International Journal of Algebra and Computation
Vol. 28, No. 5 (2018) 759–790

DOI: 10.1142/S0218196718500352

World Scientific
www.worldscientific.com

Evolution of algebraic terms 3:
Term continuity and beam algorithms

David M. Clark*
Mathematics Department, SUNY New Paltz
New Paltz, NY 12561, USA
clarkd@newpaltz.edu

Lee Spector
School of Cognitive Science, Hampshire College
Amherst, MA 01002, USA
lspector@hampshire.edu

Received 5 August 2017
Accepted 26 April 2018
Published 5 June 2018

Communicated by K. Kearnes

The first paper in this series introduced the notion of term to term operation continuity for finite groupoids, and proved that two testable conditions on a finite groupoid imply that it is term continuous (TC). The second presented an evolution inspired algorithm for finding terms for operations, and gave experimental evidence that, in general, it was successful exactly when the groupoid was both idemprimal and TC. In this paper, we describe a new class of algorithms for finding terms which brings these results together. Theorems about idemprimality and term continuity show how each of these two properties impact our algorithms. They lead to a final explanation for the success of our algorithms when the groupoid is both idemprimal and TC.

Keywords: Evolutionary computation; term operation; idemprimality; term continuity; randomizing algorithms.

Mathematics Subject Classification 2010: 08A40, 08A70

1. Introduction

This is the final installment in our three part series introducing evolutionary methods for finding terms for term operations on a finite groupoid. This is equivalent to the problem of designing a switching circuit from a performance specification with a single finite groupoid defining its gates. We will prove a theorem showing how *term continuity*, characterized in EAT 1 [3], can, in combination with *idemprimality*,

---

*Corresponding author.

drive the computational success of a class of evolution inspired algorithms, algorithms similar to the Deep Drilling Algorithm presented in EAT 2 [5]. While this theorem gives us a strong connection between computation and algebra, we will see that it simultaneously adds to a growing list of interesting and still unresolved questions and conjectures raised in [3] and [5] about this subject.

The underlying problem addressed in this series comes from the theory of primal algebras. Recall that a non-trivial finite algebra $\mathbf{G}$ is **primal** if every operation on its underlying set is the term operation $f(\vec{x})^{\mathbf{G}}$ of some term $f(\vec{x})$. Rousseau proved a corollary [12] to Rosenberg's Theorem [11] giving a very simple way to test for primality if the algebra has exactly one fundamental operation that is at least binary. It says that the algebra is primal if and only if it has no proper subalgebras and no non-trivial congruences or automorphisms. This is one of those strange and wonderful results of mathematics that guarantees the existence of something without giving any indication as to how to find it.

As an illustration of this idea, an easy application of Rousseau's theorem shows that the groupoid $\mathbf{D}_1$ in Fig. 1 is primal. This implies, for example, that there is a term representing the ternary discriminator operation $d\colon D_1^3 \to D_1$ on $\mathbf{D}_1$. So we can ask, how difficult is it to *find* a discriminator term for $\mathbf{D}_1$? The two previously known methods to do this are examined by Clark, Keijzer and Spector [5], the first primarily computational and the second primarily algebraic.

- The first is a *random search* in which we enumerate all ternary terms in increasing size order, and check each one to see if it is a discriminator term. This method is guaranteed to give us a minimal solution, but it requires searching $4^{4^3} \approx 10^{38}$ ternary operations on D_1. In [5, Fig. 3] it is estimated that this would take a fast computer with unlimited memory, checking a million terms per second, an estimated expected time of 10^{25} years.
- The second is the algebraic test of Clark, Davey, Pitkethly and Rifqui [4]. It draws on the UACalc program of Freese *et al.* [6] to efficiently produce a recursive description of the required term. But the recursion tends to produce long terms. In this case, [5, Theorem 5] returns an estimated number of variable occurrences in the resulting discriminator term of 10^{45}.

Unfortunately these two methods leave us with the unsettling choice of getting a compact term in an unfeasibly long time or quickly getting a compact recursive description of an unfeasibly long term.

$\mathbf{D}_1$

$*$	0	1	2	3
0	1	3	1	2
1	1	2	2	3
2	2	0	0	3
3	2	2	1	2

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

Fig. 1. A primal groupoid $\mathbf{D}_1 := \langle\{0,1,2,3\}, *\rangle$ and its ternary discriminator operation.

The goal of this series is to investigate search methods to find human scale solutions to problems like this in human scale time. All of our algorithms terminate if and only if they find a correct solution. They all use randomizing methods that make random choices along the way. Thus, it is unlikely that they will *always* return a correct solution, as there can be particular sequences of choices that lead to failure. Our objective is to establish conditions under which they will eventually produce a correct solution with probability one.

In order to state the general problem precisely, we need to lay out a few concepts. A **k-ary array** over a finite algebra $\mathbf{G}$ is a function $A\colon G^k \to 2^G$, that is, a G^k-indexed sequence of subsets of G. A **solution** to a k-ary array A over $\mathbf{G}$ is a k-ary term $u(\vec{x})$ such that $u(\vec{a}) \in A(\vec{a})$ for all $\vec{a} \in G^k$.

Term Generation Problem. *The Term Generation Problem (TGP) for a finite algebra* $\mathbf{G}$ *is the problem of finding a solution to a given array over* $\mathbf{G}$ *under the assumption that it does have a solution.*

For example, finding a term to represent a particular k-ary operation $f\colon G^k \to G$ is equivalent to solving the array $\langle f\rangle$ where $\langle f\rangle(\vec{d}) := \{f(\vec{d})\}$ for each $\vec{d} \in G^k$. The problem of finding partially specified terms, like Mal'cev terms, Pixley terms and majority terms, can be viewed as solving arrays over $\mathbf{G}$ which have value G on the triples that are not relevant to those respective terms. The methods used in [5] and in this paper to find terms for operations inherently require solving the broader problem of finding solutions to arrays.

Evolutionary methods to solve the TGP began with the proof-of-concept paper [13]. There the authors used a range of conventional genetic programming techniques to find terms on several sample 3-element primal groupoids in search spaces of ternary term operations of sizes up to $3^{3^3} \approx 10^{13}$. Success on that effort was sufficiently compelling to win first place in the GECCO 2008 Human Competitive competition and launch the current Evolution of Algebraic Terms 1, 2, 3 series.

* * * * *

In EAT 1 [3] Clark studied a principle that is fundamental not only to biological evolution and evolutionary algorithms, but to many search algorithms that are only remotely connected to evolution. The targets of these searches might be organisms, term operations or maybe windmill blades. In each case they typically have two forms: the *genotype*, a symbolic design (a DNA strand, a term or a set of blueprints), and a *phenotype*, the object itself (an organism, a term operation or a windmill blade). Variations in the genotypes are reflected in corresponding variations in the phenotypes, making their population more diverse and more likely to include improvements. But it is important that changes in phenotypes be small in order to preserve previously acquired benefits. The process then depends on knowing that small changes in the genotypes will result in correspondingly small changes in the phenotypes. This condition was phrased in [3] as a condition on the particular search problem.

Continuity Condition. *The map taking genotypes to phenotypes is continuous in the sense that small changes (mutations) in genotypes normally lead to small changes in their phenotypes.*

The initial task of [3] was to give a precise definition of the Continuity Condition in the context of the TGP, with terms as genotypes and term operations as phenotypes. We will review the relevant details in Sec. 5. The **height** of a term is the height of its term tree, with the height of a variable being one. As in [3] and [5], we draw term trees with the root at the bottom so that a subterm *deeper* in the term corresponds to a node *higher* in the term tree. For our purposes, here we initially phrase the continuity condition on a groupoid as follows.

Continuity Condition for Groupoids. *For a positive integer k, a finite groupoid* **G** *is k-**term continuous** (TC) if, in general, changes to a subterm of a k-ary term alter fewer values of the resulting term operation when the subterm is deeper in the term.*

A precise formulation of this asymptotic condition is given in [3] and reviewed here in Sec. 6, Theorem 14. However, that formulation does not offer any indication as to how we might in practice test a given groupoid for term continuity. In [3] two testable conditions on a finite groupoid are described that together imply term continuity. Very briefly, let **G** be a finite groupoid, k and H be positive integers, $\vec{d} \in G^k$ and $a \in G$. Let $\beta_{\vec{d},a}(H)$ be the probability that a k-ary term of height at most H will take value a at $\vec{d}$. Then **G** is **asymptotically k-complete** if the sequence $\beta_{\vec{d},a}$ is eventually bounded away from zero whenever a is in the subgroupoid of **G** generated by the range of $\vec{d}$. A non-empty, irreflexive, symmetric binary relation σ on G is a **separating relation** if it is preserved by left and right multiplication by elements of **G**. We say **G** has **no separating relations** (NSRs) if no subgroupoid of **G** has a separating relation.

Spreadsheet calculations of sample sequences $\beta_{\vec{d},a}$ are shown in [3] and [5] to normally give compelling evidence (but no proof) that a groupoid is or is not asymptotically k-complete. A simple paper and pencil test on **G** will either exhibit a separating relation or prove that none exist, thereby deciding if it has NSRs. The main result of [3] is the following theorem.

Continuity Theorem. *For a positive integer $k > 1$, if a finite groupoid is asymptotically k-complete and has NSRs, then it is k-TC.*

* * * * *

EAT 2 [5] examined the two prior methods to solve the TGP referenced above and then presented the Deep Drilling Algorithm that solves the TGP by producing human scale terms in human scale time for groupoids with exceptionally large search spaces. It was here that idemprimality arose. An element e of an algebra is an **idempotent** if $\{e\}$ forms a subalgebra. A primal algebra cannot have an idempotent, as an operation not preserving the idempotent would not be a term operation. A non-trivial finite algebra is **idemprimal** if every operation that *does* preserve its idempotents is a term operation. A well known result says that, in a specific sense, almost all finite algebras are idemprimal.

Theorem (Murskiĭ [8], 1975, or [1] Chap. 6). *Let Op be a finite type with at least one operation that is at least binary. Then the proportion of algebras* $\mathbf{G} := \langle\{0, 1, \ldots, n-1\}, Op^{\mathbf{G}}\rangle$ *that are idemprimal approaches one as* n *approaches infinity.*

Still open conjectures from [3] predict that almost all finite groupoids are asymptotically k-complete and have NSRs as well, and are therefore k-TC. EAT 2 went on to document randomized trials which experimentally demonstrated a strong positive correlation between success of the Deep Drilling Algorithm and the groupoid being both idemprimal and k-TC. However, it fell short of giving any explanation as to how term continuity contributes to the success of this algorithm. It ends with a range of still unsolved problems that arose from this work. The first of those problems concerns term continuity.

Problem 1. Prove a theorem that explains how term continuity (or the combination of asymptotic completeness and NSRs) helps to ensure the success of the Deep Drilling Algorithm.

* * * * *

This paper, EAT 3, ties together term continuity of EAT 1 [3] with a new class of algorithms for solving the TGP, the Beam Enumeration Algorithms (BEAs), that are rather comparable to the Deep Drilling Algorithm of EAT 2 [5]. The role of idemprimality in these algorithms is given by Theorem 4 (a slight extension of [5, Theorem 11]), Theorems 6 and 9. The bulk of this paper is devoted to an investigation and explanation of the role of term continuity, culminating in Theorem 21. We end by presenting a particular BEA and illustrating its success with groupoids that are both idemprimal and k-TC.

2. Valid Partial Terms and Idemprimality

In this section, we will see how the TGP can be formulated in such a way that primality (and, more generally, idemprimality) allows us to make a significant step toward a solution. Let k be a fixed positive integer. By a **term** we mean a term in the variables $\vec{x} := (x_0, x_1, \ldots, x_{k-1})$ with some fixed finite set of operation symbols. We will use a new variable $\lozenge$ (lozenge) and define a **partial k-ary term** to be a term $f(\vec{x}, \lozenge)$ in variables $x_0, x_1, \ldots, x_{k-1}, \lozenge$ in which $\lozenge$ occurs exactly once. A **completion** of a partial term $f(\vec{x}, \lozenge)$ is a term $f(\vec{x}, u(\vec{x}))$ obtained by substituting a k-ary term $u(\vec{x})$ for $\lozenge$ in $f(\vec{x}, \lozenge)$. The notion of a completion suggests a two-step method to solve an array A.

Step 1. *Generate partial terms* $f(\vec{x}, \Diamond)$ *that can be completed to solutions to* A.

Step 2. *For some partial term* $f(\vec{x}, \Diamond)$ *generated in Step* 1, *find a term* $u(\vec{x})$ *such that* $f(\vec{x}, u(\vec{x}))$ *is a solution to* A.

Clearly this method depends on finding a way to tell if $f(\vec{x}, \Diamond)$ can be completed to a solution to A *prior* to finding a term $u(\vec{x})$ that will achieve that completion. We will give a method of doing Step 1 in this section, and then devote the rest of the paper to Step 2.

We begin Step 1 by looking at a means to eliminate a generally vast proportion of the partial terms which have no completion to a solution to A. Given a k-ary array A over a finite algebra $\mathbf{G}$, we say that a partial k-ary term $f(\vec{x}, \Diamond)$ is **valid with respect to** A if,

$$\text{for all } \vec{d} \in G^k, \quad \text{there is a } b \in G \text{ such that } f(\vec{d}, b) \in A(\vec{d}).$$

Alternatively, $f(\vec{x}, \Diamond)$ is valid with respect to A provided that there is a k-ary operation $h : G^k \to G$ such that $f(\vec{d}, h(\vec{d})) \in A(\vec{d})$ for all $\vec{d} \in G^k$. Thus, $f(\vec{x}, \Diamond)$ has a completion that is a solution to A if and only if $f(\vec{x}, \Diamond)$ is valid with respect to A and there is a k-ary operation h witnessing its validity that is a term operation of $\mathbf{G}$. Given $\mathbf{G}$ and A, it is computationally easy to test partial terms for validity with respect to A. A partial term which fails that test could not possibly have a completion that is a solution to A. If $\mathbf{G}$ is idemprimal, then these two notions are equivalent.

Theorem 1. *Let* $\mathbf{G}$ *be an idemprimal algebra, let* A *be a* k*-ary array over* $\mathbf{G}$ *that has a solution, and let* $f(\vec{x}, \Diamond)$ *be a partial term. Then* $f(\vec{x}, \Diamond)$ *has a completion that is a solution to* A *if and only if it is valid with respect to* A.

Proof. Assume that $\mathbf{G}$ is idemprimal and $f(\vec{x}, \Diamond)$ is valid with respect to A. Define $g : G^k \to G$ as follows. Let $\vec{d} \in G^k$. If $\vec{d} = (e, e, \ldots, e)$ where e is an idempotent, let $g(e, e, \ldots, e) = e$. Otherwise, let $g(\vec{d})$ be any element $b \in G$ for which $f(\vec{d}, b) \in A(\vec{d})$. Since g preserves idempotents, there is a term $t(\vec{x})$ such that $t^{\mathbf{G}} = g$. If $u(\vec{x})$ is a solution to A, then we have, for each idempotent e of $\mathbf{G}$,

$$f(e, e, \ldots, e, t(e, e, \ldots, e)) = f(e, e, \ldots, e, e) = e = u(e, e, \ldots, e) \in A(e, e, \ldots, e).$$

Thus, $f(\vec{d}, t(\vec{d})) \in A(\vec{d})$ for all $\vec{d} \in G^k$, and therefore $f(\vec{x}, t(\vec{x}))$ is a solution to A. □

If $\mathbf{G}$ is idemprimal, we can do Step 1 by generating partial terms in any systematic way, and then testing each one for validity with respect to A. Theorem 1 guarantees that the valid partial terms we find will have completions to solutions to A without giving any information about how to find those completions.

Recall that a non-trivial finite algebra is **semiprimal** if every operation that preserves all subalgebras is a term operation. It is natural to ask if Theorem 1 extends to semiprimal algebras. It turns out that it does not. Consider the 3-element

Berman and Burris Catalog #**571**

$*$	0	1	2
0	0	0	1
1	0	2	1
2	2	1	1

Fig. 2. A semiprimal groupoid.

groupoid #**571**, listed as being semiprimal in the Berman and Burris Catalog [2] of 3-element groupoids (Fig. 2).

Example 2. Let $k = 1$ and let T be the array over the semiprimal groupoid #**571** from [2] taking 0 to $\{0\}$ and both 1 and 2 to $\{2\}$. Then $x^2\Diamond$ is valid with respect to T but has no completion to a solution to T.

Proof. We first check that $x^2\Diamond$ is valid with respect to T:

$$0^2 * 0 = 0 \in \{0\}, \quad 1^2 * 0 = 2 \in \{2\}, \quad 2^2 * 1 = 2 \in \{2\}.$$

Suppose that $x^2 t(x)$ were a solution to T. Then we would have $2 = 1^2 * t(1) = 2 * t(1)$ so $t(1)$ would have to be 0. But $0 \notin \{1, 2\} = \mathrm{sg}\{1\}$. $\square$

3. BEAs for the TGP

Given an idemprimal algebra **G** and an array A that has a solution, Theorem 1 says that we can do Step 1 by generating partial terms and testing them for validity. In order to do this, we first follow [3] and [5] and, from this point on, take **G** to be a groupoid, that is, an algebra with a single (often unwritten) binary operation.

We also need to fix a method to generate partial terms. By a **Valid Term Generator** (VTG) we mean a randomizing algorithm that takes as input a finite groupoid **G** and a k-ary array A over **G** that has a solution, and then generates a sequence of k-ary terms in such a way that the probability that every k-ary term operation has been represented approaches one. For each term $u(\vec{x})$ generated it checks the two partial groupoid terms $u(\vec{x})\Diamond$ and $\Diamond u(\vec{x})$ for validity with respect to A, and outputs all of those found to be valid.

In this section, we will describe a means of using a VTG to carry out Step 2. Given a partial term $f(\vec{x}, \Diamond)$ that is valid with respect to a k-ary array A over **G** from Step 1, we must find a term $u(\vec{x})$ so that $f(\vec{x}, u(\vec{x}))$ is a solution to A. To do this we define the **validity array** for $f(\vec{x}, \Diamond)$ over A, denoted by $[f(\vec{x}, \Diamond) : A]$, as

$$[f(\vec{x}, \Diamond) : A](\vec{d}) := \{b \in G \mid f(\vec{d}, b) \in A(\vec{d})\}$$

for all $\vec{d} \in G^k$. The array $[f(\vec{x}, \Diamond) : A]$ is the array of witnesses that $f(\vec{x}, \Diamond)$ is valid with respect to A. Note that $f(\vec{x}, \Diamond)$ is in fact valid with respect to A if and only if there are enough witnesses that each of the component sets of $[f(\vec{x}, \Diamond) : A]$ is nonempty. The following lemma shows that Step 2 is itself an instance of the TGP.

Lemma 3. *Let A be a k-ary array over a finite groupoid* $\mathbf{G}$, *let $f(\vec{x}, \Diamond)$ be a partial term and let $u(\vec{x})$ be a term. Then $f(\vec{x}, u(\vec{x}))$ is a solution to A if and only if $u(\vec{x})$ is a solution to the validity array $[f(\vec{x}, \Diamond) : A]$ of $f(\vec{x}, \Diamond)$ over A.*

Proof. We have that $f(\vec{x}, u(\vec{x}))$ is a solution to A

$$\begin{aligned} &\text{if and only if } f(\vec{d}, u(\vec{d})) \in A(\vec{d}) \quad \text{for all } \vec{d} \in G^k \\ &\text{if and only if } u(\vec{x}) \quad \text{is a solution to } [f(\vec{x}, \Diamond) : A]. \end{aligned}$$ □

This means that we can solve a target array A if we can find a partial term $f(\vec{x}, \Diamond)$ that is valid with respect to A and then solve the validity array $B := [f(\vec{x}, \Diamond) : A]$. In order to solve B, we could recursively apply the same procedure, finding a partial term $g(\vec{x}, \Diamond)$ that is valid with respect to B and then solving its validity array. This process could lead to a hierarchy of partial terms and associated validity arrays. If we could find a solution to any one of those validity arrays, we could back propagate solutions down to a solution to A. This process is illustrated as a tree $\mathcal{T}$ in Fig. 3 using a hypothetical example of an unspecified a target array T over an unspecified finite groupoid $\mathbf{G}$. We assume we have a particular VTG to produce valid partial terms and a finite set M of terms that will serve as guesses for solutions.

At Level 1 the root of $\mathcal{T}$ is the partial term $\Diamond$, where we take its parent array to be T. The root is labeled with $\Diamond$ and its validity array over the parent array T, which is also T. We begin by checking to see if the validity array T has a solution in M. If not, applying the VTG to $\mathbf{G}$ and T might give us the three partial terms $((x_1x_0)x_2)\Diamond$, $x_0\Diamond$ and $\Diamond x_1^2$, all valid with respect to T, at Level 2. We then calculate their respective validity arrays A, B and C over T and check to see if any of them have a solution in M. If not, we again invoke the VTG to produce partial terms at

$J := [(x_4x_1)\Diamond : H]$
$K := [\Diamond x_3 : I]$
$L := [(x_0x_3)\Diamond : I]$
$G := [(x_2x_4)\Diamond : E]$
$H := [\Diamond x_2^2 : E]$
$I := [x_2\Diamond : F]$
$D := [\Diamond x_4 : A]$
$E := [\Diamond x_3 : B]$
$F := [x_3\Diamond : C]$
$A := [((x_1x_0)x_2)\Diamond : T]$
$B := [x_0\Diamond : T]$
$C := [\Diamond x_1^2 : T]$

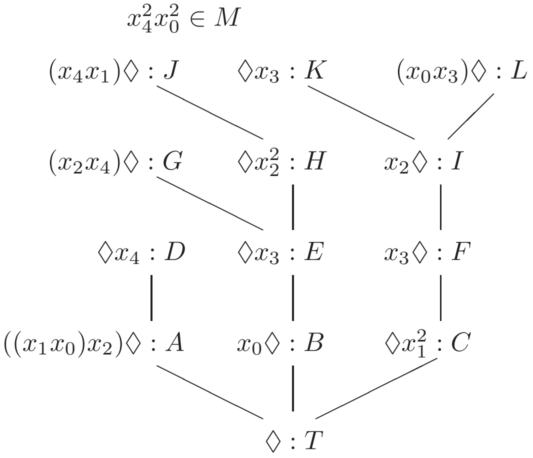

Fig. 3. Tree $\mathcal{T}$ of partial terms and their validity arrays, each valid with respect to its parent array.

Level 3 that are valid with respect to those arrays. Figure 3 illustrates the tree $\mathcal{T}$ with each partial term valid with respect to its parent array.

Suppose that at some point we were to find in M a solution to one of these validity arrays. For example, suppose $x_4^2x_0^2 \in M$ were a solution to J. Since $(x_4x_1)\Diamond$ is valid with respect to H and J is its validity array over H, Lemma 3 says that $(x_4x_1)(x_4^2x_0^2)$ would be a solution to H. Repeatedly applying Lemma 3, we see that

$((x_4x_1)(x_4^2x_0^2))x_2^2$ would be a solution to E,

$(((x_4x_1)(x_4^2x_0^2))x_2^2)x_3$ would be a solution to B, and now

$x_0((((x_4x_1)(x_4^2x_0^2))x_2^2)x_3)$ would be a solution to T!

We will now formulate this strategy as a general schema for solving the TGP that omits a number of details. A BEA is any algorithm obtained by filling in those details. Our final theorems will apply to all BEAs. Our schema will be broad enough to allow for a range of particular BEAs. At the same time, it will be restrictive enough to allow us to prove theorems saying that all BEAs will successfully find terms for most finite groupoids. Two BEAs will differ only in how efficiently they produce solutions. Among the different particular BEAs the more efficient ones will be able to produce solutions to more difficult instances of the TGP in an acceptable time frame.

These algorithms, as illustrated by the example above, are reminiscent of a commonly used method to do tree searches. Full tree searches are often unsuccessful because of the amount of time and memory space they consume. A **beam search** is a search of a tree for a particular target that starts at the root and works upward, level by level, with a fixed finite bound b on the number of nodes it will examine at each level above the root. (See, e.g., [9].) Typically it attempts to select those nodes in a way that is progressively more likely to lead to a solution. As a result, the nodes examined form a rooted tree of some finite width b, called a **beam tree**. Our algorithms for the TGP are almost beam algorithms, but the tree $\mathcal{T}$ being searched is not specified in advance. In Sec. 5, we will see the advantages of a method to recast them as beam algorithms searching a single beam tree $\mathcal{T}'$. We therefore define a BEA as any algorithm for solving the TGP obtained from the following general schema.

Schema for BEAs

Choose positive integers b and h that meet the restrictions we describe below, a finite test set M of terms that includes the single variables, a VTG constrained to produce only partial terms of height at most h and a strategy for using the VTG. Given a k-ary target array T over a finite groupoid $\mathbf{G}$ that has a solution construct a beam tree $\mathcal{T}$ in successive levels, each level above the root consisting of b partial terms and their associated arrays, as follows.

(1) The root, Level 1, will consist of the one partial term $\Diamond$ with associated array T (the validity array of $\Diamond$ over T).

(2) Once Level H of $\mathcal{T}$ has been constructed, check each Level H array to see if it has a solution in M.
   (a) If one has a solution in M, stop building $\mathcal{T}$. Substitute that solution into that Level H partial term, and back propagate solutions down to the root. Output the term generated at the root and terminate.
   (b) If no solution is found, move on to construct Level $H+1$.

(3) Use the VTG to search for a sequence of b partial terms for Level $H+1$, each valid with respect to the validity array of some partial term at Level H. Calculate the validity array of each of these partial terms over its parent array.

The example of Fig. 3 illustrates a hypothetical tree of partial terms generated by a BEA with $k = 5$ variables, $h = 3$ as the height limit of terms generated by the VTG, and $b = 3$ partial terms at each level. When the solution $x_4^2x_0^2$ to $J = [(x_4x_1)\Diamond : H]$ is found in M, the BEA invokes step (2a) to produce the solution $(x_4x_1)(x_4^2x_0^2)$ to H and back propagates down the tree $\mathcal{T}$ to produce the solution $x_0((((x_4x_1)(x_4^2x_0^2))x_2^2)x_3)$ to T.

A specific BEA is constructed by choosing values of the five parameters that have been left unspecified in this schema. We briefly list some options and constraints for making these choices.

(1) **Value of *h*.** The parameter h must be large enough to allow the VTG to produce the required valid partial terms when they do exist. This can always be achieved by choosing h large enough to include a term representing every k-ary term operation. Among values of h that are sufficiently large, smaller choices will make term generation faster. Our experience is that small values of h are generally adequate.

(2) **Value of *b*.** A wider beam offers more opportunity to find a solution at each level, but requires more time be spent at each level unless the valid partial terms are being processed in parallel. If they are not, a narrower beam will grow vertically more quickly. Note that the sequence of b partial terms may include duplicates.

(3) **Choice of Valid Term Generator.** We can suggest four different types of VTGs.

- Deterministically enumerate all terms from small to large, starting with the variables. This method results in more compact solutions because it delivers the first successful term found, which is of minimal length. However, consider any path up the tree that is produced. There are only finitely many combinations of an array A, a term operation for $u(\vec{x})$, and a choice of $\Diamond u(\vec{x})$ or $u(\vec{x})\Diamond$ valid with respect to A. If such a combination appears as a descendant of itself, the interval between the two may be repeated forever by a deterministic enumeration and the program will never terminate. This becomes more of a hazard when the groupoid is small and b is small.

- At the other extreme, we can use the Grow algorithm of Kosa [7] to generate random terms. Grow constructs a binary tree from the root up. Each new node is made a leaf or branch node with equal probability until no branch nodes remain, and then it randomly assigns variables to the leaves. It is easy to see that each term has a positive probability of being produced that drops quickly with the size of the term. In [7] it is shown that Grow will eventually terminate with probability one. This method appears to solve the repetition problem, but at the cost of producing solutions of inconsistent and often large size.
- As an intermediate method we can use the deterministic method but randomize the order of some fixed initial segment of the terms to get a mix of the advantages and liabilities of the deterministic and Grow methods.
- Many techniques are available to design evolutionary algorithms to produce valid terms directly. A natural measure of validity fitness of a partial term would be the number of k-tuples at which the condition for validity is satisfied. These techniques tend to be more time consuming than those above, but can be productive when the TGP becomes more difficult.

(4) **Use of VTG.** There are different strategies for using the VTG. For example, it could produce at each level some fixed number greater than b of valid partial terms, and then in some way prune them down to b valid partial terms based on some measure of how close a partial term is to a solution.

(5) **Choice of *M*.** A larger test set M offers more opportunities to terminate the search at the cost of more time being spent testing each valid partial term. As an extreme case of this, we could take M large enough to include a term for every k-ary term operation. In that case, we would be guaranteed to find a solution at Level 1. But the time to check through all the terms of M for a solution would quickly become unfeasible, as demonstrated in [5], Fig. 3. At minimum we require the variables to be in M since, if the only solutions to an array are among the variables, they would not otherwise be found.

The next theorem draws on Theorem 1 and Lemma 3 to explain the essential role of idemprimality in the success of BEAs.

Theorem 4. *Assume that an idemprimal groupoid* $\mathbf{G}$ *and a target array* T *that has a solution are input to a BEA. Then every array produced by the BEA also has a solution.*

Proof. This is true at the first level since T has a solution. Consider a partial term $f(\vec{x}, \Diamond)$ valid with respect to its parent array A that has a solution, and let $B = [f(\vec{x}, \Diamond) : A]$ be its validity array. By Theorem 1, there is a term $u(\vec{x})$ such $f(\vec{x}, u(\vec{x}))$ is a solution to A. By Lemma 3, the term $u(\vec{x})$ is a solution to B. $\square$

4. Termination

[5, Theorem 8] showed that the Deep Drilling Algorithm will either terminate and deliver a correct solution or will never terminate. The same is true for BEAs.

Theorem 5 (Termination Theorem). *For every finite groupoid* $\mathbf{G}$ *and target array* T *over* $\mathbf{G}$ *that are input to a BEA, the following are equivalent for each integer* $H \geq 1$:

(i) *The BEA will produce a solution to* T *from a partial term at Level* H.
(ii) *The BEA will find in* M *a solution to some validity array at Level* H.
(iii) *The BEA will terminate in step* (2a) *immediately after constructing Level* H *of the tree.*

Proof. Assume (i). This can only happen if (2a) is executed, which implies that (ii) has occurred. If (ii) occurs, then Lemma 3 tells us that the BEA will apply (2a) and generate a solution to each validity array on the path from that node of the tree to the root and will then produce a solution to T and terminate, giving us (iii). If (iii) occurs, then the BEA must have invoked step (2a) immediately after constructing Level H so (i) must have occurred. □

Knowing that solving the target is equivalent to terminating, we would like to see just what non-termination of a BEA can look like. Each run begins with Level 1 for $\mathfrak{T}$, consisting of the partial term $\Diamond$ and its validity array T over T. There are only two ways that a BEA can fail to terminate.

(1) For some $H \geq 1$ it never finds b valid partial terms to construct Level $H+1$.
(2) It constructs Level $H+1$ for every $H \geq 1$ without ever finding a solution in M to any array in $\mathfrak{T}$.

If the groupoid is idemprimal, it turns out that (2) accounts for almost all failures of a BEA to terminate.

Theorem 6. *Assume that an idemprimal groupoid* $\mathbf{G}$ *and a target array* T *that has a solution are input to a BEA. Then, for each* $H \geq 1$, *a run of the BEA that does not eventually terminate will construct Level* $H+1$ *of* $\mathfrak{T}$ *with probability one.*

Proof. Consider runs of the BEA that do not terminate and return a solution. We will show that they construct Level H of $\mathfrak{T}$ with probability one by induction on H. They all construct Level 1 since Level 1 has already been constructed when they start. Assume that they construct Level H of $\mathfrak{T}$ with probability one. If we can show that they will construct Level $H+1$ with probability one whenever they have constructed Level H, it will follow that they will construct Level $H+1$ of $\mathfrak{T}$ with probability one. Assume a run has constructed Level H of $\mathfrak{T}$.

Since it does not terminate at Level $H+1$, no solution to the Level H partial terms will be found in M at Level $H+1$. It will therefore invoke the VTG to look for partial terms valid with respect to some Level H array for Level $H+1$.

Consider any Level H array A. By Theorem 4, we know that A has a solution. Since A does not have a solution in M and M contains the variables, we know that it has a solution $u(\vec{x})v(\vec{x})$ that is a product. Consequently $u(\vec{x})\Diamond$ and $\Diamond v(\vec{x})$ are two partial terms valid with respect to A. By our choice of h, there are terms $u'(\vec{x})$ and $v'(\vec{x})$ having, respectively, the same term operations as $u(\vec{x})$ and $v(\vec{x})$ and both having height at most h. Thus, for each array at level H there are at least two partial terms, $u'(\vec{x})\Diamond$ and $\Diamond v'(\vec{x})$, that the VTG could produce at Level $H+1$ which are valid with respect to that array. Since we allow duplicates, the VTG will eventually produce b of them for Level $H+1$ with probability one. Calculating their validity arrays will construct Level $H+1$ with probability one. □

In view of Theorem 6, it remains to find out if the BEA will eventually find in M a solution to some array in $\mathcal{T}$. Looking at Fig. 3, we must ask:

(∗) *What would give us reason to believe that some array in* $\mathcal{T}$ *is likely to have a solution in* M*?*

Perhaps a prudent choice of M? Recall that our only constraint on the finite set M is that it contain the single variables. A larger choice of M has more chance of containing a solution, but costs in time to check every validity array produced by the VTG for that solution. We are generally looking for the solution in a vast search space of k-ary term operations, so we cannot expect it to arise by accident. We need a good reason to believe that the BEA will find one.

5. Partial Terms Valid with Respect to the Target Array

To answer the question (∗), we consider an arbitrary finite groupoid $\mathbf{G}$ with a k-ary target array T that has a solution as inputs to a BEA. In the tree $\mathcal{T}$ of partial terms that it produces, each partial term is accompanied by its validity array over the array at its parent node (or over T, if it is the root). We will show in this section that there is another way we can label the nodes of the tree $\mathcal{T}$ that might be more illuminating. We start with the underlying tree structure of $\mathcal{T}$ and construct the **integrated tree** $\mathcal{T}'$ by recursively relabeling the nodes with only partial terms.

(1) The root of $\mathcal{T}'$ is labeled $\Diamond$.
(2) Assume that a parent node is labeled $p(\vec{x}, \Diamond)$ in $\mathcal{T}'$ and a child of that node is labeled $g(\vec{x}, \Diamond)$ in $\mathcal{T}$. Then that child node is labeled $p(\vec{x}, g(\vec{x}, \Diamond))$ in $\mathcal{T}'$.

Applying this construction to the tree $\mathcal{T}$ in Fig. 3, we obtain the tree $\mathcal{T}'$ in Fig. 4.

Note that the partial term at the upper left is valid with respect to T because we can complete it to the solution we previously found to T by replacing $\Diamond$ with $x_4^2x_0^2$. Similarly, the partial term at its parent node is valid with respect to T using instead the replacement term $(x_4x_1)(x_4^2x_0^2)$. Continuing in this same way down to the root suggests that perhaps *every* partial term in $\mathcal{T}'$ is valid with respect to T if $\mathbf{G}$ is

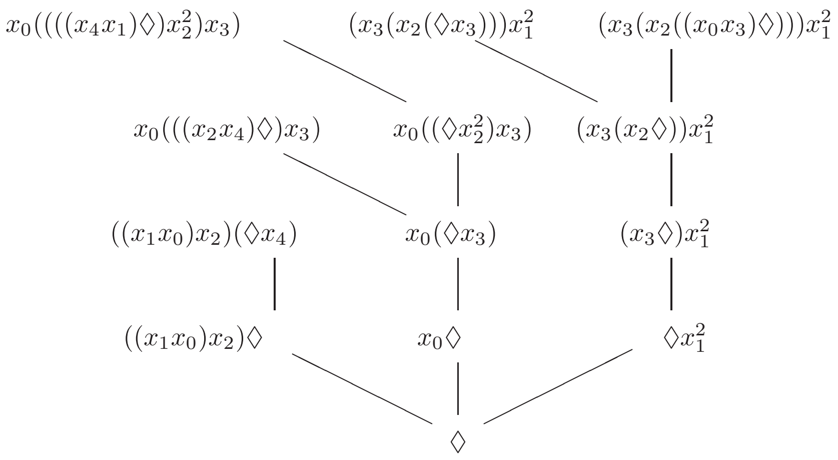

Fig. 4. Integrated tree $\mathcal{T}'$, with each partial term a compilation of those up to it in $\mathcal{T}$.

idemprimal since all the validity arrays have solutions. The next theorem gives two important connections between the trees $\mathcal{T}$ and $\mathcal{T}'$.

Theorem 7. *Assume that a finite groupoid* $\mathbf{G}$ *and* k*-ary target array* T *are input to a BEA, which then generates the tree* $\mathcal{T}$ *of partial terms and their validity arrays. Let* $\mathcal{T}'$ *be the resulting integrated tree and let* $f(\vec{x},\Diamond)$ *be any partial term of* $\mathcal{T}'$.

(i) *The height* (*level*) *of* $f(\vec{x},\Diamond)$ *in* $\mathcal{T}'$ *is the same as the height* (*depth*) *of* $\Diamond$ *in the term tree of* $f(\vec{x},\Diamond)$.

(ii) $f(\vec{x},\Diamond)$ *has the same validity array over* T *as the corresponding partial term in* $\mathcal{T}$ *has over its parent array.*

Proof. (i). At the root of $\mathcal{T}'$, the partial term $\Diamond$ is at Level 1 in $\mathcal{T}'$ and has the variable $\Diamond$ at depth 1 in its term tree. Assume the statement is true for partial terms at Level H in $\mathcal{T}'$, and that $f(\vec{x},\Diamond)$ is at Level $H+1$. Then there is a term $u(\vec{x})$ and a partial term $p(\vec{x},\Diamond)$ at Level H in $\mathcal{T}'$ such that $f(\vec{x},\Diamond)$ is either $p(\vec{x},u(\vec{x})\Diamond)$ or $p(\vec{x},\Diamond u(\vec{x}))$. In either case $\Diamond$ has depth $H+1$ in the term tree of $f(\vec{x},\Diamond)$. The proof follows by induction on the level of $f(\vec{x},\Diamond)$ in $\mathcal{T}'$.

(ii). At the root of $\mathcal{T}'$ the partial term $\Diamond$ has validity array $[\Diamond : T] = T$ over T while $\mathcal{T}$ has $\Diamond$ with parent array T and validity array $[\Diamond : T] = T$. Assume it is true for partial terms at Level H nodes and that $f(\vec{x},\Diamond)$ is at a Level $H+1$ node ν in $\mathcal{T}'$. Let $p(\vec{x},\Diamond)$ be the partial term at the Level H parent node μ of ν. In $\mathcal{T}$ let $q(\vec{x},\Diamond)$ be the partial term at μ and let A be the validity array of $q(\vec{x},\Diamond)$ over its parent array B. Let $g(\vec{x},\Diamond)$ be the partial term at ν in $\mathcal{T}$ that is valid with respect to A. (see Fig. 5.)

We must verify that $[f(\vec{x},\Diamond) : T] = [g(\vec{x},\Diamond) : A]$. From the definition of $\mathcal{T}'$ we have $f(\vec{x},\Diamond) = p(\vec{x}, g(\vec{x},\Diamond))$. By induction, $[p(\vec{x},\Diamond) : T] = [q(\vec{x},\Diamond) : B]$. Let $\vec{a} \in G^k$

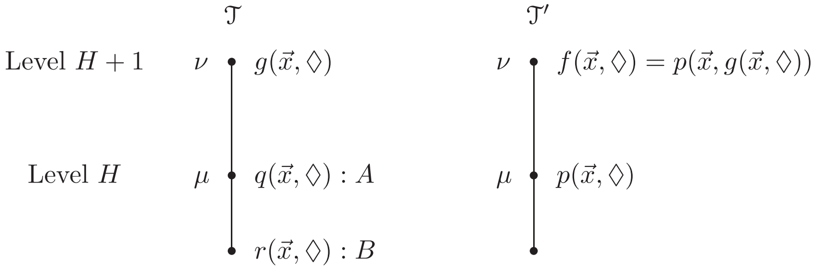

Fig. 5. Theorem 7(ii) $[g(\vec{x}, \Diamond) : A] = [f(\vec{x}, \Diamond) : T]$.

and $b \in G$. Then we have

$$\begin{aligned} b \in [f(\vec{x}, \Diamond) : T](\vec{a}) &\Leftrightarrow f(\vec{a}, b) \in T(\vec{a}) \\ &\Leftrightarrow p(\vec{a}, g(\vec{a}, b)) \in T(\vec{a}) \\ &\Leftrightarrow g(\vec{a}, b) \in [p(\vec{x}, \Diamond) : T](\vec{a}) \\ &= [q(\vec{x}, \Diamond) : B](\vec{a}) = A(\vec{a}) \\ &\Leftrightarrow b \in [g(\vec{x}, \Diamond) : A](\vec{a}) \end{aligned}$$

and consequently $[f(\vec{x}, \Diamond) : T] = [g(\vec{x}, \Diamond) : A]$. □

Corollary 8. *Assume that a finite groupoid* **G** *and a target array T are input to a BEA. Then some term in M will complete some partial term of $\mathfrak{T}$ to a solution to its parent array if and only if that same term of M completes the corresponding partial term of $\mathfrak{T}'$ to a solution to T.*

Proof. A term of M will complete a partial term of $\mathfrak{T}$ to a solution to its parent array if and only if it is a solution to the validity array of that partial term over its parent array. Similarly, a term of M will complete the corresponding partial term of $\mathfrak{T}'$ to a solution to T if and only if it is a solution to the validity array of that partial term over T. By Theorem 7(ii) those two validity arrays are the same. □

This corollary to Theorem 7(ii) tells us that the question $(*)$ translates into the equivalent question

$(*')$ *What would give us reason to believe that some term of M will complete some partial term in $\mathfrak{T}'$ to a solution to T?*

At last, Theorem 7(i) suggests an answer! As we go up the tree $\mathfrak{T}'$, the $\Diamond$s go progressively deeper in the partial terms. This fact could have significant consequences if the groupoid is *k-TC*.

Here is how it might work. Assume that the groupoid **G** is both idemprimal and k-TC, and we input **G** and a target array T that has a solution to a BEA.

Now consider a partial term $f(\vec{x}, \Diamond)$ that is very high in the tree $\mathcal{T}'$, and therefore has $\Diamond$ very deep in its term tree. If $f(\vec{x}, u(\vec{x}))$ is any completion of $f(\vec{x}, \Diamond)$, then term continuity says that other completions $f(\vec{x}, v(\vec{x}))$ of $f(\vec{x}, \Diamond)$ will have term operations that differ from $f(\vec{x}, u(\vec{x}))^{\mathbf{G}}$, and therefore from each other, in few places. That means that the set of all term operations, we get from completions of $f(\vec{x}, \Diamond)$ should be small, each being, on average, produced by a large proportion of the k-ary terms. Since **G** is idemprimal and T has a solution, Theorem 4 says that some completion $f(\vec{x}, u(\vec{x}))$ of $f(\vec{x}, \Diamond)$ is a solution to the parent array of $f(\vec{x}, \Diamond)$ and therefore, by Corollary 8, is also a solution to T. But then we should expect, for a large proportion of k-ary terms $v(\vec{x})$, that $f(\vec{x}, v(\vec{x}))$ will have the same term operation as $f(\vec{x}, u(\vec{x}))$ and therefore also be a solution to T. As the proportion of terms that produce a solution to T by completing a partial term of $\mathcal{T}'$ rises, the likelihood that a term of M will be among them should rise as well. If one does appear in M, then we will get a solution to T.

If this is all correct, then we would have an answer to $(*')$, and therefore also to $(*)$: We would expect that some term of M will eventually give us a solution to T if the groupoid **G** is *idemprimal* and *k-TC*. In Sec. 6, we will prove Theorem 21 saying that, if $k > 1$ and the groupoid is k-TC, then the set of term operations that can be produced from a partial term high in $\mathcal{T}'$ will often be small in a dramatic sense: It will contain only one term operation! If the groupoid is also idemprimal, then some term must complete those partial terms to a solution to T. It follows that *every* term will complete them to a solution to T, and the choice of the test set M will not even be relevant!

6. Constant Partial Terms from Term Continuity

The strategy above depends on the following notion. A finite groupoid **G** and a partial k-ary term $f(\vec{x}, \Diamond)$ together define the map from the set of all k-ary terms into the set of k-ary term operations on **G** taking $u(\vec{x})$ to $f(\vec{x}, u(\vec{x}))^{\mathbf{G}}$. We say that the partial term $f(\vec{x}, \Diamond)$ is **constant** on **G** if the range of this map contains only one operation. This notion allows us to state a condition that is sufficient to guarantee the success of a run of a BEA.

Theorem 9. *Assume that an idemprimal groupoid* **G** *and target array T that has a solution are input to a BEA. If the BEA eventually generates a partial term in $\mathcal{T}$ whose associated partial term in $\mathcal{T}'$ is constant on* **G**, *then it will return a solution to T and terminate.*

Proof. Let $g(\vec{x}, \Diamond)$ be the first partial term produced by the BEA whose corresponding partial term $f(\vec{x}, \Diamond)$ at the same node in $\mathcal{T}'$ is constant, and let A be the parent array of $g(\vec{x}, \Diamond)$. Since **G** is idemprimal and T has a solution, Theorem 4 and Corollary 8 say that some term completes $f(\vec{x}, \Diamond)$ to a solution to T. Since $f(\vec{x}, \Diamond)$ is constant, every term must complete it to a solution to T.

Let $m(\vec{x})$ be the first term of M that is tested on $g(\vec{x},\Diamond)$. Since $f(\vec{x},m(\vec{x}))$ is a solution to T, Corollary 8 tells us that $g(\vec{x},m(\vec{x}))$ is a solution to A. By the Termination Theorem 5(ii) it follows that the BEA will then (i) return a solution to T and (iii) terminate. □

The remainder of this section will not be concerned with either the TGP, BEAs or idemprimality. Its only goal is to investigate, for a finite groupoid, the frequency of occurrence of constant partial terms among all partial terms. In general there is no reason to expect a non-trivial finite groupoid to have any constant partial terms at all, as a familiar example shows.

Example 10. A non-trivial finite group $\mathbf{G}$ has no constant partial term.

Proof. Suppose $f(\vec{x},\Diamond)$ is a constant partial term and let $\|G\| = n$. Since $\mathbf{G}$ is associative and satisfies $x^n \approx e$, there are groupoid terms $p(\vec{x})$ and $q(\vec{x})$ such that $\mathbf{G}$ satisfies the identity $f(\vec{x},\Diamond) \approx p(\vec{x})\Diamond q(\vec{x})$. Then $\mathbf{G}$ satisfies

$$x_0 \approx f(\vec{x}, p(\vec{x})^{n-1}x_0q(\vec{x})^{n-1}) \approx f(\vec{x}, p(\vec{x})^{n-1}x_0^2q(\vec{x})^{n-1}) \approx x_0^2.$$

Consequently $\mathbf{G}$ satisfies $e \approx x_0$ and is therefore trivial, a contradiction. □

Let $\mathbb{N}$ denote the set of all positive integers. We will fix a finite groupoid $\mathbf{G} = \langle G, *\rangle$ and an integer $k \in \mathbb{N}$. Let $\mathcal{T}m$ be the set of all k-ary terms. For all $h, H \in \mathbb{N}$ with $h \leq H$, let $\mathcal{T}m_H$ be the set of all terms in $\mathcal{T}m$ of height at most H, let $\mathcal{F}_H$ be the set of partial k-ary terms of height at most H and $\mathcal{F}_{H,h}$ be the set of partial terms in $\mathcal{F}_H$ with $\Diamond$ at height h. Let $\mathcal{C}$ denote the set of all partial k-ary terms that are constant on $\mathbf{G}$ and define $\mathcal{C}_H := \mathcal{F}_H \cap \mathcal{C}$ and $\mathcal{C}_{H,h} := \mathcal{F}_{H,h} \cap \mathcal{C}$. Then we have partitions

$$\begin{aligned} \mathcal{F}_H &= \mathcal{F}_{H,1} \,\dot\cup\, \mathcal{F}_{H,2} \,\dot\cup\, \cdots \,\dot\cup\, \mathcal{F}_{H,H-1} \,\dot\cup\, \mathcal{F}_{H,H}, \\ \mathcal{C}_H &= \mathcal{C}_{H,1} \,\dot\cup\, \mathcal{C}_{H,2} \,\dot\cup\, \cdots \,\dot\cup\, \mathcal{C}_{H,H-1} \,\dot\cup\, \mathcal{C}_{H,H}. \end{aligned} \tag{1}$$

It will be useful to have names for the sizes of all of these sets. Taking $t_H := \|\mathcal{T}m_H\|$, we have $t_1 = k$ and $t_{H+1} = t_H^2 + k$, as every term of height at most $H+1$ is either a product of two terms of height at most H or a variable. This sequence grows very fast for all $k \in \mathbb{N}$. In particular, if $k > 1$, then

$$t_H \geq k^{2^{H-1}} \tag{2}$$

for all positive integers H. For $h \leq H$ in $\mathbb{N}$, we will use the abbreviations

$$f_H := \|\mathcal{F}_H\| \quad f_{H,h} := \|\mathcal{F}_{H,h}\| \quad c_H := \|\mathcal{C}_H\| \quad c_{H,h} := \|\mathcal{C}_{H,h}\|$$

to obtain from (1) the breakdowns

$$\begin{aligned} f_H &= f_{H,1} + f_{H,2} + \cdots + f_{H,H-1} + f_{H,H}, \\ c_H &= c_{H,1} + c_{H,2} + \cdots + c_{H,H-1} + c_{H,H}. \end{aligned} \tag{3}$$

In this section, we will examine the asymptotic behavior of the sequence

$$q_H = \frac{\|\mathcal{C}_H\|}{\|\mathcal{F}_H\|} = \frac{c_H}{f_H} = \frac{c_{H,1} + c_{H,2} + \cdots + c_{H,H-1} + c_{H,H}}{f_{H,1} + f_{H,2} + \cdots + f_{H,H-1} + f_{H,H}}.$$

* * * * *

Lemma 11. *For every $k > 1$ in $\mathbb{N}$, every non-trivial finite groupoid* $\mathbf{G}$ *and every $h \leq H$ in $\mathbb{N}$, we have*

(i) $f_{H,1} = 1$ *and* $f_{H+1,h+1} = 2t_H f_{H,h}$,
(ii) $f_{H+1} = 1 + 2t_H f_H$,
(iii) $c_{H,1} = 0$ *and* $c_{H+1,h+1} \geq 2t_H c_{H,h}$,
(iv) $c_{H+1} \geq 2t_H c_H$.

Proof. (i) As $\Diamond$ is the only partial term with $\Diamond$ at height one, $\mathcal{F}_{H,1} = \{\Diamond\}$ and so $f_{H,1} = 1$. Since every partial term in $\mathcal{F}_{H+1,h+1}$ is a unique product

$$\mathcal{F}_{H+1,h+1} = \mathcal{T}m_H * \mathcal{F}_{H,h} \mathbin{\dot\cup} \mathcal{F}_{H,h} * \mathcal{T}m_H,$$

we have $f_{H+1,h+1} = t_H f_{H,h} + f_{H,h} t_H = 2t_H f_{H,h}$.

(ii) Part (i) and (3) give us

$$\begin{aligned} f_{H+1} &= f_{H+1,1} + f_{H+1,2} + \cdots + f_{H+1,H} + f_{H+1,H+1} \\ &= 1 + 2t_H(f_{H,1} + \cdots + f_{H,H-1} + f_{H,H}) \\ &= 1 + 2t_H f_H. \end{aligned}$$

(iii) Since $k > 1$ and $\mathbf{G}$ is non-trivial, the partial term $\Diamond$ is not constant so $\mathcal{C}_{H,1}$ is empty and $c_{H,1} = 0$. From (i) each partial term $f(\vec{x}, \Diamond)$ of $\mathcal{F}_{H,h}$ extends to $2t_H$ partial terms of $\mathcal{F}_{H+1,h+1}$. If $f(\vec{x}, \Diamond)$ is constant, then those $2t_H$ extensions are all constant as well. Thus, $c_{H+1,h+1} \geq 2t_H c_{H,h}$.

(iv) Applying (iii) and (3), we have

$$\begin{aligned} c_{H+1} &= c_{H+1,1} + c_{H+1,2} + \ldots + c_{H+1,H} + c_{H+1,H+1} \\ &\geq 0 + 2t_H(c_{H,1} + \ldots + c_{H,H-1} + c_{H,H}) = 2t_H c_H. \end{aligned}$$ □

Our initial goal is to prove that the sequence $q_1, q_2, q_3, \ldots$ does indeed converge. Although it is not necessarily non-decreasing, it comes very close.

Lemma 12. *If* $\mathbf{G}$ *is a non-trivial finite groupoid with $J \geq H$ and $k \in \mathbb{N}$, then*

$$q_H - q_J < \frac{1}{k^{H-2}}.$$

Proof. If $k = 1$, this is true, so assume $k > 1$. We can now apply Lemma 11(ii), (iv) and (2) to show that q_H cannot exceed q_{H+1} by more than a *very* small margin.

$$\begin{aligned} q_H - q_{H+1} &= \frac{c_H}{f_H} - \frac{c_{H+1}}{f_{H+1}} \leq \frac{c_H}{f_H} - \frac{2t_H c_H}{1 + 2t_H f_H} \\ &= \frac{c_H(1 + 2t_H f_H) - f_H(2t_H c_H)}{f_H(1 + 2t_H f_H)} \\ &= \frac{c_H}{f_H(1 + 2t_H f_H)} < \frac{f_H}{f_H(0 + 1t_H \cdot 1)} \\ &= \frac{1}{t_H} \leq \frac{1}{k^{2^{H-1}}} < \frac{1}{k^{H-1}}, \end{aligned}$$

$$\text{and therefore } q_H - q_{H+1} < \frac{1}{k^{H-1}}.$$

Using this inequality, we obtain, for all $J > H$ in $\mathbb{N}$,

$$\begin{aligned} q_H - q_J &= \sum_{I=H}^{J-1} (q_I - q_{I+1}) < \sum_{I=H}^{J-1} \frac{1}{k^{I-1}} < \sum_{I=H}^{\infty} \frac{1}{k^{I-1}} \\ &= \frac{1}{k^{H-2}(k-1)} \leq \frac{1}{k^{H-2}}. \end{aligned}$$

□

Theorem 13. *For every finite groupoid* $\mathbf{G}$ *and every integer* $k > 1$, *the sequence* $q_1, q_2, q_3, \ldots$ *converges to some limit* λ *in* $[0, 1]$.

Proof. We assume that $\mathbf{G}$ is non-trivial since the sequence converges to $\lambda = 1$ when it is trivial. For each $H \in \mathbb{N}$ the set $\{q_H, q_{H+1}, q_{H+2}, \ldots\}$ is bounded above by 1. Let u_H be the least upper bound of this set. Then the sequence $u_1, u_2, u_3, \ldots$ is non-increasing and is bounded below by 0. We will take λ to be the greatest lower bound of the set $\{u_1, u_2, u_3, \ldots\}$ and show that $q_1, q_2, q_3, \ldots$ converges to λ.

Let $\epsilon > 0$ and choose $H \in \mathbb{N}$ such that $u_H < \lambda + \epsilon/2$ and $1/k^{H-2} < \epsilon/2$. Since $u_H - \epsilon/2$ is not an upper bound for $\{q_H, q_{H+1}, q_{H+2} \ldots\}$, there is a $J \geq H$ such that $u_H - \epsilon/2 < q_J$. Now choose any $K > J$ in $\mathbb{N}$. By Lemma 12, we have $q_J < q_K + 1/k^{J-2} < q_K + \epsilon/2$ and therefore

$$\frac{\epsilon}{2} > u_H - q_J > \lambda - \left(q_K + \frac{\epsilon}{2}\right). \text{ Thus, } \lambda - \epsilon < q_K \leq u_K \leq u_J \leq u_H < \lambda + \frac{\epsilon}{2}.$$

It follows that $|q_K - \lambda| < \epsilon$ for all $K > J$. □

* * * * *

It remains to examine the value of the limit λ. For example, if $\mathbf{G}$ is a non-trivial finite group, then Example 10 tells us that each q_H is 0 and consequently $\lambda = 0$. However, the discussion at the end of Sec. 5 suggests that term continuity may lead to positive values of λ. To see if this is true, we will review background about term continuity from [3] and then, in Theorem 15, present a new condition equivalent to term continuity that is tailored to the present context.

We say that a set $\mathcal{V} \subseteq \mathcal{T}m$ is a **mutation set** if it is finite and it contains the set $\mathcal{T}m_1$ of variables. For each $H \in \mathbb{N}$ let $\mathcal{U}_H$ be the set of all pairs $(t(\vec{x}, \Diamond), u(\vec{x}))$ such that $t(\vec{x}, \Diamond)$ is a partial term, $u(\vec{x})$ is a term, and $t(\vec{x}, u(\vec{x})) \in \mathcal{T}m_H$. Relative to a choice $\mathcal{V}$ of mutation set, we define

$$\mathcal{M}_H := \mathcal{U}_H \times \mathcal{V} = \{(t, u, v) \mid (t, u) \in \mathcal{U}_H \text{ and } v \in \mathcal{V}\},$$

where "(t, u, v)" is an abbreviation for "$((t, u), v)$". A triple $M := (t, u, v) = (t(\vec{x}, \Diamond), u(\vec{x}), v(\vec{x}))$ is a **mutation** if it is in $\mathcal{M}_H$ for some $H \in \mathbb{N}$. We think of M as replacing the subterm $u(\vec{x})$ of $t(\vec{x}, u(\vec{x}))$ with $v(\vec{x})$ to form the term $t(\vec{x}, v(\vec{x}))$. Note that each $\mathcal{M}_H$ is finite, allowing us to compute sums and probabilities over $\mathcal{M}_H$.

We can now state the definition from [3] of the Continuity Condition for Groupoids mentioned in the introduction. Given a mutation $M = (t, u, v)$, we denote by $HD(M)$ the Hamming distance between $t(\vec{x}, u(\vec{x}))^{\mathbf{G}}$ and $t(\vec{x}, v(\vec{x}))^{\mathbf{G}}$, that is, the number of k-tuples on which their values differ. For each positive integer H, the **mean Hamming distance**, $\mu HD(H)$, is the average value of $HD(M)$ over all $M \in \mathcal{M}_H$. The finite groupoid $\mathbf{G}$ is k-**TC** if, for every mutation set $\mathcal{V}$,

$$\lim_{H \to \infty} \mu HD(H) = 0.$$

In [3] Theorem 3 shows that this condition is equivalent to two others. We will make use of the third.

Theorem 14 ([3, Theorem 3(iii)]). *Let* $\mathbf{G}$ *be a finite groupoid and let* $1 < k \in \mathbb{N}$. *Then* $\mathbf{G}$ *is* k*-TC if and only if, for every mutation set* $\mathcal{V}$,

$$\lim_{H \to \infty} \operatorname{Prob} \langle t(\vec{x}, u(\vec{x}))^{\mathbf{G}} = t(\vec{x}, v(\vec{x}))^{\mathbf{G}} : (t, u, v) \in \mathcal{M}_H \rangle = 1.$$

Continuing with a fixed finite groupoid $\mathbf{G}$ we extend the above notation from [3] by defining, for all positive integers $h \leq H$, the sets

$$\begin{aligned}
\mathcal{U}_H^{\mathcal{C}} &:= \{(f, u) \in \mathcal{U}_H \mid f(\vec{x}, \Diamond) \in \mathcal{C}_H\}, \\
\mathcal{U}_{H,h} &:= \{(f, u) \in \mathcal{U}_H \mid f(\vec{x}, \Diamond) \in \mathcal{F}_{H,h}\}, \\
\mathcal{U}_{H,h}^{\mathcal{C}} &:= \mathcal{U}_H^{\mathcal{C}} \cap \mathcal{U}_{H,h}, \\
\mathcal{M}_H^{\mathcal{C}} &:= \mathcal{U}_H^{\mathcal{C}} \times \mathcal{V}.
\end{aligned}$$

These sets give us partitions of $\mathcal{U}_H$ and $\mathcal{U}_H^{\mathcal{C}}$ as

$$\begin{aligned}
\mathcal{U}_H &= \mathcal{U}_{H,1} \,\dot\cup\, \mathcal{U}_{H,2} \,\dot\cup\, \cdots \,\dot\cup\, \mathcal{U}_{H,H-1} \,\dot\cup\, \mathcal{U}_{H,H}, \\
\mathcal{U}_H^{\mathcal{C}} &= \mathcal{U}_{H,1}^{\mathcal{C}} \,\dot\cup\, \mathcal{U}_{H,2}^{\mathcal{C}} \,\dot\cup\, \cdots \,\dot\cup\, \mathcal{U}_{H,H-1}^{\mathcal{C}} \,\dot\cup\, \mathcal{U}_{H,H}^{\mathcal{C}}.
\end{aligned} \tag{4}$$

For each positive integer H, we define

$$\mathcal{M}_H^{=} := \{(t, u, v) \in \mathcal{M}_H : t(\vec{x}, u(\vec{x}))^{\mathbf{G}} = t(\vec{x}, v(\vec{x}))^{\mathbf{G}}\}.$$

Thus, $\mathcal{M}_H^{=}$ consists of all mutations that have no effect on the underlying term operation of $\mathbf{G}$. Note that, for every mutation set $\mathcal{V}$ and every positive integer H,

$$\mathcal{M}_H^{\mathcal{C}} \subseteq \mathcal{M}_H^{=} \subseteq \mathcal{M}_H.$$

In this notation, the condition for term continuity of Theorem 14 above can be rephrased as

$$\lim_{H\to\infty} \frac{\|\mathcal{M}_H^{=}\|}{\|\mathcal{M}_H\|} = 1 \quad \text{for every mutation set } \mathcal{V}. \tag{5}$$

Note that neither this condition nor the definition of term continuity either mention constant partial terms or in any immediate way imply the existence of constant partial terms. We now prove a theorem which gives a fourth condition equivalent to term continuity that is significant in two ways. One is that it *makes no mention of the mutation set* $\mathcal{V}$. The other is that it *guarantees an abundance of constant partial terms for any finite groupoid that is k-TC.*

Theorem 15. *For* $1 < k$, *a finite groupoid* $\mathbf{G}$ *is* k*-TC if and only if*

$$\lim_{H\to\infty} \frac{\|\mathcal{U}_H^{\mathcal{C}}\|}{\|\mathcal{U}_H\|} = 1.$$

In words this says that, asymptotically, for almost all pairs $(f(\vec{x},\Diamond), u(\vec{x}))$ *for which* $f(\vec{x}, u(\vec{x}))$ *has height at most* H, *the partial* k*-ary term* $f(\vec{x},\Diamond)$ *is constant.*

Proof. We will use the condition (5) for k-term continuity. First assume that the sequence displayed in the statement of the theorem converges to 1. Then, for every mutation set $\mathcal{V}$ and every $H \in \mathbb{N}$, we have

$$\frac{\|\mathcal{M}_H^{=}\|}{\|\mathcal{M}_H\|} \geq \frac{\|\mathcal{U}_H^{\mathcal{C}} \times \mathcal{V}\|}{\|\mathcal{M}_H\|} = \frac{\|\mathcal{U}_H^{\mathcal{C}} \times \mathcal{V}\|}{\|\mathcal{U}_H \times \mathcal{V}\|} = \frac{\|\mathcal{U}_H^{\mathcal{C}}\|}{\|\mathcal{U}_H\|}.$$

It follows that (5) holds, and therefore $\mathbf{G}$ is k-TC by Theorem 14.

Conversely assume $\mathbf{G}$ is k-TC and choose $\epsilon > 0$. Since $\mathbf{G}$ is finite, we can choose a (finite) mutation set $\mathcal{V}$ such that every k-ary term operation of $\mathbf{G}$ is represented by some member of $\mathcal{V}$. By (5) there is a positive integer H_0 such that $H \geq H_0$ implies that

$$\frac{\|\mathcal{M}_H^{=}\|}{\|\mathcal{M}_H\|} > 1 - \frac{\epsilon}{\|\mathcal{V}\|}.$$

Let $H > H_0$ and define $\mathcal{N}_H := \mathcal{M}_H \backslash \mathcal{M}_H^{=}$ and $\mathcal{N}'_H := \mathcal{U}_H \backslash \mathcal{U}_H^{\mathcal{C}}$. Then

$$1 - \frac{\epsilon}{\|\mathcal{V}\|} < \frac{\|\mathcal{M}_H^{=}\|}{\|\mathcal{M}_H\|} = 1 - \frac{\|\mathcal{N}_H\|}{\|\mathcal{U}_H \times \mathcal{V}\|} = 1 - \frac{\|\mathcal{N}_H\|}{\|\mathcal{U}_H\| \cdot \|\mathcal{V}\|}$$

and consequently $\|\mathcal{N}_H\| < \|\mathcal{U}_H\|\epsilon$.

We will show that there are many constant partial terms from $\mathcal{U}_H$ by showing that the set $\mathcal{N}'_H$ of pairs (f,u) in $\mathcal{U}_H$ for which f is *not* constant is also small. Let $(f,u) \in \mathcal{N}'_H$. By our choice of $\mathcal{V}$, there is a $v \in \mathcal{V}$ such that $f(\vec{x}, u(\vec{x}))^{\mathbf{G}} \neq f(\vec{x}, v(\vec{x}))^{\mathbf{G}}$. Since $(f,u) \mapsto (f,u,v) \in \mathcal{N}_H$ is a one-to-one map from $\mathcal{N}'_H$ into $\mathcal{N}_H$, we see that $\|\mathcal{N}'_H\| \leq \|\mathcal{N}_H\|$. Now we have

$$\frac{\|\mathcal{U}_H^{\mathcal{C}}\|}{\|\mathcal{U}_H\|} = 1 - \frac{\|\mathcal{N}'_H\|}{\|\mathcal{U}_H\|} \geq 1 - \frac{\|\mathcal{N}_H\|}{\|\mathcal{U}_H\|} > 1 - \frac{\|\mathcal{U}_H\|\epsilon}{\|\mathcal{U}_H\|} = 1 - \epsilon.$$

Consequently $\|\mathcal{U}_H^{\mathcal{C}}\|/\|\mathcal{U}_H\|$ converges to 1. □

Corollary 16. *For* $1 < k$*, a finite groupoid* $\mathbf{G}$ *is* k*-TC if and only if*

$$\lim_{H\to\infty} \frac{c_{H,1}t_H + c_{H,2}t_{H-1} + \cdots + c_{H,H-1}t_2 + c_{H,H}t_1}{f_{H,1}t_H + f_{H,2}t_{H-1} + \cdots + f_{H,H-1}t_2 + f_{H,H}t_1} = 1.$$

Proof. We show that this statement is obtained by writing Theorem 15 in more explicit detail. Note that for each positive integer $h \leq H$, we have

$$\mathcal{U}_{H,h} = \mathcal{F}_{H,h} \times \mathcal{T}m_{H-h+1} \quad \text{and} \quad \mathcal{U}^{\mathcal{C}}_{H,h} = \mathcal{C}_{H,h} \times \mathcal{T}m_{H-h+1}.$$

Using the abbreviation $c_{H,h} := \|\mathcal{C}_{H,h}\|$ and the partitions (4) of $\mathcal{U}^{\mathcal{C}}_H$ and $\mathcal{U}_H$, we can expand the quotient in Theorem 15.

$$\begin{aligned}\frac{\|\mathcal{U}^{\mathcal{C}}_H\|}{\|\mathcal{U}_H\|} &= \frac{\|\mathcal{U}^{\mathcal{C}}_{H,1} \,\dot\cup\, \mathcal{U}^{\mathcal{C}}_{H,2} \,\dot\cup\, \cdots \,\dot\cup\, \mathcal{U}^{\mathcal{C}}_{H,H-1} \,\dot\cup\, \mathcal{U}^{\mathcal{C}}_{H,H}\|}{\|\mathcal{U}_{H,1} \,\dot\cup\, \mathcal{U}_{H,2} \,\dot\cup\, \cdots \,\dot\cup\, \mathcal{U}_{H,H-1} \,\dot\cup\, \mathcal{U}_{H,H}\|}\\ &= \frac{\|\mathcal{U}^{\mathcal{C}}_{H,1}\| + \|\mathcal{U}^{\mathcal{C}}_{H,2}\| + \cdots + \|\mathcal{U}^{\mathcal{C}}_{H,H-1}\| + \|\mathcal{U}^{\mathcal{C}}_{H,H}\|}{\|\mathcal{U}_{H,1}\| + \|\mathcal{U}_{H,2}\| + \cdots + \|\mathcal{U}_{H,H-1}\| + \|\mathcal{U}_{H,H}\|}\\ &= \frac{c_{H,1}t_H + c_{H,2}t_{H-1} + \cdots + c_{H,H-1}t_2 + c_{H,H}t_1}{f_{H,1}t_H + f_{H,2}t_{H-1} + \cdots + f_{H,H-1}t_2 + f_{H,H}t_1}.\end{aligned} \tag{6}$$

□

Lemma 17. *For all* $k \in \mathbb{N}$*, every finite groupoid* $\mathbf{G}$ *and all* $h < H$ *in* $\mathbb{N}$*, we have*

(i) $f_{H,h+1} = 2t_{H-h}f_{H,h}$,
(ii) $c_{H,h+1} \geq 2t_{H-h}c_{H,h}$,
(iii) $\frac{c_{H,h}}{f_{H,h}} \leq \frac{c_{H,H}}{f_{H,H}}$.

Proof. (i) Define $\alpha : \mathcal{F}_{H,h+1} \to \mathcal{F}_{H,h}$ as follows. If $f(\vec{x}, \Diamond) \in \mathcal{F}_{H,h+1}$, then there is a unique $g(\vec{x}, \Diamond) \in \mathcal{F}_{H,h}$ and $u(\vec{x}) \in \mathcal{T}m_{H-h}$ such that $f(\vec{x}, \Diamond)$ is either $g(\vec{x}, \Diamond u(\vec{x}))$ or $g(\vec{x}, u(\vec{x})\Diamond)$. Let $\alpha(f(\vec{x}, \Diamond)) := g(\vec{x}, \Diamond)$. Then each partial term in $\mathcal{F}_{H,h}$ is the image of exactly $2t_{H-h}$ members of $\mathcal{F}_{H,h+1}$.

(ii) If $g(\vec{x}, \Diamond) \in \mathcal{C}_{H,h}$, then $\alpha^{-1}(g(\vec{x}, \Diamond))$ is a set of $2t_{H-h}$ members of $\mathcal{C}_{H,h+1}$.

(iii) Applying (i) and (ii), we have

$$\frac{c_{H,h}}{f_{H,h}} \leq \frac{c_{H,h+1}}{f_{H,h+1}} \leq \frac{c_{H,h+2}}{f_{H,h+2}} \leq \cdots \leq \frac{c_{H,H}}{f_{H,H}}.$$

□

Lemma 18. *For* $1 < k \in \mathbb{N}$ *and every finite groupoid* $\mathbf{G}$*, the sequence*

$$\frac{c_{1,1}}{f_{1,1}}, \frac{c_{2,2}}{f_{2,2}}, \frac{c_{3,3}}{f_{3,3}}, \ldots$$

is non-decreasing and bounded above by 1*, and therefore converges to the least upper bound* $\kappa \leq 1$ *of its range.*

Proof. Using Lemma 11(i) and (iii), we have

$$1 \geq \frac{c_{H+1,H+1}}{f_{H+1,H+1}} \geq \frac{2t_H c_{H,H}}{2t_H f_{H,H}} = \frac{c_{H,H}}{f_{H,H}}.$$

□

As an illustration of Lemma 18, we know from Example 10 that $\kappa = 0$ if $\mathbf{G}$ is a non-trivial finite group. We can now exhibit the one place where term continuity enters the argument, which is to determine the value of κ.

Lemma 19. *If* $1 < k \in \mathbb{N}$ *and the finite groupoid* $\mathbf{G}$ *is* k*-TC, then*

$$\lim_{H\to\infty} \frac{c_{H,H}}{f_{H,H}} = \kappa = 1.$$

Proof. By Lemma 17(iii) and Lemma 18, for all $h \leq H$,

$$\frac{c_{H,h}}{f_{H,h}} \leq \frac{c_{H,H}}{f_{H,H}} \leq \kappa \quad \text{so that } c_{H,h} \leq f_{H,h}\kappa.$$

From (6), we have

$$\begin{aligned}\frac{\|\mathcal{U}_H^{\mathcal{C}}\|}{\|\mathcal{U}_H\|} &= \frac{c_{H,1}t_H + c_{H,2}t_{H-1} + \cdots + c_{H,H-1}t_2 + c_{H,H}t_1}{f_{H,1}t_H + f_{H,2}t_{H-1} + \cdots + f_{H,H-1}t_2 + f_{H,H}t_1} \\ &\leq \frac{f_{H,1}t_H\kappa + f_{H,2}t_{H-1}\kappa + \cdots + f_{H,H-1}t_2\kappa + f_{H,H}t_1\kappa}{f_{H,1}t_H + f_{H,2}t_{H-1} + \cdots + f_{H,H-1}t_2 + f_{H,H}t_1} = \kappa.\end{aligned}$$

Thus,

$$\frac{\|\mathcal{U}_H^{\mathcal{C}}\|}{\|\mathcal{U}_H\|} \leq \kappa \leq 1.$$

By Theorem 15, term continuity of $\mathbf{G}$ says that the quotient on the left converges to 1, and therefore κ must be 1. □

Lemma 20. *For all* $H \in \mathbb{N}$ *and* $k > 1$, *we have*

$$\frac{f_{H,H}}{f_H} > \frac{k}{k+1}.$$

Proof. It is easy to verify this conclusion separately for $H = 1, 2$, so assume $H > 2$. Applying the recursive description Lemma 17(i) of $f_{H,h}$, we have

$$\begin{aligned} f_{H,2} &= 2t_{H-1}f_{H,1} = 2t_{H-1}, \\ f_{H,3} &= 2t_{H-2}f_{H,2} = 2^2 t_{H-2}t_{H-1}, \\ f_{H,4} &= 2t_{H-3}f_{H,3} = 2^3 t_{H-3}t_{H-2}t_{H-1}. \end{aligned}$$

In general, this gives us, for $1 < h \leq H$ in $\mathbb{N}$,

$$f_{H,h} = 2^{h-1} t_{H-(h-1)} t_{H-(h-2)} t_{H-(h-3)} \cdots t_{H-2} t_{H-1}. \tag{7}$$

Using (3) and (7), we obtain

$$\begin{aligned} f_H - f_{H,H} &= f_{H,1} + f_{H,2} + f_{H,3} + \cdots + f_{H,H-1} \\ &= 1 + 2t_{H-1} + 2^2 t_{H-1}t_{H-2} + \cdots + 2^{H-2} t_{H-1}t_{H-2} \ldots t_3 t_2 \\ &< (1 + 2 + 2^2 \ldots + 2^{H-2}) t_{H-1}t_{H-2} \ldots t_3 t_2 \\ &< 2^{H-1} t_{H-1}t_{H-2} \ldots t_3 t_2 t_1 / t_1 = f_{H,H}/k. \end{aligned}$$

Thus,

$$f_H - f_{H,H} < \frac{f_{H,H}}{k} \Rightarrow f_H < f_{H,H}\left(\frac{k+1}{k}\right) \Rightarrow \frac{f_{H,H}}{f_H} > \frac{k}{k+1}$$

as required. □

Since most partial terms of height at most H are in $\mathcal{F}_{H,H}$ (Lemma 20) and almost all partial terms in $\mathcal{F}_{H,H}$ are constant (Lemma 19), it should follow that most partial terms of height at most H are constant. Our next theorem confirms a more precise formulation of this expectation.

Theorem 21 (Constant Term Theorem). *If* $1 < k \in \mathbb{N}$ *and the finite groupoid* $\mathbf{G}$ *is* k*-TC, then there is a number* λ *such that*

$$\frac{2}{3} \le \frac{k}{k+1} \le \lambda \le 1 \quad \text{and} \quad \lim_{H\to\infty} \frac{\|\mathcal{C}_H\|}{\|\mathcal{F}_H\|} = \lim_{H\to\infty} \frac{c_H}{f_H} = \lambda.$$

In particular, most partial k*-ary terms are constant.*

Proof. Since $1 < k$, Theorem 13 says that $q_1, q_2, q_3, \ldots$ converges to a number $\lambda \in [0,1]$. Let $\epsilon > 0$. By Lemma 19 there is a $J \in \mathbb{N}$ such that $H > J$ implies that

$$\frac{c_{H,H}}{f_{H,H}} > 1 - \frac{(k+1)\epsilon}{k}.$$

Applying Lemma 20, we have, for all $H > J$,

$$\frac{c_H}{f_H} \ge \frac{c_{H,H}}{f_H} = \left(\frac{c_{H,H}}{f_{H,H}}\right)\left(\frac{f_{H,H}}{f_H}\right) > \left(1 - \frac{(k+1)\epsilon}{k}\right)\left(\frac{k}{k+1}\right) = \frac{k}{k+1} - \epsilon.$$

Consequently the limit λ is at least $\frac{k}{k+1}$. □

7. Conclusion

Our results point to the conclusion that, in general,

> *a BEA will successfully solve the TGP for* k*-ary targets if the groupoid is idem-primal and* k*-TC with* $k > 1$.

To recapitulate, consider a run of any BEA with a finite groupoid $\mathbf{G}$ and a k-ary target array T that has a solution as inputs. Theorem 5 tells us that the BEA will successfully find a solution to T if and only if it terminates. There are three possible outcomes of this run, two that fail and one that succeeds.

1. For some $H > 1$ it never finds b valid partial terms to construct Level H.
2. It constructs Level H for every $H \ge 1$ without solving any array in $\mathcal{T}$.
3. It produces a partial term in $\mathcal{T}'$ which it can complete to a solution to T with some term in M, returns that solution and terminates.

If **G** is idemprimal, then Theorem 6 tells us that, with probability one, (1) will not occur. If outcome (1) does not occur, then failure must be a result of outcome (2).

If **G** is both idemprimal and k-TC with $k > 1$, we have good reason to believe that (2) will not occur either. Theorem 21 tells us that, as H grows large, the proportion of partial k-ary terms that are constant will exceed each number less that 2/3 from some point on. Assuming that the partial terms generated by the BEA are in some minimal way representative of all partial terms, we might well expect that, at high enough levels of $\mathcal{T}'$, a constant partial term will soon appear. At that point Theorem 9 says that the first term of M that is tested will complete that partial term in $\mathcal{T}'$ to a solution to T. By Corollary 8 the same term of M will complete the corresponding partial term in $\mathcal{T}$ to a solution to its parent array, thereby preventing (2) from occurring.

If neither (1) nor (2) occur, we will be left with (3) as the expected outcome. This conclusion falls short of a theorem since we have no direct knowledge of the distribution of constant partial terms among the partial terms that are generated by a BEA. However, many experiments like the one presented in the next section suggest that our BEA is never impeded by either of the obstacles (1) or (2) when **G** is both idemprimal and k-TC, and that it normally is so impeded otherwise. Our theorems explain why that should be the case.

8. A Sample BEA

In this section, we will present a specific BEA, exhibit its performance on a randomly generated sample of 4-element groupoids to illustrate the results presented here, and state some questions and conjectures that arise from this work. In particular, we would like to see how success of the BEA correlates with the presence of idemprimality, NSRs and asymptotic completeness.

We will say that a finite groupoid **G** is **IPr** if it is idemprimal and that it has **NSR** if no subgroupoid of **G** has a separating relation. We will say that **G** is **asymptotically complete** (**AC**) if it is asymptotically k-complete for every positive integer k, and that it is **TC** if it is k-TC for every positive integer k. This terminology gives an immediate consequence of the Continuity Theorem of [3] cited in our introduction.

Corollary 22. *A finite groupoid is TC if it is AC and has NSR.*

The intent of [3] was to produce this result as a means to test a finite groupoid **G** for TC. An efficient algorithm to determine whether or not **G** has NSR is given in [3], but no conclusive test for AC is known. What is given in [3] instead is a recursive formula to calculate the probability distributions $\beta_{\vec{d}}(1), \beta_{\vec{d}}(2), \beta_{\vec{d}}(3), \ldots$ for a given choice of k and $\vec{d} \in G^k$. Experience suggests that, for every choice of k and $\vec{d} \in G^k$, a spreadsheet calculation of an initial segment of this sequence will produce a fairly convincing case that the probabilities are bounded away from zero or that they are not. Assuming we accept these conclusions, we can, for any positive integer k, test each $\vec{d} \in G^k$ to get a good sense as to whether or not $\mathbf{G}$ is k-AC. But we can do this with only finitely many choices of k, leaving us with no convincing basis to conclude that $\mathbf{G}$ is AC in general.

An efficient solution to this problem, first conjectured in [3], remains unproven and is still supported by all of the spreadsheet runs we have done to date.

Conjecture 23. *Let* $\mathbf{G}$ *be a finite groupoid and let* k *and* k' *be positive integers. Assume that the coordinates of* $\vec{d} \in G^k$ *and* $\vec{d'} \in G^{k'}$ *generate the same subgroupoid of* $\mathbf{G}$. *If* a *is in that subgroupoid, then* $\beta_{\vec{d},a}$ *is eventually bounded away from zero if and only if* $\beta_{\vec{d'},a}$ *is eventually bounded away from zero.*

For our work in this section, we will tentatively assume that this conjecture is true, and that the spreadsheet runs we do accurately predict the presence or absence of AC. In that case, we can determine if $\mathbf{G}$ is or is not AC by testing only one choice of $\vec{d}$ for each non-trivial subgroupoid of $\mathbf{G}$.

In order to present a particular BEA, we will take $k = 3$ and consider the TGP for ternary operations with variables x, y, z. We must specify choices for the five unspecified parameters listed in Sec. 3 after the schema for BEAs. Doing multiple informal experiments with different choices of these parameters, we found considerable variation in the efficiency with which each of these choices found terms for 4-element groupoids. Ultimately we settled on the following choices as giving the best performance we could obtain.

(1) $h = 4$,
(2) $b = 1$,
(3) the Grow algorithm of [7] as the Valid Term Generator,
(4) produce a single valid term without pruning and
(5) $M = \mathcal{T}m_2$, the set of 12 terms $\{x, y, z, xx, xy, xz, yx, yy, yz, zx, zy, zz\}$.

Our experience was that, without explicit parallel processing, choosing more than $b = 1$ terms at each level, or producing more than $b = 1$ and pruning down to $b = 1$, took more time to find solutions.

We have seen that, in the presence of TC and IPr, the eventual success of a BEA does not depend on the choice of the test set M. But the choice of M does affect the total run time. A larger set M makes it likely to find a solution in M at a lower level of the tree $\mathcal{T}$ because there are more test cases. But a smaller set M makes it quicker to get to higher levels because fewer tests need to be done. Our choice of $M = \mathcal{T}m_2$ appeared to be a good balance between these two constraints.

As a typical application of this BEA, we applied it to a search for a discriminator term for the groupoid $\mathbf{D}_1$ presented in the introduction. It easily passed the NSR test. As it is primal, it is IPr and has no proper sub-groupoids. Thus, only one choice of $\vec{d}$ was needed to confirm that it is AC. Thus, our results predicted that it would yield terms under the BEA. The outcome of this run was that the BEA delivered the discriminator term below for $\mathbf{D}_1$ with 492 variable occurrences in 1.3 s.

$$
\begin{aligned}
t(x,y,z) = {} & (((((x * y) * x) * x) * (((x * y) * x) * ((z * x) * (y * x)))) * (((x * y) \\
& *(((x * x) * x) * x)) * (((((y * x) * (y * x)) * x) * ((y * x) * y)) * (((y * x) \\
& * ((x * y) * (y * (y * x)))) * ((z * (z * ((x * z)z))) * (((x * (z * x)) \\
& * ((x * (z * x)) * x)) * (((z * x) * (((z * x) * x) * (x * (z * z)))) \\
& * (((z * x) * (z * x)) * ((((x * z) * z) * ((z * z) * z)) * (((((y * z) * z) * x) \\
& * (((z * x) * (y * y)) * x))(((z * (z * x)) * ((z * (z * x)) * x)) * ((x * x) \\
& * ((x * x) * ((x * x) * ((x * x) * ((x * x) * ((x * x) * ((x * x) \\
& * ((x * x) * ((x * x) * ((x * x) * ((x * x) * (((x * (x * (y * y))) * (x * x)) \\
& * ((x * x) * ((x * x)((x * x) * ((x * x) * ((x * x) * ((x * x) * ((x * x) \\
& * ((x * x) * ((x * x) * ((x * x) * ((x * x) * ((x * x) * ((x * x) * ((x * x) \\
& * (((z * z) * (((y * x) * y) * z)) * ((y * ((z * y) * (y * y))) * ((y * (y \\
& * ((z * z)(y * y)))) * ((z * (z * ((z * x) * (y * x)))) * (((((x * z) \\
& * (z * y)) * y) * ((y * (z * y)) * ((z * y) * (y * y)))) * ((z * (z * ((x * x) \\
& * z))) * ((y * y) * ((y * y) * ((y * y) * ((y * y) * ((y * y) * ((y * (x * ((y * x) \\
& \times (y * y)))) * ((x * x) * ((x * x) * ((x * x) * ((x * x) * ((x * (x * x)) \\
& * (((x * ((x * z) * y)) * (x * ((x * z) * (z * x)))) * (((x * (z * (x * z))) \\
& * (z * (x * (x * z)))) * ((z * z) * ((z * z) * ((z * z) * ((z * z)((z * z) \\
& * ((x * x) * ((x * x) * ((x * x) * ((x * x) * ((x * x) * ((x * x) * ((y * (y * y)) \\
& * (((x * y) * ((y * x) * (x * (x * y)))) * ((y * y) * ((z * (z * z)) \\
& * ((y * (y * y)) * ((y * y) * ((y * (y * y)) * (((y * z)(y * ((x * z) * (y * z)))) \\
& * ((z * z) * (((x * x) * (x * z)) * (((((x * y) * y) * x) * ((z * (y * z)) \\
& * (y * (x * x)))) * ((z * (z * ((x * x) * z))) * (((y * y) * (((z * x) * y) \\
& * (y * (y * z)))) * ((((z * y) * x)((x * y) * ((y * x) * (z * y)))) \\
& * ((y * (((x * y) * z) * (z * y))) * (((y * y) * ((y * y) * (y * y))) * ((y * y) \\
& * ((y * y) * ((y * y) * (((y * x) * (x * (z * (z * y)))) * ((x * x) * ((x * x) \\
& * (((y * ((y * y)(y * y))) * (y * (x * x))) * ((y * y) * ((x * x) * ((x * x) \\
& * ((x * (x * ((x * z) * z))) * (((y * x) * x) * ((x * x) * ((x * x)
\end{aligned}
$$

$$
\begin{aligned}
&* ((((y * (x * x)) * (x * (y * x))) * (z * ((y * z) * (y * x)))) * ((y * x)\\
&* ((x * x)((x * x) * ((x * x) * ((x * (x * ((x * x) * (z * z)))) * ((z * x)\\
&* ((x * x) * (((((y * ((y * y) * y)) * y) * ((y * (y * (x * z))) * ((((y * z)\\
&* ((z * z) * z)) * ((z * (x * y)) * z)) * (((z * z) * (z * ((y * x)(z * y))))\\
&* (((x * (y * (z * x))) * (((y * x) * (z * x)) * (z * x))) * (((((z * x) * y) * x)\\
&* (y * z)) * (((z * z) * (((((y * z) * x) * ((z * y) * x)) * (((y * x) * (y * x))\\
&* ((z * z) * x)))((x * ((y * (y * y)) * x)) * (((x * x) * (((x * y) * y)\\
&* (z * x))) * (((((((x * z) * y) * z) * (y * y)) * (y * z)) * (y * (x * (x * y))))\\
&* ((y * ((z * y) * (y * z))) * (((x * z) * (z * x))(y * (y * x)))))))))\\
&* (x * y))))))) * (yy * (y * ((y * (y * y)) * y))))))))))))))))))))))))))))))\\
&\quad))
\end{aligned}
$$

While we found this to be a rather large term by normal standards, we do consider it a sizable improvement over either a term with 10^{45} variable occurrences or a 10^{25}-year search.

We were not able to get this BEA to find terms for 5-element groupoids, which have much larger search spaces. We suspect that better performance might be achieved if multiple copies of the VTG could be run in parallel and a larger number of valid terms could be produced at each level and then efficiently but effectively pruned down to b of them.

Problem 2. Find different choices for the five unspecified BEA parameters that will result in a BEA that can find terms for larger groupoids as was done in [5].

In order to get a sense as to when this BEA is and is not successful, we ran it on some randomly chosen inputs. We began by getting 160 randomly chosen digits in $\{0, 1, 2, 3\}$ from ⟨*www.random.com*⟩, which draws on atmospheric noise. Putting those digits into ten blocks of 16 each generated tables for ten randomly chosen groupoids shown in Fig. 6. We chose $\mathbf{D}_1$ as the first of these ten.

We then proceeded to test the BEA on each of these groupoids. To do this, we needed a target operation on each groupoid that we knew in advance was a term operation. If a groupoid does not have a discriminator term, it is in general difficult to determine which operations are term operations. In order to find a target term operation for each of these groupoids, we used a randomizing algorithm to generate a term with 100 variable occurrences for each one. We then calculated the values of that term at each triple, and used those values as our target operation for that groupoid. We did five runs of the BEA with each of the ten groupoids on its target operation, a total of fifty runs. A run was timed out if it did not terminate with a

$*$	0	1	2	3
0	1	3	1	2
1	1	2	2	3
2	2	0	0	3
3	2	2	1	2

$\mathbf{D}_1$

$*$	0	1	2	3
0	1	0	0	1
1	0	1	1	0
2	2	1	3	2
3	1	2	1	0

$\mathbf{D}_2$

$*$	0	1	2	3
0	2	1	1	3
1	3	0	2	1
2	1	3	2	0
3	2	2	1	2

$\mathbf{D}_3$

$*$	0	1	2	3
0	2	2	0	2
1	0	0	0	1
2	1	2	0	2
3	0	0	2	1

$\mathbf{D}_4$

$*$	0	1	2	3
0	2	3	3	3
1	2	2	2	2
2	0	0	0	3
3	3	2	2	0

$\mathbf{D}_5$

$*$	0	1	2	3
0	3	0	0	3
1	2	2	1	0
2	3	0	2	0
3	2	0	1	3

$\mathbf{D}_6$

$*$	0	1	2	3
0	2	2	3	3
1	0	0	0	2
2	1	1	2	1
3	0	1	0	3

$\mathbf{D}_7$

$*$	0	1	2	3
0	2	3	3	0
1	0	3	1	0
2	1	0	3	1
3	3	3	2	2

$\mathbf{D}_8$

$*$	0	1	2	3
0	0	2	2	3
1	0	3	2	0
2	3	2	0	0
3	3	2	0	3

$\mathbf{D}_9$

$*$	0	1	2	3
0	1	1	1	1
1	2	1	0	0
2	3	3	3	2
3	2	2	1	0

$\mathbf{D}_{10}$

Fig. 6. Ten random 4-element groupoids.

Table 1. BEA run times in seconds.

	Run 1	Run 2	Run 3	Run 4	Run 5	
$t/o =$	10 min	10 min	10 min	10 min	1 h	Mean
$\mathbf{D}_1$	23.0	6.2	5.4	51.3	13.6	19.9
$\mathbf{D}_2$	t/o	t/o	t/o	t/o	t/o	
$\mathbf{D}_3$	2.4	2.0	1.6	2.1	1.8	2.0
$\mathbf{D}_4$	t/o	t/o	t/o	t/o	t/o	
$\mathbf{D}_5$	t/o	t/o	t/o	t/o	t/o	
$\mathbf{D}_6$	3.5	9.2	5.2	10.8	10.2	7.8
$\mathbf{D}_7$	0.9	0.9	0.9	1.0	1.2	1.0
$\mathbf{D}_8$	1.2	3.0	1.7	3.9	4.1	2.8
$\mathbf{D}_9$	t/o	t/o	t/o	t/o	t/o	
$\mathbf{D}_{10}$	1.4	2.6	11.8	1.0	9.7	5.3
Mean						6.5

solution after ten minutes. If four runs timed out at ten minutes, we let the fifth run go for sixty minutes before timing out. The results of these runs are shown in Table 1.

What we found was that the ten groupoids divided cleanly between six that terminated on every run in less than a minute $(\mathbf{D}_1, \mathbf{D}_3, \mathbf{D}_6, \mathbf{D}_7, \mathbf{D}_8, \mathbf{D}_{10})$, with an overall mean run time of 6.5 s, and four that timed out on every single run $(\mathbf{D}_2, \mathbf{D}_4, \mathbf{D}_5, \mathbf{D}_9)$.

Our final step was to examine the ten groupoids to see, for each one, which of the conditions NSR, AC and IPr it satisfied.

NSR. Using the algorithm given in [3] we quickly verified that these groupoids all have NSR with the exception of $\mathbf{D}_2$. The table for $\mathbf{D}_2$ shows that $\{0, 1\}$ forms a subgroupoid with $\sigma = \{(0, 1), (1, 0)\}$ as a separating relation. These facts are recorded in the second column of Table 2.

Table 2. Success (+) or failure (−) of the BEA and properties of groupoids.

	BEA	NSR	AC	IPr	(Primal)	Proper subgroupoids
$\mathbf{D}_1$	+	∗	∗	∗	∗	
$\mathbf{D}_2$	−					$\{1\}, \{0, 1\}$
$\mathbf{D}_3$	+	∗	∗	∗		$\{2\}$
$\mathbf{D}_4$	−	∗				$\{0, 1, 2\}$
$\mathbf{D}_5$	−	∗				$\{0, 2, 3\}$
$\mathbf{D}_6$	+	∗	∗	∗		$\{2\}, \{3\}$
$\mathbf{D}_7$	+	∗	∗	∗		$\{2\}, \{3\}$
$\mathbf{D}_8$	+	∗	∗	∗	∗	
$\mathbf{D}_9$	−	∗				$\{0\}$, $\{3\}$, $\{0, 3\}$, $\{0, 2, 3\}$
$\mathbf{D}_{10}$	+	∗	∗	∗		$\{1\}$

AC. Using Conjecture 23 to judge AC, we took $\vec{d} := (3, 0, 2, 3, 1, 1, 2, 3, 2, 3)$, with coordinates generating each groupoid itself, and ran the spreadsheets until the values appeared to form a stable pattern to six decimal places. This led us to conclude that $\mathbf{D}_2, \mathbf{D}_4, \mathbf{D}_5$ and $\mathbf{D}_9$ are not AC because, for some element a, the sequence $\beta_{\vec{d},a}$ converges to zero. Each of the other six has no non-trivial proper subgroupoids, so Conjecture 23 required that we only examine this one choice of $\vec{d}$. In contrast to the first four, these groupoids each reached a stable repeating pattern of all non-zero values in less than 32 iterations. We therefore recorded them as being AC in the third column of Table 2.

IPr. Judging IPr posed a different kind of challenge. The definition of "idemprimal" offers no practical way to tell whether or not a finite groupoid is idemprimal. But Pixley [10] and Werner [14] found an important characterization of idemprimality that proved helpful.

Pixley/Werner Theorem. *A non-trivial finite algebra is idemprimal if and only if it has no non-trivial congruences, automorphisms or proper subalgebras and has the ternary discriminator as a term operation.*

The four groupoids that are not AC, $\mathbf{D}_2, \mathbf{D}_4, \mathbf{D}_5, \mathbf{D}_9$, are not IPr because they each harbor a non-trivial proper subgroupoid, as indicated in the last column of Table 2. Of the remaining six, $\mathbf{D}_1$ and $\mathbf{D}_8$ are easily checked to be primal by Rousseau's Theorem [12], and are therefore idemprimal. The other four, $\mathbf{D}_3, \mathbf{D}_6, \mathbf{D}_7, \mathbf{D}_{10}$, have no non-trivial congruences, automorphisms or proper subalgebras. But they are not primal because they each have at least one idempotent, listed as a one element subgroupoid in Table 2.

It remained to determine whether or not $\mathbf{D}_3, \mathbf{D}_6, \mathbf{D}_7$ and $\mathbf{D}_{10}$ are idemprimal. By the Pixley/Werner Theorem, this meant determining whether or not they have discriminator terms. If one of them did not have a discriminator term, we would have had no efficient means of establishing that fact. However, if one did have a discriminator term, then Theorems 6, 9 and 21 said that our BEA should eventually be able to find it. As an *application* of our BEA, we ran it on each of these four groupoids with the ternary discriminator operation as target. The good news is that it did successfully find a discriminator term for each one in the following times.

$$\mathbf{D}_3 : 2.2\,\mathrm{s} \quad \mathbf{D}_6 : 16.0\,\mathrm{s} \quad \mathbf{D}_7 : 29.6\,\mathrm{s} \quad \mathbf{D}_{10} : 77.3\,\mathrm{s}.$$

We have therefore listed these four as IPr in the fourth column of Table 2.

The first column of Table 2 summarizes the data of Table 1 by indicating which groupoids successfully yielded terms under our BEA. Comparing these results with the next three columns of Table 2, we are presented with a striking correlation.

Experimental Results. *The six groupoids for which terms were efficiently found for random targets each satisfy the three conditions NSR, AC and IPr, while the four that timed out on all runs each fail at least one of those conditions.*

While this is not a statistically significant experiment, we present it as an illustration of the correlation between success of a BEA and satisfaction of NSR, AC and IPr predicted in Sec. 7.

The significance of the conclusion of our work given in Sec. 7 depends on the frequency that finite groupoids satisfy NSR, AC and IPr. Note that the IPr groupoids in Table 2 are exactly the ones that satisfy both NSR and AC, and are therefore TC. This is not generally the case, as [5, Table 6] gives examples of 5-element groupoids showing that no two of the properties NSR, AC and IPr imply the third. But [5, Table 5] shows that, of ten random 5-element groupoids, ten have NSR, eight are AC and six are IPr. Altogether these data suggest that NSR and AC are at least as common as IPr. In view of Murskiĭ's Theorem, this observation lends support for two conjectures implicit in Clark [3], Question 34.

Conjecture 24. *The proportion of groupoids* $\mathbf{G} := \langle\{0, 1, \ldots, n-1\}, *\rangle$ *that have NSRs approaches one as* n *approaches infinity.*

Conjecture 25. *The proportion of groupoids* $\mathbf{G} := \langle\{0, 1, \ldots, n-1\}, *\rangle$ *that are AC approaches one as* n *approaches infinity.*

If these two conjectures are true, Corollary 22 would give us an immediate consequence.

Conjecture 26. *The proportion of groupoids* $\mathbf{G} := \langle\{0, 1, \ldots, n-1\}, *\rangle$ *that are TC approaches one as* n *approaches infinity.*

If this conjecture were true, our work here would indicate that the proportion of n-element groupoids that efficiently produce terms under a BEA will approach one as n approaches infinity.

Acknowledgment

The authors wish to express their gratitude to the anonymous referee whose very thorough reviews of our papers led to many valuable improvements.

References

[1] C. Bergman, *Universal Algebra: Fundamentals and Selected Topics* (CRC Press, 2012).

[2] J. Berman and S. Burris, A computer study of 3-element groupoids, *Logic and Algebra Proc. Magari Conf. Held in Siena*, Italy, ed. A. Ursini and P. Aglian (Marcel Dekker, 1996), pp. 379–430.

[3] D. Clark, Evolution of algebraic terms 1: Term to term operation continuity, *Int. J. Algebra Comput.* **23**(5) (2013) 1175–1205.

[4] D. Clark, B. Davey, J. Pitkethly and D. Rifqui, Flat unars: The primal, the semi-primal and the dualisable, *Algebra Universalis* **63**(4) (2010) 303–329.

[5] D. Clark, M. Keijzer and L. Spector, Evolution of algebraic terms 2: Deep drilling algorithm, *Int. J. Algebra Comput.* **26**(6) (2016) 1141–1176.

[6] R. Freese, E. Kiss and M. Valeriote, UACalc, a Universal Algebra Calculator, Available at: www.uacalc.org (2011).

[7] J. Kosa, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, 1992).

[8] V. L. Murskiĭ, Konéčnaá baziruémost′ toźdéstv i drugié svojstva "počti vséh" konéčnyh algébr (A finite basis of identities and other properties of "almost all" finite algebras), *Problémy Kibérnétiki* **30** (1975) 43–56.

[9] P. Norvig, *Paradigms of Artificial Intelligence Programming: Case Studies in Common LISP* (Morgan Kaufmann, 1992).

[10] A. Pixley, Functionally complete algebras generating distributive and permutable classes, *Math. Z.* **114** (1970) 361–372.

[11] I. Rosenberg, Über die funktionale Vollständigkeit in den mehrwertigen Logiken (Struktur der Funktionen von mehreren Veränderlichen auf endlichen Mengen), *Rozpravy Československé Akad. Věd Řada Mat. Přírod.* **80** (1970) 1–93.

[12] G. Rousseau, Completeness in finite algebras with a single operation, *Proc. Am. Math. Soc.* **18** (1967) 1009–1013.

[13] L. Spector, D. Clark, B. Barr, J. Klein and I. Lindsay, Genetic programming for finite algebras, *Genetic and Evolutionary Computation Conf.* (*GECCO*) 2008 *Proc.*, Atlanta GA (July 2008), Editor-in-Chief Maarten Keijzer, Association for Computing Machinery (ACM), pp. 1291–1298. [Paper won first place in the GECCO 2008 Human Competitive competition.] ⟨www.genetic-programming.org/hc2011/combined.html⟩.

[14] H. Werner, Eine Charakterisierung funktional vollständiger Algebren, *Arch. Math.* (Basel) **21** (1970) 381–385.