $\mathcal{A}$, and is harder than the graph reachability problem (under logspace many-one reductions).

Our Results: We start with an observation that the problem of testing reachability on labelled graphs over semigroups can be reduced to testing reachability on an associated

[1]A language is said to be in unambiguous logspace if there exists a non-deterministic logspace Turing machine M such that for all inputs x, M has at most one accepting computation.

directed graph (called product graphs - see section 3.1). From a complexity-theoretic view point, this motivates the study of the labelled reachability problem. More specifically, in order to show that for a graph class the reachability problem is in L, it is sufficient to reduce it to the reachability problem in a labelled graph such that reachability in the product graph can be solved in L. We prove several properties of this directed graph and explore graph classes and algebraic structures for which the $\mathcal{A}$-REACH problem is in L. In particular, we study this for undirected graphs (for which reachability problem is in L[18]) and show:

Logspace Upper Bounds for Polynomially Growing Groups: We show that when $\mathcal{A}$ is a group, such that $|\mathcal{A}| = O(n^c)$, where n is the size of the graph, and c is a constant (hence equivalently presented as a multiplication table at the input), and the graph is undirected, the $\mathcal{A}$-REACH problem is L-complete.

NL-hardness for Monoids and Matrix Groups: In contrast, we observe that there exists a fixed monoid $\mathcal{A}$ such that $\mathcal{A}$-REACH problem for undirected graphs is NL-complete. Working over a more structured labelling set, we show NL-hardness of the problem over bidirected graphs[2], when $\mathcal{A}$ is a finitely generated subgroup of $\mathsf{GL}_k(\mathbb{Q})$ (for $k \geq 2$)- the group of invertible $k \times k$ matrices with rational entries.

(NL vs L) Dichotomy for Aperiodic Monoids: We show a dichotomy for aperiodic monoids: for any fixed aperiodic monoid $\mathcal{A}$, the $\mathcal{A}$-REACH problem for undirected graphs is either NL-complete or is in L.

Logspace Upper Bounds for Quasigroups: When $\mathcal{A}$ is a fixed quasigroup, the $\mathcal{A}$-REACH problem is L-complete.

Logspace Upper Bounds for Treewidth k Graphs labelled with Monoids: When the graph has bounded treewidth, for any fixed monoid $\mathcal{A}$ the $\mathcal{A}$-REACH problem for undirected graphs is L-complete.

While the general DAGREACH problem is complete for NL, the reachability problem over planar DAGs is known to be in UL[6]. We show that DAGREACH can be reduced to $\mathcal{A}$-REACH over planar DAGs when $\mathcal{A}$ is a specific monoid M. Tightly complementing this, we show that the instances of labelled graph reachability problem obtained in this reduction, with the additional restriction that the graph is bipartite, can be solved in deterministic logspace. Moving towards groups, we show that DAGREACH can be reduced to $\mathcal{A}$-REACH over planar graph when $\mathcal{A}$ is a specific exponentially growing group.

Related Work: Group labelled graphs have been extensively studied in the literature as a generalization of signed graphs (see [11]) with the aim to extend the graph minor theory to group labelled graphs over a fixed finite Abelian group. An important comparison that we make is with the results by Kawase *et al*[14], where they consider the reachability problem in group labelled graphs: to check if there is a *simple path* from s to t in the given group labelled graph so that the yield of the path is a given element α. It is observed in [14] that the problem is NP-complete over $\mathbb{Z}$, since the undirected Hamiltonian path problem reduces to this problem by replacing each edge with a pair of two arcs of opposite directions with label 1 and letting $\alpha = n - 1$. Huynh[13] showed the problem is polynomial time solvable if the group is a fixed Abelian group.

[2] Directed graph $(G(V, E))$ such that $\forall v_i, v_j \in V, (v_i, v_j) \in E \implies (v_j, v_i) \in E$. However, $\phi(v_i, v_j)$ need not be equal to $\phi(v_j, v_i)$. To complement this, we observe (see Corollary 1) that the logspace upper bound for groups whose size is polynomially bounded in terms of input, holds even for bidirected graphs.

However, we point out two important differences in our setting. Firstly, in our setting, the problem does not look for simple paths from s to t, and hence the NP-completeness result does not apply. Secondly, for the case of undirected graphs with labelling from a group, the above mentioned results (including [14]) assume that if an edge (u, v) in the graph is labelled with g, the edge (v, u) is labelled with g^{-1}. In our setting, this is not the case - if an edge in the undirected graph is labelled $g \in G$, then the edge contributes to the final product as g itself *irrespective* of the direction that is taken by the path through the edge. Thus, the above results of the problem do not apply in our case.

2 Preliminaries

In this section, we list notations and preliminaries used in the paper. For standard notations and definitions of complexity classes, we refer the reader to the textbook[2]. We now define some graph theoretic terminologies required. A *tree decomposition* of a graph $G = (V, E)$ is a tree $T = (I, F)$ where each vertex $i \in I$ has a label $X_i \subseteq V$ with $\bigcup_{i \in I} X_i = V$ such that: for any edge $(u, v) \in E$, there exists an $i \in I$ with $u, v \in X_i$ and, for any $v \in V$, the vertices containing v in their label form a connected subtree of T. Given a tree decomposition $T = (I, F)$, the *width* of the decomposition is $\max_{i \in I}(|X_i|) - 1$. The *treewidth* of a graph G is the minimum k such that G has a tree decomposition of width k.

We define the algebraic structures that we refer to in the paper. Let $\mathcal{A}$ be an algebraic structure where $*$ is the binary operator. If $*$ has the closure property, $\mathcal{A}$ is said to be a groupoid. Groupoids for which $*$ is associative are called semi-groups. Semi-groups which have an identity element e (that is, $\forall a \in \mathcal{A}, ae = ea = a$)[3] are called monoids. Groups are monoids for which every element has an inverse with respect to $*$. That is, $\forall a \in \mathcal{A}, \exists\ b \in A$ such that $ab = ba = e$. In general, a monoid M is said to be divided by another monoid N if there exists a surjective morphism from a submonoid of M to N [22]. Quasigroups are a generalization of groups in a different direction; the operation in a quasigroup need not be associative but they are left and right cancellative (that is, $ab = ac \Rightarrow b = c$ and $ab = cb \Rightarrow a = c$).

$\mathcal{A}$-REACH Problem: Let $\mathbf{A} = \{(\mathcal{A}, *)\}$ be an infinite collection of algebraic structures where each $(\mathcal{A}, *)$ is the algebraic structure with set of elements $[k] = \{1, 2, \ldots, k\}$ and the binary operation $*$ defined over $\mathcal{A}$. Let F be a subset of $\mathcal{A}$. Consider a graph $G = (V, E)$ and a function $\phi : E \to \mathcal{A}$. We extend the definition of ϕ to the yield of a path $p = v_0, v_1, \ldots, v_m$, as $\phi(p) = \prod_{i=0}^{m-1} \phi((v_i, v_{i+1}))$ where product is the operation $*$ on the concatenated labels of p.

Definition 1. *($\mathcal{A}$-REACH) Fix an algebraic structure $\mathcal{A}$. The $\mathcal{A}$-REACH problem asks: given a graph G on n vertices and the labelling function ϕ for the edges, two nodes s, t and*

[3]We do not use the operator, whenever it is clear from the context. We use 1 and e interchangeably for the identity element.

accepting set[4] $F \subseteq \mathcal{A}$*, test whether there is a path* p *(need not be simple) from* s *to* t *such that* $\phi(p) \in F$*.*

For studying the variants of the problem, we introduce the following notation: A-GReach refers to the $\mathcal{A}$-Reach problem defined over the algebraic structure A(Monoid, Aperiodic Monoid, Commutative Aperiodic Monoid, Group, Quasigroup and Semigroup) and the input graphs are restricted to the class G(Tree, Planar, DAG, k-Treewidth, Undirected(U) and Bidirected(B)).

Aperiodic Monoids and Quasigroups: The monoid class **DA** is defined as the class of monoids that satisfy $(stu)^n t (stu)^n = (stu)^n$, for some n, for all s, t, u in the monoid. A language $L = A_0^* a_1 A_1^* a_2 \cdots a_k A_k^*$ is said to be unambiguous if for all $w \in L$, there is a unique factorization $w = w_0 a_1 w_1 a_2 \cdots a_k w_k$, such that $w_i \in A_i^*$ for $i = 0, 1, \ldots, k$. Pin *et al* [16] showed the following characterization:

Proposition 1. *[16]* $L \subseteq A^*$ *is recognized by a monoid in* **DA** *if and only if* L *is a disjoint, finite union of unambiguous products* $A_0^* a_1 A_1^* \cdots a_k A_k^*$*, where* $A_i \subseteq A, a_i \in A$*, for* $i \in [k]$*.*

The following was proved by Raymond *et al* [17, 22].

Proposition 2. *([17, 22]) Let* M *be a finite, non-commutative monoid. Then* M *is divided by one of the following aperiodic monoids.*

1. BA_2*, the syntactic monoid*[5] *of* $(c^*ac^*bc^*)^*$*.*
2. U*, the syntactic monoid of* $((b+c)^*a(b+c)^*b(b+c)^*)^*$
3. *The syntactic monoid of* $c^*ac^*bc^*$*.*
4. *The syntactic monoid of* c^*aA^* *or* A^*ac^**.*

Moreover, if $M \notin$ **DA**, M *is divided by either* BA_2 *or* U*.*

In [4], Beaudry *et al* define a *comb* as a left to right bracketing over a word, and claim that any bracketing over the word, for a quasigroup, can be viewed as a finite tree with each leaf as a comb. We state this as the following:

Proposition 3. *([4]) Let* $q_1 q_2 \cdots q_n$ *be a word over a quasigroup* Q*. If there is a bracketing such that* $q_1 q_2 \cdots q_n$ *evaluates to* q *under that bracketing, then there is a bracketing with at most* $8^{|Q|}$ *combs which yields* q*.*

[4]If the size of $\mathcal{A}$ is fixed (or even polynomially bounded) we will assume that $|F| = 1$. We also assume that the accepting element a is given as a part of the input. All our results except Theorem 8 hold even if a is fixed apriori.

[5]We denote the elements of this monoid by $\{1, \alpha, \beta, \alpha\beta, \beta\alpha, 0\}$

3 Logspace Upper Bounds

In this section, we explore the algebraic structure of the label set and graphs which enable us to solve the problem in L. As our main tool, we introduce the product graph, which is inspired by that of product graphs defined in the context of L-Reach problem by Yannakakis *et al*[23].

3.1 Product Graph and Properties

Let $G = (V, E)$ be a labelled graph, with a labelling $\phi : E \to M$, where M is a semigroup. We construct a new directed graph $G' = (V', E')$ as follows. We set $V' = V \times M$, and define the edge set E' as $\{((v_1, m_1), (v_2, m_2)) | (v_1, v_2) \in E \text{ and } m_1\phi(v_1, v_2) = m_2\}$. We show the following proposition.

Proposition 4. *For $s, t \in V$, $m \in M$, there is a path p from s to t in G such that $\phi(p) = m$ if and only if there is a path from (s, e) to (t, m) in G'.*

Proof. Suppose we have a path in G from s to t yielding m, say $s = v_1, v_2, \ldots, v_k = t$. By construction, G' has edges from (v_i, m_i) to $(v_{i+1}, m_i\phi(v_i, v_{i+1}))$. In particular, we have edges from (s, e) to $(v_2, \phi(s, v_2))$, $(v_2, \phi(s, v_2))$ to $(v_3, \phi(s, v_2)\phi(v_2, v_3))$ and so on. Since $\prod_{i=1}^{k-1} \phi(v_i, v_{i+1}) = m$, this gives us a path from (s, e) to (t, m). Suppose we have a path from (s, e) to (t, m) in G'. Let this path be $(s = v_1, e = m_1), (v_2, m_2), (v_3, m_3), \ldots, (v_k = t, m_k = m)$. By construction, $s = v_1, v_2, v_3, \ldots v_k = t$ is a path in G. Also, for all i, $m_i\phi(v_i, v_{i+1}) = m_{i+1}$. Hence, $\prod_{i=1}^{k-1} \phi(v_i, v_{i+1}) = m$. □ □

Similarly, we can argue that a path from (s, m_1) to (t, m_2) in G' exists if and only if there is a path from s to t in G, yielding m such that $m_1 m = m_2$.

NL Upper Bounds: Since the above proposition holds even if G has cycles in it, this implies that SemiGroup-Reach is in NL. In later sections, we show more properties of the product graph. The product graph of an undirected graph labelled with a group is Eulerian. See Theorem 2. Also, the product graph of a graph with a bounded treewidth (labelled with a finite monoid) also has bounded treewidth. See Theorem 1.

If the algebraic structure is non-associative, we have to deal with all possible bracketings. Let us denote the set of all elements obtained by different bracketings of a word w by Yield(w). Caussinus *et al* [7] showed that languages recognized by finite quasigroups are regular. Hence, there exists a morphism ψ from any quasigroup Q to a monoid M, and subsets $Q' \subseteq Q, M' \subseteq M$ such that for any word $w \in Q^*$, Yield$(w) \cap Q' \neq \phi$ if and only if $\psi(w) \in M'$. Hence, the product graph construction shows that Quasigroup-Reach can be solved in NL.

Original Graph vs Product Graph while Group Labelling: It is a natural question to ask when the original graph appears as a subgraph in the product graph. We answer this for group labelled graphs.

Let $G = (V, E)$ be a directed acyclic graph, labelled with a group H, via labelling ϕ. Let the product graph be G'. Suppose G is a tree. We show that G' contains a copy of G. Let us start with any vertex v. For any $g \in H$, (v, g) is in G'. Now, consider all neighbors of

v in G. If (v,u) is an edge in G, we have a corresponding edge $((v,g),(u,g\phi(v,u)))$ in G'. Similarly, if (u,v) is an edge in G, $((u,g\phi(u,v)^{-1}),(v,g))$ is an edge in G'. Continuing in a breadth first search manner, we get a copy of G in G'. Hence, it is easy to see that if G is a tree, G' contains G as a subgraph. To extend this to general DAGs, we need to understand when (undirected) cycles of G appear in G'. If all cycles (undirected) of G appear in the product graph, G also appears in the product graph.

Let $C = v_1, v_2, \ldots v_k$ be an undirected cycle in G. We define $\psi(v_i,v_j)$ as follows: $\psi(v_i,v_j) = \phi(v_i,v_j)$ if $(v_i,v_j) \in E$ and $\phi(v_j,v_i)^{-1}$ if $(v_j,v_i) \in E$.

Proposition 5. *G appears as a subgraph in G' if and only if for each cycle $C = v_1, v_2, \ldots v_k$ in G, $\prod_{i=1}^{k} \psi(v_i,v_{i+1}) = \psi(v_k,v_1)^{-1}$*

Proof. It is sufficient to prove the following claim: C is present in G' if and only if $\psi(v_1,v_2)\psi(v_2,v_3)\cdots\psi(v_{k-}$
1, where 1 is the identity element of H. We argue both implications. The forward direction follows from the definition of G'. Indeed, the product graph has edges connecting (v_i,g) and $(v_{i+1}, g\psi(v_i,v_{i+1}))$, for all $i \in \{1,2,\ldots k-1\}, g \in H$. Hence, the product graph has the (undirected) path

$$(v_1,g),(v_2,g\psi(v_1,v_2)),(v_3,g\psi(v_1,v_2)\psi(v_2,v_3)),\ldots,(v_k,g\prod_{i=1}^{k-1}\psi(v_i,v_{i+1})).$$

It also has an edge between $(v_k, g\prod_{i=1}^{k-1}\psi(v_i,v_{i+1}))$ and $(v_1, g(\prod_{i=1}^{k-1}\psi(v_i,v_{i+1})\psi(v_k,v_1) = g)$. Hence, the cycle exists in G' for each $g \in H$.

For the reverse direction, suppose $\psi(v_1,v_2)\psi(v_2,v_3)\ldots\psi(v_{k-1},v_k)\psi(v_kv_1) = h$. Now, for every $g \in H$, consider the vertex (v_1,g). As seen earlier, we have an (undirected) path $(v_1,g),(v_2,g\psi(v_1,v_2)),(v_3,g\psi(v_1,v_2)\psi(v_2,v_3)),\ldots(v_k,g\prod_{i=1}^{k-1}\psi(v_i,v_{i+1}))$. The last edge, however is between $(v_k,g\prod_{i=1}^{k-1}\psi(v_i,v_{i+1}))$ and (v_1,gh), not $(v_k,g\prod_{i=1}^{k-1}\psi(v_i,v_{i+1}))$ and (v_1,g). Since this is true for all g, this cycle never appears in G'.

□ □

3.2 Bounded Treewidth and Monoid Labelling

Das *et al* [9] showed that reachability in bounded treewidth graphs can be tested in L. We show that, when a bounded treewidth graph G is labelled with a constant sized monoid M, the product graph of G still has constant treewidth, and hence, reachability in the labelled graph is also in L.

Theorem 1. Monoid-k-TreewidthReach *is in* L.

Proof. Let M be the given finite monoid, and $G = (V,E)$ be the given graph, with labelling $\phi : E \to M$. Since the treewidth of G is given to be a constant k, we can compute a tree decomposition T of G in L [10]. Let G' be the product graph of G. We give a tree decomposition T' for G' such that the width is $(k+1)|M|-1$. Since k and M are constants, the treewidth of G' is a constant. We create T' as follows. For each node $n =$

$\{v_{i_0}, v_{i_1}, \ldots, v_{i_k}\} \in T$, we create node $n' = \{(v_{i_j}, m) | 0 \leq j \leq k, m \in M\}$. For each edge $(n_i, n_j) \in T$, we add edge $(n'_i, n'_j) \in T'$. We notice that T' is a tree, and the size of each node n' is a constant. We need to show that for each edge $(v_i, m), (v_j, m') \in E'$, there exists a node n'_p, such that $(v_i, m), (v_j, m') \in n'_p$. Since $(v_i, v_j) \in E$, there exists an n_ℓ such that $v_i, v_j \in n_\ell$. Thus, by our definition of T', $(v_i, m), (v_j, m') \in n'_\ell$. The last condition for T' to be a constant treewidth decomposition of G' is that if nodes n'_i and n'_j contain vertex (v, m), all nodes in the path between n'_i and n'_j should also contain (v, m). This is immediately true, as all nodes in the path between n_i and n_j in T contain v, and hence, in T', they contain (v, m). Hence, G' has constant treewidth, and connectivity in G' can be tested in L [9]. □ □

3.3 Group Labelled Graphs

Now we show that the $\mathcal{A}$-REACH problem can be solved in L, when the graph is undirected, and labelled with elements of a group, when the group size is polynomial in the size of the graph.

Theorem 2. GROUP-UREACH *is* L*-complete.*

Proof. To show that GROUP-UREACH is in L, we reduce the problem to REACH on Eulerian graphs, by showing that the product graph G' is Eulerian. From [19] we know that this problem can be solved in L and hence, this is sufficient. To solve this in L, Reingold *et al* [19] observed (without proof) that, when each component of the given directed graph is Eulerian, a directed edge can be replaced by an undirected edge, and this does not alter connectivity of the graph. For completeness, we include the proof of this in the appendix A.1.
To show that G' is Eulerian, consider an edge (v_i, v_j) in G. Let $\phi((v_i, v_j)) = g$. Each vertex (v_i, g_k), is hence connected to $(v_j, g_k g)$. We notice that for each k, $g_k g$ defines a different element in H. Similarly, each vertex (v_j, g_ℓ) is adjacent to $(v_i, g_\ell g)$. Hence, the edge (v_i, v_j) in G corresponds to $2|H|$ edges in G', and these edges are such that each vertex of the form (v_i, g_k) and (v_j, g_ℓ) each have an indegree of 1 and an outdegree of 1. Since each edge in G increases the indegree and outdegree of any vertex in G' by the same amount, G' is Eulerian. Using the result from [19] we see that GROUP-UREACH is in L. To show hardness, we see that GROUP-UREACH is the undirected reachability problem when the underlying group is trivial. Hence, GROUP-UREACH is complete for L. □ □

Observing that, for any $g \in G$, $g_k g$, is a different element for all k, and that each edge (v_i, v_j) in G gives rise to one incoming and one outgoing edge for each (v_i, g_k) holds even when the graph is bidirected. Hence, we conclude the following corollary.

Corollary 1. GROUP-BREACH *is* L*-complete.*

3.4 Logspace algorithm for Quasigroup-UReach

We notice from the proof of Theorem 2, that the product graph G' is Eulerian if the H has right cancellation, that is, if $ab = cb \Rightarrow a = c$. Since this holds for quasigroups as well, the constructed graph is Eulerian when H is a quasigroup. However, since evaluation of a word is over all possible bracketings, checking for a path from (s, e) to (t, h) is no longer sufficient (since this would correspond to only checking a left to right bracketing). We use Proposition 3 to prove the below theorem.

Theorem 3. Quasigroup-UReach *is* L*-complete.*

Proof. Let $G = (V, E)$ be the given graph, labelled with a quasigroup Q, via a mapping ϕ. Let $s, t \in V$ and $q \in Q$. We denote the all possible bracketings among $8^{|Q|}$ combs by T. Let $k = 8^{|Q|}$.

The main idea of the algorithm is to break a walk from s to t into finitely many subwords, evaluate each subword as a comb, and check if any bracketing over results of the comb gives us the required element.

```
Input: Graph $G = (V, E)$, Quasigroup $Q$, mapping $\phi : E \to Q$, $s, t \in V, q \in Q$
Output: Accept, if there exists a walk from $s$ to $t$ yielding $q$ and Reject,
        otherwise.
Notation: $k = 8^{|Q|}$, $G'$ is the product graph of $G$.
$T$ is the set of all possible bracketings between $8^{|Q|}$ combs.
for $(q_1, q_2, \ldots, q_k) \in Q^k$ do
    for $(v_1, v_2, \ldots, v_{k-1}) \in V^{k-1}$ do
        Initialize $v_0 = s; v_k = t$;
        if $\forall i$ there is a path from $(v_{i-1}, e)$ to $(v_i, q_i)$ in $G'$ then
            if $\exists \tau \in T$ such that $\tau(q_1, q_2, \ldots, q_k) = q$ then
                Accept
            end
        end
    end
end
Reject
```

Algorithm 1: Deterministic Logspace Algorithm for Quasigroup-UReach

We claim that above algorithm solves the problem in L, when Q is finite. This follows since k is finite, implying V^{k-1} is polynomial in the size of the input, and Q^k and T are finite. Since the algorithm runs over all possible values of $(q_1, q_2, \ldots, q_k)$, we see that it checks all possible values the k combs can take. By running over $(v_1, v_2, \ldots, v_{k-1})$, it also verifies if there are paths in the graph evaluating to corresponding quasigroup element. Moreover, the algorithm also verifies that there is a bracketing between $(q_1, q_2, \ldots, q_k)$ which yields q. From Proposition 3, we know that this gives us all possible evaluations of the word. Since the algorithm only accepts when there is a path and a bracketing which yields q, the other direction follows. Hence, the algorithm is correct.

To show hardness, we notice that QUASIGROUP-UREACH is equivalent to testing reachability in undirected graphs when the quasigroup is trivial. Hence, QUASIGROUP-UREACH is complete for L.

□ □

4 Symmetrizing by Labelling

In this section, we explore the question of whether we can reduce (in logspace) reachability over directed acyclic graphs to labelled reachability over undirected paths. We call this task as *symmetrization by labelling*. We first observe that symmetrization can be done when the algebraic structure is a specific aperiodic monoid or a specific, finitely generated, matrix group over $\mathbb{Q}$.

4.1 Labelling with Aperiodic Monoids

We give a labelling with a non-commutative aperiodic monoid, which makes the $\mathcal{A}$-REACH problem NL-hard. In [15], Komarath *et al* give a labelling with $(ab)^*$, for all directed acyclic graphs. We show that the syntactic monoid of this language is aperiodic. We give the multiplication table of the monoid below:

	0	1	α	β	$\alpha\beta$	$\beta\alpha$
0	0	0	0	0	0	0
1	0	1	α	β	$\alpha\beta$	$\beta\alpha$
α	0	α	0	$\alpha\beta$	0	α
β	0	β	$\beta\alpha$	0	β	0
$\alpha\beta$	0	$\alpha\beta$	α	0	$\alpha\beta$	0
$\beta\alpha$	0	$\beta\alpha$	0	β	0	$\beta\alpha$

To see that it is aperiodic, we verify that for all a in the monoid, $a^3 = a^2$. Hence, this monoid is aperiodic with index 2. We also observe that the monoid is non-commutative.

4.2 Labelling with a Finitely Presented Group

In this subsection, we show that for matrix groups (even of size 2) with entries from $\mathbb{Q}$, symmetrization can be done. In section 3, we saw that if symmetrization is done when the algebraic structure is either a polynomially growing group or a fixed size quasigroup, it implies that NL = L.

Theorem 4. *$\mathcal{A}$ is the group of invertible $k \times k$ matrices with rational entries. $\mathcal{A}$-BREACH is NL-hard.*

Proof. We first show this for $k = 2$. We work over the following subgroup, $H = \left\{ \begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix} : \alpha \in \mathbb{Z} \right\}$. This group is finitely generated, since $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$ generate H. We define element $a = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. Given an instance $(G(V, E), s, t)$ of REACH we construct an instance $(G'(V', E'), H, s, t, e)$ of GROUP-BREACH as follows. For every edge $(v_i, v_j) \in E$, we add 2 edges (v_i, v_j) and (v_j, v_i) to E'. We label edge (v_i, v_j) with e, and edge (v_j, v_i) with a.

We now argue correctness of this construction. Suppose there was a directed path from s to t in G. Let this path be $v_0 = s, v_1, v_2, \ldots, v_k = t$. Now, in G', we have the same path. Moreover, since each edge within the path is labelled with e, the entire path multiplies out to the identity element. Thus, we have a path from s to t in G' whose yield is identity.

Suppose there is no path from s to t in G, but there is a path from s to t in G' which yields identity. Let this path be $s = v_0, v_1, v_2, \ldots v_m = t$. Since this path does not exist in G, there must be an i such that $(v_i, v_{i+1}) \notin E$. Hence, the label on this edge must be a. We can have several edges like this in the path. Thus, the yield of the path is a^k, for some $k \geq 1$. Since the path yields identity, we have

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

However, we see that $a^k = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$. Hence, the yield cannot be identity, and there is no path from s to t in G' yielding identity.

To extend this to $k \times k$ matrices, we notice that we can embed a into a $k \times k$ matrix b by setting

$$b[i, j] = \begin{cases} a[i, j], \text{ if } i \leq 2, j \leq 2 \\ 1, \text{ if } i, j > 2, \text{ and } i = j \\ 0, \text{ if } i, j > 2, \text{ and } i \neq j \end{cases}$$

The forward direction of the proof is easy to see. The reverse direction follows from the fact that b^k can never be identity, for $k > 1$. □ □

5 A Dichotomy Theorem for Aperiodic Labelling

In this section, we prove the main result of the paper, which is the dichotomy theorem for finite aperiodic monoids with respect to the reachability in labelled graphs. We first settle the complexity in the case of commutative monoids. For non-commutative monoids, we show the dichotomy using the classification of aperiodic monoids by [17, 22] (see Proposition 2). We show deterministic logspace algorithms when the monoid is in DA and for the other cases (when the monoids are divided by U or BA_2), we show that the reachability is complete for NL.

5.1 Logspace algorithm for CommutativeAperiodic-UReach

Let $M = \{1, \alpha_1, \alpha_2, \ldots, \alpha_k\}$ be a commutative aperiodic monoid.

Theorem 5. *Let G be an undirected graph labelled with elements from M, a commutative aperiodic monoid. Checking if there is a path from s to t which evaluates to an element α can be done in L.*

Proof. We notice that any element α in M can be thought of as several tuples of integers $(n_1, n_2, \ldots, n_k)$, such that $\alpha = \alpha_1^{n_1}\alpha_2^{n_2}\cdots\alpha_k^{n_k}$. Hence, checking if a path evaluates to particular element is equivalent to checking if the the number of occurrences of each element in each path is one of the tuples associated with the element. We also know that M is aperiodic with index q ($\forall \alpha \in M, \alpha^{q+1} = \alpha^q$). This implies that, if $\alpha = \alpha_1^{n_1}\alpha_2^{n_2}\cdots\alpha_i^{q}\cdots\alpha_k^{n_k}$, then $\alpha = \alpha_1^{n_1}\alpha_2^{n_2}\cdots\alpha_i^{q+1}\cdots\alpha_k^{n_k}$. Hence, checking for tuples where each value is bounded by q is sufficient.

Let $G = (V, E)$ be the given graph, labelled with a commutative aperiodic monoid M, via a mapping ϕ. Let $s, t \in V$ and $(n_1, n_2, \ldots, n_k)$ be a tuple, where $n_i \leq q, \forall i$. Let $N = \sum_i n_i$. The algorithm is as follows:

```
Input: Graph $G = (V, E)$, Aperiodic Commutative Monoid $M = \{\alpha_1, \alpha_2, \cdots, \alpha_k\}$
       with index $q$, $T = (n_1, n_2, \ldots, n_k)$, mapping $\phi : E \to M$, $s, t \in V$. Let
       $N = n_1 + n_2 + \cdots + n_k$
Output: Accept, if there exists a walk from $s$ to $t$ whose yield is $T$ Reject,
        otherwise
for $(u_1, v_1), (u_2, v_2), \ldots, (u_N, v_N)$ in $E^N$ do
    if Labels of $(u_1, v_1) \ldots (u_N, v_N)$ is $T$ then
        $v_0 = s; u_{k+1} = t;$
        $P = \{e\} \cup \{\alpha_i | n_i = q\}$
        Let $G'$ be $G$ with edges labelled only from the set $P$.
        if $\forall i$ there is a path from $v_i$ to $u_{i+1}$ in $G'$ then
            Accept
        end
    end
end
Reject
```

Algorithm 2: CommutativeAperiodic-UReach in in L

We see that the algorithm uses only logspace, since N is at most qk.

Correctness: The algorithm iterates over all possible edges, such that the labels of the edges give the tuple. For each set of edges, it verifies if there is a path between these edges, which uses only those labels which have crossed the index (captured by set P). This ensures that the resulting path also evaluates to the same element.

For the reverse direction, since every graph accepted by this algorithm has a path whose tuple is of the form $(n'_1, n'_2, \ldots, n'_k)$, where $n'_i = n_i$ if $n_i < q$, and $n'_i \geq n_i$ if $n_i = q$. Hence, the elements that both these tuples evaluate to must be the same. □ □

5.2 Logspace algorithm for DA-UREACH

We give a logspace algorithm to solve DA-UREACH, when the graph is labelled with letters from an unambiguous concatenation $L = A_0^*a_1A_1^*a_2 \cdots a_kA_k^*$, where A is the alphabet, $A_i \subseteq A, a_i \in A, \forall i$. From Proposition 1, this is sufficient to show that DA-UREACH can be solved in logspace.

Theorem 6. *Let G be an undirected graph labelled with elements from an alphabet A. Let s and t be given vertices in G. Let $L = A_0^*a_1A_1a_2A_2^* \cdots a_kA_k^*$ be an unambiguous concatenation, where $A_i \subseteq A, a_i \in A, \forall i$. Checking if there is a path from s to t, whose yield is in L can be done in logspace.*

Proof. Let $G = (V, E)$ be the given graph, labelled with an alphabet A, via a mapping ϕ. Let $L = A_0^*a_1A_1^*a_2 \cdots a_kA_k^*$. Let $s, t \in V$. The algorithm does the following:

```
Input: Graph $G = (V, E)$, Alphabet $A$, $L = A_0^*a_1A_1^*a_2 \cdots a_kA_k^*$, mapping $\phi : E \to A$,
       $s, t \in V$
Output:
ACCEPT, if there exists a walk from $s$ to $t$ whose yield is in $L$
REJECT, otherwise
for $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$ in $E^k$ do
    if $\forall i$, $\phi(u_i, v_i) = a_i$ then
        $v_0 = s$
        $u_{k+1} = t$
        Let $G_i$ be $G$ with only the edges labelled with elements from $A_i$.
        if $\forall i$, there is a path from $v_i$ to $u_{i+1}$ in $G_i$ then
            ACCEPT
        end
    end
end
REJECT
```

Algorithm 3: Determinsitic Logspace Algorithm for DA-UREACH

Since k is finite, we see that the algorithm chooses all possible edges for the a_i's, and check if paths between these edges are in A_i^*. The algorithm uses only logspace. The correctness of this algorithm is easy to see - if there exists a path from s to t in L, the algorithm will eventually find it, since it runs over all possible edges. For the other direction, we notice that the algorithm only accepts paths in L. □ □

The above algorithm does not use the fact that the concatenation is unambiguous. We show that, if the concatenation is not unambiguous, the resultant language is not recognized by an aperiodic monoid. We know that any aperiodic monoid not in **DA** can be divided by one of BA_2 or U. We argue that the languages recognized by BA_2 or U cannot be written as an concatenation of the form $A_0^*a_1A_1^*a_2 \cdots a_kA_k^*$. Hence, no language which divides these monoids can be written in that form.

Theorem 7. *BA_2, the syntactic monoid of $L_1 = (c^*ac^*bc^*)^*$ and U, the syntactic monoid of $L_2 = ((b+c)^*a(b+c)^*b(b+c)^*)^*$ cannot be expressed as a disjoint union of products $A_0^*a_1A_1^* \cdots a_kA_k^*$, where $A_i \subseteq A, a_i \in A$, for $i \in [k]$.*

Proof. **Case 1**: BA_2
If any of the A_i, $0 \leq i \leq k$, contains a or b, then the A_i^* can generate strings which contains consecutive a's or b's. But it is clear that there is no two consecutive a's and no two consecutive b's in L_1. So for all i, A_i can contains only c's and the language generated by the product $A_0^*a_1A_1^* \cdots a_kA_k^*$ consists only finite a's and b's. Hence, BA_2 cannot be expressed as the disjoint union of the products $A_0^*a_1A_1^* \cdots a_kA_k^*$.
Case 2: U
Similar argument holds for L_2 also. If any of the A_i, $0 \leq i \leq k$, contains a, then the A_i^* can generate strings which conatins consecutive a's. But it is clear that there is no two consecutive a's in L_2. So for all i, A_i must not contains a's and the language generated by the product $A_0^*a_1A_1^* \cdots a_kA_k^*$ contains only finite number of a's. Hence, U cannot be expressed as the disjoint union of the products $A_0^*a_1A_1^* \cdots a_kA_k^*$.
□ □

5.3 Labelling with Non-commutative Aperiodic Monoids

We show that labelling an undirected graph with either BA_2 or U makes the $\mathcal{A}$-Reach problem NL-hard. Komarath *et al* [15] give a labelling with $(ab)^*$, for all directed acyclic graphs. This immediately gives us a labelling with BA_2. We give a similar labelling with U. We know that any non-commutative aperiodic monoid M not in **DA** is divisible by either U or BA_2. Hence, we have a surjective morphism from a submonoid of M to either U or BA_2. We show that labelling an undirected graph with BA_2 or U makes the $\mathcal{A}$-Reach problem NL-hard. By using the morphism, we can get instances of $\mathcal{A}$-Reach problem over undirected graphs, labelled with M, which are NL-hard.

Theorem 8. *$\mathcal{A}$-Reach for undirected graphs is NL-complete when the graph is labelled with U.*

Proof. We give a labelling from $L = (b^*ab^*bb^*)^*$ (whose syntactic monoid is U) similar to that in [15]. Let $G = (V, E)$ be a directed acyclic graph, with vertices s and t. Without loss of generality, we assume that s is a source (that is, it has only outgoing edges). We create a labelled, undirected graph $G' = (V', E')$ as follows. Each vertex in V is copied to V'. Additionally, for each directed edge (v_i, v_j), we add a vertex m_{ij} to V'. Edges and labels are constructed as follows. If (v_i, v_j) is an edge in G, (v_i, m_{ij}), (m_{ij}, v_j) are edges in G', with (v_i, m_{ij}) being labelled with b, and (m_{ij}, v_j) is labelled with a. That is, we split each edge, labelling the first half with b, and the second half with a. We also add a new vertex t', and add an edge (t, t'), labelled with b. We claim that there is a path from s to t in G if and only if there is a path from s to t' in G', whose yield is in L.

The forward direction is easy to see. Suppose there is a path from s to t in G. Let the path be $s = v_{i_1}, v_{i_2}, \ldots, v_{i_m} = t$. We claim that the path $s = v_{i_1}, m_{i_1i_2}, v_{i_2}, m_{i_2i_3}, v_{i_3}, \ldots, m_{i_{m-1}i_m}, v_{i_m} =$

t, t' exists in G' and the yield of the path is in L. By our construction, each of these edges exist in G'. To see the yield, we notice that since $(v_{i_\ell}, v_{i_{\ell+1}})$ is in E, $(v_{i_\ell}, m_{i_\ell i_{\ell+1}})$ is labelled with b, whereas $(m_{i_\ell i_{\ell+1}}, v_{i_{\ell+1}})$ is labelled with a, for all ℓ. Hence, the yield is $(ba)^m b$ which is in L.

Suppose we do not have a path from s to t in G, but there is a path from s to t' in G' with a yield in L. Since there is no path in G, the path in G' must have taken some edges incorrectly. Let (u, v) be the first incorrect edge taken. That is, suppose $(v_i, v_j) \in E$. The edge taken is either of the form (v_j, m_{ij}) or (m_{ij}, v_i). For the first, the yield up to this point is $(ba)^\ell$, for some ℓ, and the edge is labelled with a. This results in two consecutive a's, which cannot be in the language, and the path in G' cannot have any edge of this form. For the second, we see that since (m_{ij}, a_i) is the first incorrect edge taken, the edge taken before this is (a_i, m_{ij}), and both these edges can be ignored. Thus, if all incorrect edges taken are of the second form, we can create a path from s to t in G, contradicting our initial assumption. □ □

6 Planarizing by Labelling

We now present a reduction from the reachability problem to $\mathcal{A}$-Reach over planar DAGs when $\mathcal{A}$ is the fixed monoid BA_2. The same reduction can be achieved with group labelling when the size of the groups is allowed to be exponentially growing.

6.1 By Labelling with a Fixed Monoid

We give a reduction from Reach to Monoid-PlanarReach and hence it is NL-hard.

Theorem 9. *Let $G = (V, E)$ be a graph. Let $\phi : E \to BA_2$ be a labelling function. Then* Reach *reduces to* Monoid-PlanarReach.

Proof. Let $(G(V, E), s, t)$ be an instance of Reach, we construct an instance of $(G'(V', E'), BA_2, \phi, s, t, \alpha\beta)$ as follows. Without loss of generality, we assume G is a layered graph. We fix an embedding of G by assigning vertices to grid points, and having the edges as straight lines between these points. Moreover, we assume that each edge participates in only one crossing (this can be achieved by splitting an edge wherever necessary). For each non-crossing edge (v_i, v_j) in G, we add a new vertex p_{ij}, and edges (v_i, p_{ij}), and (p_{ij}, v_j), labelling them with α and β respectively. For any two crossing edges (v_i, v_j) and (v_k, v_ℓ) we add three new vertices $m_{ijk\ell}, q_{ijk\ell}$ and $q'_{ijk\ell}$. We split the edge (v_k, v_ℓ) as $(v_k, m_{ijk\ell})$ and $(m_{ijk\ell}, v_\ell)$, with labels α and β respectively. We split the edge (v_i, v_j) into four edges: $(v_i, q_{ijk\ell}), (q_{ijk\ell}, m_{ijk\ell}), (m_{ijk\ell}, q'_{ijk\ell})$ and $(q'_{ijk\ell}, v_j)$, labelling them with α and β alternatively. See figure 1.

We argue the correctness of the reduction. Suppose there is a path from s to t in G, we need to argue that there is a path from s to t in G' whose yield is $\alpha\beta$. We claim something stronger: for any path in the graph G there is a corresponding path between same pair of vertices in G' with the yield $\alpha\beta$. Let path p be a path in G' with non-crossing edges $e_1, e_2, \ldots, e_k$ and crossing edges $f_1, f_2, \ldots, f_k$. By the construction, each non-crossing edge e_i

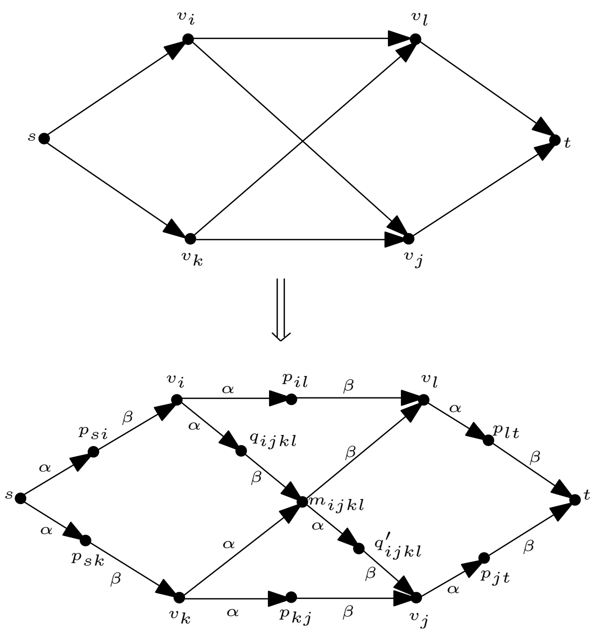

Figure 1: Reach $\leq$ Monoid-PlanarReach

is subdivided into two adjacent edges e_{i_1} and e_{i_2} with label α and β respectively. Similarly each crossing edge f_i is either subdivided into two adjacent edges f_{i_1} and f_{i_2} with label α, β respectively or into four adjacent edges f_{i_1}, f_{i_2}, f_{i_3} and f_{i_4} with label α, β alternatively. Hence the yield of p will be of the form $\alpha\beta\alpha\beta\cdots\alpha\beta$ which multiplies out to $\alpha\beta$.

Suppose there is no path from s to t in G. We want to argue that there is no path from s to t in G' whose yield is $\alpha\beta$. Suppose there is. Since there is no path from s to t in G, this path must have used a newly introduced vertex $m_{ijk\ell}$ for some i, j, k, ℓ. Hence it must include at least one pair of the edges $(q_{ijk\ell}, m_{ijk\ell})$ and $(m_{ijk\ell}, v_\ell)$ (or $(v_k, m_{ijk\ell})$ and $(m_{ijk\ell}, q'_{ijk\ell})$). But the yield corresponds to such paths contains $\beta\beta$ (or $\alpha\alpha$) which multiplies out to 0.

Hence any path from s to t in G corresponds to a path from s to t in G' whose yield is $\alpha\beta$. □ □

6.2 From Bipartititeness of G to Planarity of H

In a close contrast to the results in the previous section, we show that if the labels are coming from BA_2, and in particular from the set $\{\alpha, \beta\}$ and the graph is bipartite, then $\mathsf{NL} = \mathsf{UL}$. That is, if the labelling had preserved bipartiteness of the graph (which we can ensure in the reachability instances by subdividing every edge into two edges by introducing an intermediate vertex), then $\mathsf{NL} = \mathsf{UL}$. We show this by the following theorem.

Theorem 10. *Let $G = (V, E)$ be a planar graph whose underlying undirected graph is bipartite, and labelled with BA_2 with $\phi : E \to \{\alpha, \beta\}$. The $\mathcal{A}$-Reach problem (between any two vertices) in G can be reduced (in logspace) to testing reachability in planar DAGs and* *hence is in* UL.

Proof. We first describe the reduction. Let $G = (V, E)$ be bipartite, such that $V = V_1 \cup V_2$, $V_1 \cap V_2 = \phi$, and $\forall (v_1, v_2) \in E$, either $v_1 \in V_1, v_2 \in V_2$ or $v_1 \in V_2, v_2 \in V_1$.

We first consider a simple case - suppose all outgoing edges of V_1 are labelled with α and all outgoing edges of V_2 are labelled with β. We describe a subgraph H of the product graph G'. Let H be the induced subgraph of $V_H = \{(v, \alpha\beta) | v \in V_1\} \cup \{(v, \alpha) | v \in V_2\}$. We claim that this subgraph H is exactly a copy of G itself, with every vertex $v \in V$ being mapped to (v, g), where g is either α or $\alpha\beta$. Suppose (v_1, v_2) is an edge in E. Without loss of generality assume that $v_1 \in V_1$. This implies that the label on (v_1, v_2) is α. We show that $((v_1, \alpha\beta), (v_2, \alpha))$ is an edge in H. It is easy to see that $(v_1, \alpha\beta)$ and (v_2, α) are vertices in V_H. By the definition of the product graph, we also have this edge in G', and hence, in H. Thus, all edges in G exist in H. To see the other direction, we note that since H is a subgraph of G', all edges in H correspond to some edge in G. Hence, an edge $((v_1, g_1), (v_2, g_2))$ in H implies that we have an edge (v_1, v_2) in G. Thus, we see that H is a copy of G, and all properties of G also exist in H. In particular, H is planar.

Now, consider the case when the edges are labelled arbitrarily with either α or β. To begin with, notice that G is the union of two bipartite, planar graphs G_1 and G_2, one with all outgoing edges of V_1 labelled with α and outgoing edges of V_2 labelled with β, and the other with all outgoing edges from V_1 as β and outgoing edges of V_2 as α. Hence, for each of these graphs, we can construct the subgraph of the product graph, as defined above. Let us call these subgraphs as H_1 and H_2, respectively. Let H be the subgraph of G' induced by the vertices of H_1 and H_2.

Planarity: We show that the graph H defined above is planar. As seen above, H_1 is a copy of G_1 and H_2 is a copy of G_2. Since G is planar, G_1 and G_2 are also planar, implying H_1 and H_2 are planar. The product graph G' of G cannot have edges between vertices of H_1 and H_2, because this is equivalent to multiplying α by itself (or $\alpha\beta$ by β), which yields 0. Hence, this subgraph H of G' is planar.

Correctness: We show the correctness of the reduction now. We argue that there is a path from a vertex s to a vertex $t (\neq s)$ in G, such that the yield is $\alpha\beta$ if and only if there is a path from $(s, \alpha\beta)$ to $(t, \alpha\beta)$ in H, for all s and t in G. Similar arguments can be made for α, β, and $\beta\alpha$ as well.

Suppose there is a path from s to t in G, yielding $\alpha\beta$. Let this path be $s = v_0, v_1, \ldots, v_m = t$. Since all edges in G are labelled with α or β, the yield can be $\alpha\beta$ if and only if the first edge is labelled with α, and subsequent edges alternate between β and α. For i, when it is even, we know that the edge (v_i, v_{i+1}) is labelled with α, and hence, $(v_i, \alpha\beta), (v_{i+1}, \alpha)$ are present in H_1. Similarly, for i, when it is odd, the edge (v_i, v_{i+1}) is labelled with β, and hence, $(v_i, \alpha), (v_{i+1}, \alpha\beta)$ are present in H_1. Hence, the path $(v_0, \alpha\beta), (v_1, \alpha), \ldots, (v_m, \alpha\beta)$ is present in H_1, and hence in H.

For the other direction, suppose we have a path from $(s, \alpha\beta)$ to $(t, \alpha\beta)$ in H. Since H is a subgraph of the product graph G', any path in G' can be traced by a path in G. This implies that there is a path from s to t in G, yielding an element m such that $(\alpha\beta)m = \alpha\beta$. The only possibilities for m in this case are $\alpha\beta$, and 1. Since $s \neq t$, and the edges are labelled only with α and β, the path between s and t cannot evaluate to 1, and hence, the path evaluates to $\alpha\beta$. □ □

6.3 By Labelling with a Finitely Presented Group

Following the quest for more structure in the labelling set, in this section, we now give a reduction from Reach to Group-PlanarReach, when labelled with a group having size exponential in the size of the graph, thus showing that it is NL-hard.

Theorem 11. Group-PlanarReach *is* NL*-hard when the group size is* $\Omega(2^{n^4})$ *where* n *is the size of the graph.*

Proof. The group we will be using is $\mathbb{Z}$. Given an instance $(G(V,E),s,t)$ of Reach, we construct an instance $(G'(V',E'),\mathbb{Z},\phi,s,t,0)$ of Group-PlanarReach as follows. We fix an embedding of G as in the proof of Theorem 9. Without loss of generality, we can assume that G is a layered graph. To construct G', we first order the set of crossing edges, as $e_1, e_2, \dots, e_t$, where t is the number of crossing edges ($t = poly(n)$). For simplicity, we assume that each edge participates in only one crossing. If not, we can always split each edge to ensure this. For crossing edges $e_i = (v_{i_1}, v_{i_2})$ and $e_j = (v_{j_1}, v_{j_2})$, we add a new vertex m_{ij}. We also split the edge e_i into (v_{i_1}, m_{ij}) and (m_{ij}, v_{i_2}), labelling them 2^i and -2^i respectively. Similarly, we split edge e_j as (v_{j_1}, m_{ij}) and (m_{ij}, v_{j_2}), labelling them as 2^j and -2^j respectively. We label each non-crossing edge as 0.

We now argue correctness. Suppose there is a directed path from s to t in G. Let the edges in the path be $e_{i_1}, e_{i_2}, \dots, e_{i_k}$. Now, we have a corresponding path in G'. If there are no crossing edges in this path, all edges exist in G', and are labelled with 0, and hence, our path evaluates to 0. Else, we have some crossing edges. Suppose e_{i_ℓ} is one such edge. Now, the corresponding split edges are labelled with 2^{i_ℓ} and -2^{i_ℓ}, and hence, the path still evaluates to 0. Thus, we have a path from s to t in G', which evaluates to 0.

Assume that there is no directed path in G from s to t. We need to argue that there is no path in G' from s to t evaluating to 0. Suppose there is. We say that a label 2^i is *matched* if we use the split edges labelled with 2^i and -2^i in the path, and *unmatched* otherwise. Since the corresponding path does not exist in G, we must have used at least one newly created vertex m_{ij}, and included either edges (v_{i_1}, m_{ij}) and (m_{ij}, v_{j_2}) or (v_{j_1}, m_{ij}) and (m_{ij}, v_{i_2}). Hence, we have at least one unmatched label. Let 2^i be the largest (absolute) value of the unmatched labels in the path. Without loss of generality, assume the edge labelled with 2^i is in the path. Since the path evaluates to 0, the sum of all negative unmatched labels must be -2^i, or lesser. But, this sum can be at least $-2^i + 1$, since 2^i is the largest unmatched value. Hence, the path can never evaluate to 0, and no such path exists in G'. Since there are $O(n^4)$ crossings in the graph, the evaluation of any path cannot be more than $O(2^{n^4})$. Hence, Reach reduces to GroupPlanarReach, when the size of the group is $\Omega(2^{n^4})$. □ □

7 Discussion & Conclusion

In this paper, we studied the variant of reachability problem on labelled graphs when the labels come from a set having algebraic properties. We showed a complete classification of reachability problem on labelled graphs where the labels are from an aperiodic monoid of fixed size. We also showed logspace upper bounds for the case when the labels are from a group, even when the size of the group is polynomial in the size of the graph.

On the monoid labeling side, an important question is whether the classification can be extended to non-aperiodic monoids as well. More specifically, is it true that over commutative monoids the reachability problem on labelled graphs be solved in L?

On the complexity front, observing that reachability testing on bounded treewidth graph can be solved in L[9], a natural question is whether we can construct a monoid M such that for the product graph G', there is an absolute constant $c > 0$, for all graphs G $tw(G') \leq tw(G)/c$. Repeating this logarithmically many times, we can get to a product graph whose treewidth is bounded by a constant. Notice that this would immediately imply $\mathsf{NL} = \mathsf{L}$. In this connection it important to construct labellings where the original graph does not appear as a subgraph in the product graph. We have given one characterization for the same when the labelling is from a group (See Proposition 5).

References

[1] Eric Allender. Reachability problems: An update. In *Third Conference on Computability in Europe, CiE 2007*, pages 25–27, 2007.

[2] S. Arora and B. Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[3] David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of nc^1. *J. ACM*, 35(4):941–952, 1988.

[4] Martin Beaudry, François Lemieux, and Denis Thérien. Finite loops recognize exactly the regular open languages. In *24th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 110–120, 1997.

[5] François Bédard, François Lemieux, and Pierre McKenzie. Extensions to barrington's m-program model. In *Proceedings: Fifth Annual Structure in Complexity Theory Conference, Universitat Politècnica de Catalunya, Barcelona, Spain, July 8-11, 1990*, pages 200–209, 1990.

[6] Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. Directed planar reachability is in unambiguous log-space. *ACM Trans. Comp. Theory*, 1(1), 2009.

[7] Hervé Caussinus and François Lemieux. The complexity of computing over quasigroups. In *Foundations of Software Technology and Theoretical Computer Science, 14th Conference, December 15-17, 1994, Proceedings*, pages 36–47, 1994.

[8] Ashok K. Chandra, Steven Fortune, and Richard J. Lipton. Unbounded fan-in circuits and associative functions. *J. Comput. Syst. Sci.*, 30(2):222–234, 1985.

[9] Bireswar Das, Samir Datta, and Prajakta Nimbhorkar. Log-space algorithms for paths and matchings in k-trees. In *27th STACS*, pages 215–226, 2010.

[10] Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of bodlaender and courcelle. In *Foundations of Computer Science (FOCS)*, pages 143–152, 2010.

[11] Jim Geelen and Bert Gerards. Excluding a group-labelled graph. *J. Comb. Theory Ser. B*, 99(1):247–253, January 2009.

[12] Susan Horwitz, Thomas W. Reps, and David Binkley. Interprocedural slicing using dependence graphs. *ACM Trans. on Prog.Lang. and Systems*, 12(1):26–60, 1990.

[13] Tony Chi Thong Huynh. The linkage problem for group-labelled graphs. In *Ph.D. Thesis, University of Waterloo*, 2009.

[14] Yasushi Kawase, Yusuke Kobayashi, and Yutaro Yamaguchi. Finding a path in group-labeled graphs with two labels forbidden. In *42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 797–809, 2015.

[15] Balagopal Komarath, Jayalal Sarma, and K. S. Sunil. On the complexity of L-reachability. *Fundam. Inform.*, 145(4):471–483, 2016.

[16] Jean-Eric Pin, H. Straubing, and D. Thérien. Locally trivial categories and unambiguous concatenation. *Jl. of Pure and Applied Algebra*, 52(3):297 – 311, 1988.

[17] Jean-François Raymond, Pascal Tesson, and Denis Thérien. An algebraic approach to communication complexity. In *25th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 29–40, 1998.

[18] Omer Reingold. Undirected connectivity in log-space. *Jl. of the ACM*, 55(4), 2008.

[19] Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks on regular digraphs and the RL vs. L problem. In *Proceedings of STOC*, pages 457–466, 2006.

[20] Thomas W. Reps. On the sequential nature of interprocedural program-analysis problems. *Acta Informatica*, 33(8):739–757, 1996.

[21] Thomas W. Reps. Program analysis via graph reachability. *Information & Software Technology*, 40(11-12):701–726, 1998.

[22] Pascal Tesson. An algebraic approach to communication complexity. In *Masters Thesis, McGill University, Montreal*, 1998.

[23] Mihalis Yannakakis. Graph-theoretic methods in database theory. In *Proceedings of the 9th ACM PODS*, pages 230–242, 1990.

A Appendix

A.1 Connectivity of Eulerian graphs can be checked in L

Reingold *et al* [19] observe, for an Eulerian graph, replacing directed edges by undirected edges does not alter connectivity. Hence, testing reachability in graphs where each component is Eulerian, can be done in L using the Reingold's algorithm [18] for testing reachability in undirected graphs. For completeness, we provide a proof of this observation by showing that, for a graph with Eulerian components, each weakly connected component is strongly connected, implying that if a path from s to t exists in the undirected setting, we also have a path from s to t in the directed setting.

Lemma 1. *For Eulerian graphs, if vertices s and t are weakly connected then s and t are strongly connected.*

Proof. Let $G = (V, E)$ be an Eulerian graph. Suppose vertices s and t are weakly connected.

For an Eulerian graph, for each vertex v, we know that the indegree of v is equal to the outdegree of v. We denote this by $In(v)$, $Out(v)$. We extend the notation of indegree and outdegree to sets of vertices: for $V' \subseteq V$, $In(V') = |\{e = (v_1, v_2) | v_1 \in V', v_2 \in V \backslash V'\}|$ and $Out(V') = |\{e = (v_1, v_2) | v_1 \in V \backslash V', v_2 \in V'\}|$. We first prove that for any subset $V' \subseteq V$ in an Eulerian graph $In(V') = Out(V')$, using induction on the size of V'. This trivially holds when $|V'| = 1$, since the indegree and outdegree of any vertex are equal. Suppose this property holds for all subsets $|V'| < k$. We show that it holds for any subsets V', such that $|V'| = k$. Choose any vertex $v \in V'$. We know that

$$In(V') = In(V' \backslash \{v\}) + In(v) - |\{(v, v_1) | v_1 \in V'\}| - |\{(v_1, v) | v_1 \in V'\}|$$

$$Out(V') = Out(V' \backslash \{v\}) + Out(v) - |\{(v, v_1) | v_1 \in V'\}| - |\{(v_1, v) | v_1 \in V'\}|$$

Since $|V' \backslash \{v\}| = k - 1$, from our induction hypothesis, $In(V' \backslash \{v\}) = Out(V' \backslash \{v\})$. This immediately gives us $In(V') = Out(V')$, when $|V'| = k$. By induction, for any subset $V' \subseteq V$ in an Eulerian graph, $In(V') = Out(V')$.

We describe an algorithm to find a path between s and t. We partition the vertex set of the graph, and at each step, increment one of the partitions.

Initially, we start with $V_1 = \{s\}$, and $V_2 = V \backslash V_1$. We increase the size of V_1 as follows. For all vertices $u \in V_1$, if (u, v) is an edge, and $v \notin V_1$, we add v to V_1, and remove v from V_2. We notice that if no such v exists, then, there is no cut edge between V_1 and V_2 - since the number of edges crossing the cut in both directions is always equal.

We claim that when the algorithm terminates, $t \in V_1$. Suppose not. Then $t \in V_2 \neq \phi$. However this implies that there exists a partition of the graph into V_1 and V_2, such that these subgraphs are disjoint. Hence, s and t cannot be weakly connected, and this contradicts our given statement. Hence, s and t are strongly connected if s and t are weakly connected. □ □