Chapter 12
Strong Hom-Associativity

Lars Hellström

Abstract The concept of an algebra being hom-associative is examined, and found to allow some awkward complications. A modified concept of strong hom-associativity is introduced to eliminate those quirks. It is proved that the basic "Yau twist" construction of a hom-associative algebra from an associative algebra does in fact produce strongly hom-associative algebras. It is proved that the axioms for a strongly hom-associative algebra yields a confluent rewrite system, and a basis for the free strongly hom-associative algebra is given a finite presentation through a parsing expression grammar.

Keywords Hom-associative algebra · Strongly hom-associative algebra · Rewrite system

MSC 2010 Classification: 17A30 · 16S15 · 68Q42

12.1 Introduction

A *hom-algebra* is an algebra which in addition to the bilinear multiplication operation μ has a unary linear operation α, known as the "hom" because in some seminal applications it was a homomorphism. Imposing some condition along the line of making α a homomorphism is common in studies of hom-algebras, but that is not a specialisation taken in this paper; α is only required to be linear.

The first class of hom-algebras to be defined were the *hom-Lie algebras* [4], where the Jacobi identity is deformed by applying α to the factor that only goes through the bracket once. Later [9] the class of *hom-associative* algebras were defined to establish a hom-analogue of the relation between ordinary associative and Lie algebras: the commutator of a (hom-)associative algebra is automatically the bracket of a

L. Hellström (✉)
Division of Applied Mathematics, School of Education, Culture and Communication, Mälardalen University, Box 883, 72123 Västerås, Sweden
e-mail: lars.hellstrom@mdh.se

S. Silvestrov et al. (eds.), *Algebraic Structures and Applications*,
Springer Proceedings in Mathematics & Statistics 317,
https://doi.org/10.1007/978-3-030-41850-2_12

(hom-)Lie algebra. However to date this correspondence remains incomplete, in that it is not known whether every hom-Lie algebra arises as the commutator algebra of a hom-associative algebra; it is straightforward to hom-ify the construction of the enveloping algebra of a Lie algebra [12], but so far we lack a hom-counterpart of the Poincaré–Birkhoff–Witt (PBW) theorem to tell us what that enveloping algebra looks like, and therefore cannot be sure that applying the commutator construction to the enveloping hom-associative algebra will give us (a superalgebra of) the original hom-Lie algebra.

Part of the problem here is that we lack a combinatorial description of the *free* hom-associative algebra. In the classical case, associative products can be modelled as words, where the enveloping algebra relations then allow us to swap letters, resulting in the PBW conclusion that the enveloping algebra as a vector space is isomorphic to the commutative algebra on the same number of variables. Likewise the Lyndon words [8, 10] provide a combinatorial model for the free Lie algebra, which is essential for the Shirshov [11] analogue of Gröbner basis theory for Lie algebras. The free hom-associative algebra is trickier, and although there are results for special cases [3, 13], the general case remains open; we know that the structure of parenthetisation matters quite a lot, so a basis for the free hom-associative algebra does not consist of flat words as in the associative case, but on the other hand it is not the arbitrary binary trees of a free magma either.

The reason to not assume anything except linearity of the hom α in what follows is twofold. The first is generality: whereas making α an algebra homomorphism is a very elegant way of defining it in the context of enveloping algebras of hom-Lie algebras—the hom-Lie algebra providing a definition of α on the generators, and the homomorphism property then extending α to the whole hom-associative algebra—it is not *necessarily* the appropriate way of defining it. It *could* happen that α, which arguably is a kind of deformation, requires corrections to multiplicativity at higher degrees for everything to fit together; at present we do not know. The second reason is the combinatorial side of the matter: the distinct pattern (demonstrated in the next section) that appears in the rules derivable from hom-associativity alone, is lost when additionally imposing multiplicativity of α. It therefore makes sense to first explore the combinatorial structure without imposing multiplicativity, and only later (if needed) consider the effects of multiplicativity.

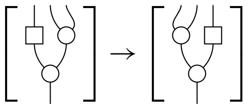
Section 12.2 of this chapter defines a strengthening of hom-associativity—the titular *strong* hom-associativity—and explains why this strengthening is probably rather mild, even if it is voluminous on the axiomatic side. Section 12.3 proves that strong hom-associativity gives rise to a confluent rewrite system (the general concept whose specialisation to commutative polynomials is classical Gröbner basis theory). Finally Sect. 12.4 gives a combinatorial model, in the form of a parsing expression grammar, for the free hom-associative algebra.

12.2 Canyons and Identities

A systematic way of seeking a combinatorial model for the free algebra modulo some set of axioms is to raise the level of abstraction one notch, and do rewriting in the free operad, taking (an orientation of) those axioms as one's initial rewrite system. Using traditional function notation for elements of the free operad is somewhat cumbersome, since even such a familiar rule as associativity ends up as as a relation

$$\bigl(\lambda x, y, z \colon \mu(x, \mu(y, z))\bigr) \rightarrow \bigl(\lambda x, y, z \colon \mu(\mu(x, y), z)\bigr)$$

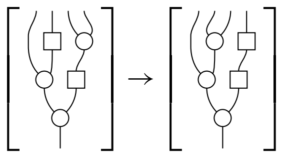
of two lambda-terms. Far more transparent is to use the graphical (diagrammatic, string diagram) *network notation*, where the operations in an expression are vertices in an (open) graph, edges show how operations are composed, inputs are edges from the top boundary of the network, and the output (result) is the edge to the bottom boundary. Since these expressions are operadic, the graphs will all be trees. In this notation, the rewrite rule for hom-associativity becomes

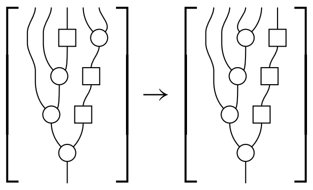
 (12.1)

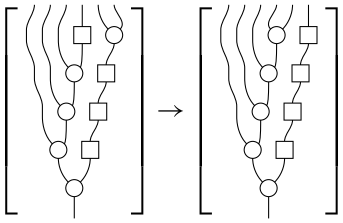
where round vertices denote the multiplication μ and square vertices denote the hom-operation α. This rule then states that every occurrence of the left hand side (some μ with an α as left operand and another μ as right operand) as a subexpression may be replaced by the right hand side, with the understanding that doing so produces an equivalent expression, since we wish to explore the equational theory where these left and right hand sides in fact are equal.

The way that one does this is to *complete* the rewrite system generated by one's axioms, by adding new rules whenever an ambiguity fails to resolve. The main theorem in commutative Gröbner basis theory is that this process always ends (although it may have to run for a very large number of steps), but beyond commutative algebra we *may* run into problems. Two operatic cases where everything still works out are those of ordinary associativity and the Leibniz identity—see [5] for a precise statement—but hom-associativity turns out to be more complicated. Indeed, it is apparent from the calculations in [5] that the completion of (12.1) is going to be infinite. This means we cannot arrive at a description of free hom-associative algebras simply by performing mechanical calculations, but we may still be able to find one by analysing (and then predicting) the rules that the completion procedure produces. The rules all turn out to have a number of features in common.

The most apparent thing about the rules computed in [5] is that only two vertices differ between left and right hand side: an α and a μ have switched places. These two vertices always sit next to each other, but on opposite sides of a "canyon" whose height is not bounded. For example, there are rules

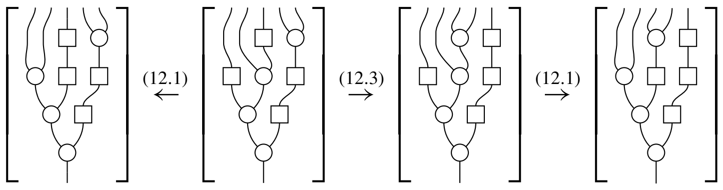

(12.2)


(12.3)


(12.4)

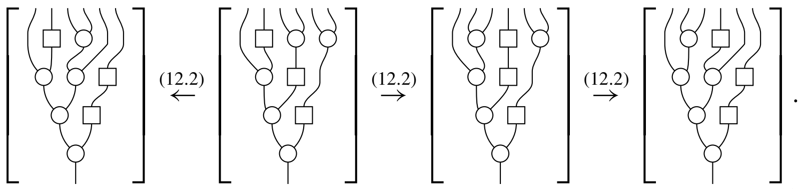
and this family continues with ever deeper canyons; to derive the next rule in this sequence, one only need to consider overlapping the bottom multiplication in the canyon by the top multiplication in the axiom rule (12.1) (or vice versa overlapping the bottom multiplication in the hom-associativity axiom with the multiplication on top of the right canyon wall—both yield the same result). Though infinite, this $\mu \otimes \alpha$ family (left wall always μ, right wall always α) of canyon identities is easily comprehensible, so if this had been all there was then a combinatorial model of the free hom-associative algebra would not have been too hard to construct. However, the hom-associative axiom admits self-interactions that are far more intricate than those shown so far.

For one, (12.1) has an overlap with the left wall of (12.3), leading to the rewrites



; (12.5)

the canyon in the rule constructed from this failed resolution has height 2, and whereas the bottom layer is again $\mu \otimes \alpha$, the top layer is here $\alpha \otimes \alpha$: both walls sport α as operation. It is equally possible to get μ in both walls: rule (12.2) has a nontrivial overlap with itself, leading to the failed resolution



(12.6)

Further calculations suggest that all canyon compositions arise: any height (number of layers between the μ at the bottom and the α–μ pair being exchanged at the top) is possible, and each layer may independently consist of two μs, two αs, or one of each. This is still comprehensible, but not as easy as having just one rule for each height.

What really makes things complicated is however that the expressions in rules (12.5) and (12.6) contain a bit more than just the canyon cores; there is also some surrounding padding, and not the same padding in (12.5) as in (12.6). Because of the way rewrite system completion works, and thanks to the rewrite rules in question all preserving the number of operations in the expressions, *this padding is provably necessary for the α–μ exchange across the canyon to be a logical consequence of the hom-associativity axiom* (12.1); without it, the left and right hand sides can be distinct.

Example 12.1 Let $(\mathcal{F}, \mu, \alpha)$ be the free hom-associative $\mathbb{Q}$-algebra generated by one element x. Then

$$\delta = \mu\Bigl(\mu\bigl(x, \alpha(x)\bigr), \mu\bigl(\mu(x, x), x\bigr)\Bigr) - \mu\Bigl(\mu\bigl(x, \mu(x, x)\bigr), \mu\bigl(\alpha(x), x\bigr)\Bigr) \tag{12.7}$$

is nonzero, because even though the two terms in δ are structured like the left and right hand side canyons of (12.6), they are missing the padding which would make equality a consequence of hom-associativity. On the other hand

$$\mu\bigl(\delta, \alpha(\alpha(y))\bigr) = 0 \qquad \text{for all } y \in \mathcal{F}, \tag{12.8}$$

so at the very least this δ is a zero-divisor in the free hom-associative algebra.

The conclusion (12.8) continues to hold if generalising δ to $\delta(x_1, x_2, x_3, x_4, x_5)$ where the five x factors in δ are allowed to be independent variables (although the same five in both terms), but then there might potentially be choices of $x_1, x_2, x_3, x_4, x_5 \in \mathcal{F}$ which make $\delta(x_1, x_2, x_3, x_4, x_5) = 0$. The freeness of $\mathcal{F}$ ensures $\delta \neq 0$ when one picks all of them to be the algebra generator.

Though the expressions involved are more complicated, this situation has similarities to one that arises with the definition of anti-commutativity. Recall that the naive definition of this for a bracket operation $[\cdot, \cdot]$ is that $[x, y] = -[y, x]$ for all x and y, but for squaring this only implies $2[x, x] = 0$, which in even positive characteristic is noticeably weaker than $[x, x] = 0$. Where the distinction is relevant, $[x, x] = 0$ is therefore called *strong* anticommutativity, whereas $[x, y] = -[y, x]$ is designated *weak* anticommutativity. Example 12.1 suggests something similar may be appropriate for hom-associativity.

Definition 12.1 Let $(\mathcal{A}, \mu, \alpha)$ be a hom-algebra. Taking the lift of composition $\circ$ to sets to have greater binding strength than union $\cup$, one may inductively define the family $\{C_{k,l,n}\}_{k,l\in\mathbb{Z},n\in\mathbb{N}}$ of maps $\mathcal{A}^{\otimes(k+2+l)} \longrightarrow \mathcal{A}$ (canyons graded by side-arities and height) by

$$\begin{aligned}
C_{k,l,0} &= \begin{cases} \{\mu\} & \text{if } k = l = 0,\\ \varnothing & \text{otherwise,}\end{cases}\\
C_{k,l,n+1} &= C_{k,l,n} \circ \{\mathrm{id}^{\otimes k} \otimes \alpha \otimes \alpha \otimes \mathrm{id}^{\otimes l}\}\\
&\quad \cup C_{k-1,l,n} \circ \{\mathrm{id}^{\otimes(k-1)} \otimes \mu \otimes \alpha \otimes \mathrm{id}^{\otimes l}\}\\
&\quad \cup C_{k,l-1,n} \circ \{\mathrm{id}^{\otimes k} \otimes \alpha \otimes \mu \otimes \mathrm{id}^{\otimes(l-1)}\}\\
&\quad \cup C_{k-1,l-1,n} \circ \{\mathrm{id}^{\otimes(k-1)} \otimes \mu \otimes \mu \otimes \mathrm{id}^{\otimes(l-1)}\}
\end{aligned}$$

for all $k, l \in \mathbb{Z}$ and $n \in \mathbb{N}$. Now the algebra $(\mathcal{A}, \mu, \alpha)$ is said to be **strongly hom-associative** if

$$c \circ (\mathrm{id}^{\otimes k} \otimes \alpha \otimes \mu \otimes \mathrm{id}^{\otimes l}) = c \circ (\mathrm{id}^{\otimes k} \otimes \mu \otimes \alpha \otimes \mathrm{id}^{\otimes l}) \quad \text{for all } c \in C_{k,l,n} \text{ and } k, l, n \in \mathbb{N}. \tag{12.9}$$

An algebra is said to be **(weakly) hom-associative** if it satisfies the $k = l = n = 0$ case of (12.9).

The analogy with weak and strong anticommutativity does not extend to the point that it is all-or-nothing depending on the characteristic (weak anticommutativity being a null condition in characteristic 2, but equivalent to strong anticommutativity in odd or zero characteristic); it is more like anticommutativity in (say) characteristic 8, where the weak form says an expression like $[x, x]$ or δ respectively is pretty special, even if it does not have to be 0. In the free strongly hom-associative algebra, the δ of (12.7) is zero by the $k = l = n = 1$ case of (12.9).

Theorem 12.1 *Let $(\mathcal{A}, \cdot)$ be an associative algebra, let $\alpha\colon \mathcal{A} \longrightarrow \mathcal{A}$ be a homomorphism of that algebra, and let $\mu\colon \mathcal{A} \times \mathcal{A} \longrightarrow \mathcal{A}$ be defined by $\mu(x, y) = \alpha(x \cdot y)$ for all $x, y \in \mathcal{A}$. Then $(\mathcal{A}, \mu, \alpha)$ (the "Yau twist" of $(\mathcal{A}, \cdot)$ by α) is a strongly hom-associative algebra.*

Proof Since $\alpha(x \cdot y) = \alpha(x) \cdot \alpha(y)$ for all $x, y \in \mathcal{A}$, any canyon identity

$$c \circ \mathrm{id}^{\otimes k} \otimes \alpha \otimes \mu \otimes \mathrm{id}^{\otimes l} = c \circ \mathrm{id}^{\otimes k} \otimes \mu \otimes \alpha \otimes \mathrm{id}^{\otimes l}$$

can be rewritten to the form $M_1 \circ A_1 = M_2 \circ A_2$, where the M_i parts are compositions of the associative multiplication $\cdot$ and the A_i parts are (tensor products of) compositions of α. Since the left–right order of factors are preserved during this, M_1

and M_2 end up as the same associative product of $k+3+l$ factors. Likewise A_1 and A_2 both have the form $\bigotimes_{i=1}^{k+3+l} \alpha^{\circ m_i}$, where each power m_i is the number of vertices on the path from the ith input to the output; since these are the same on both sides of the equality, $A_1 = A_2$ as well.

Here it is notable that this result hinges upon *both sides of the canyon having the same height*, because there is one path that goes along the right wall in the left hand side of the equality but along the left wall in the right hand side; if those walls are of unequal height, values passed along this path will be subject to an unequal number of operations. One could more generally speak about a "rift" if there are two walls of unequal height being joined by a μ multiplication at the bottom, and define "rift identities" as having the form

$$f \circ (\mathrm{id}^{\otimes k} \otimes \alpha \otimes \mu \otimes \mathrm{id}^{\otimes l}) = f \circ (\mathrm{id}^{\otimes k} \otimes \mu \otimes \alpha \otimes \mathrm{id}^{\otimes l})$$

for some rift f. Taking $\mathcal{A} = \mathbb{Q}\langle X \rangle$ where $X = \{x_n\}_{n\in\mathbb{N}}$ and letting the homomorphism α be defined by $\alpha(x_n) = x_{n+1}$, we will however through the same twist construction arrive at a hom-algebra in which all non-canyon "rift identities" are *false*. Having walls of equal height is thus a key feature of the hom-associativity canyons.

The further exploration of the realm between weak and strong hom-associativity is the subject of a research collaboration between the author, Abdenacer Makhlouf, Sergei Silvestrov, and possibly others. Preliminary studies there include tabulating the canyon cores of rewrite rules in the completion of the (weak) hom-associativity axiom, and what paddings enable those canyons. Another problem is to find examples other than the free one of (weakly) hom-associative algebras which are not strongly hom-associative.

Open Problem 1 *Find a finite-dimensional weakly hom-associative algebra which is not strongly hom-associative.*

Any hom-algebra which is not strongly hom-associative must fail some finitely large canyon identity, so this will be an example of an algebra satisfying one equational theory with finitely many axioms but not another. Then there should also be a finite(-dimensional) example.

Open Problem 2 *What is the minimal dimension of an algebra which is weakly hom-associative but not strongly hom-associative?*

12.3 Applying the Diamond Lemma

For the calculations (ambiguity resolutions) required for a proof by diamond lemma, it is convenient to have a more compact encoding of operad terms than the diagrammatic and product category ones. Apart from the basic nested parentheses notation, the most classical notation for general tree expressions is the *Polish notation*, where each symbol has a known arity and each term consists of a function symbol followed by the subterms that are the operands of that function with no intervening punctuation. If (as is appropriate for a hom-algebra) m has arity 2, a has arity 1, and x has arity 0 (is a constant), then $\mathsf{maaxmaxx}$ is Polish notation for what with parentheses and commas would be written $\mathsf{m}(\mathsf{a}(\mathsf{a}(\mathsf{x})), \mathsf{m}(\mathsf{a}(\mathsf{x}), \mathsf{x}))$. The use of m and a rather than μ and α here is because we now deal with formal terms rather than expressions in some generic algebra.

To deal with operadic terms, which may have inputs, a natural extension of classical Polish notation is to let an integer i denote the ith input; this was done in [5], and makes the action of permutations on the terms particularly easy to specify. The terms needed for (strong) hom-associativity are however all such that the inputs are used in order—first input 1, then input 2, and so on—which means the notation can be simplified: one may use a single symbol $\llcorner\!\lrcorner$ for all inputs, and let it depend on which occurrence of this symbol in the term that it is to determine which input it denotes. With that simplification, ordinary associativity of m is the equivalence $\mathsf{m}\llcorner\!\lrcorner\mathsf{m}\llcorner\!\lrcorner\llcorner\!\lrcorner \equiv \mathsf{mm}\llcorner\!\lrcorner\llcorner\!\lrcorner\llcorner\!\lrcorner$ (with numbered inputs $\mathsf{m1m23} \equiv \mathsf{mm123}$), whereas hom-associativity is $\mathsf{ma}\llcorner\!\lrcorner\mathsf{m}\llcorner\!\lrcorner\llcorner\!\lrcorner \equiv \mathsf{mm}\llcorner\!\lrcorner\llcorner\!\lrcorner\mathsf{a}\llcorner\!\lrcorner$ (with numbered inputs $\mathsf{ma1m23} \equiv \mathsf{mm12a3}$). This simplification is convenient when stating the canyon identities defining strong hom-associativity, because even though the left and right hand sides of the canyon identities only differ in how three sequential inputs are used, the exact numbers of these inputs are of little relevance; what should be emphasised is their positions relative to the canyon.

Definition 12.2 Recursively define the family $\{L_k\}_{k=0}^{\infty}$ of words by

$$L_0 = \{\mathsf{1}\} \quad \text{(the length 0 word)},$$
$$L_{k+1} = \{l\mathsf{a}, l\mathsf{m}\llcorner\!\lrcorner \mid l \in L_k\} \quad \text{for } k \in \mathbb{N}.$$

Analogously define the family $\{R_k\}_{k=0}^{\infty}$ of pairs of words by

$$R_0 = \bigl\{(\mathsf{1}, \mathsf{1})\bigr\},$$
$$R_{k+1} = \Bigl\{ (r\mathsf{a}, p), (r\mathsf{m}, \llcorner\!\lrcorner p) \Bigm| (r, p) \in R_k \Bigr\} \quad \text{for } k \in \mathbb{N}.$$

Then the rewrite system for strong hom-associativity is

$$S = \bigcup_{k\in\mathbb{N}} \Bigl\{ \mathsf{m}l\mathsf{a}\llcorner\!\lrcorner r\mathsf{m}\llcorner\!\lrcorner\llcorner\!\lrcorner p \rightarrow \mathsf{m}l\mathsf{m}\llcorner\!\lrcorner\llcorner\!\lrcorner r\mathsf{a}\llcorner\!\lrcorner p \Bigm| l \in L_k, (r, p) \in R_k \Bigr\}.$$

What happens here is that k is the height of the canyon, L_k is the set of all height k left walls of a canyon, whereas R_k is the set of all height k right walls of a canyon. Right walls are a bit more complicated to encode than left walls, because the operation at the top of the wall will in Polish notation come between the ms in the wall and their corresponding $\llcorner\!\lrcorner$s that are outside the canyon; therefore a right wall has one word r with the as and ms building up the wall, and another word p with the $\llcorner\!\lrcorner$s needed to complete the m subterms. The left wall words in L_k instead always continue along the last operand of a symbol, so there it is not necessary to separate $\llcorner$s from the other symbols.

12.3.1 Ambiguity Resolutions

The ambiguities of S all consist of two canyons with some overlap. Without loss of generality we can always denote by s_1 the rule that has the lowest m as its canyon bottom. This means there are four possibilities for the position of the m that is the bottom of the other rule s_2: it may sit in the left wall of the s_1 canyon, it may coincide with the bottom of the s_1 canyon, it may sit in the right wall of the s_1 canyon, or it may coincide with the m being moved by s_1 (sitting on top of the s_1 right canyon wall). These cases will now be treated in order.

As a general observation, if some subexpression is both part of a left canyon wall and a right canyon wall, then it must be a tower of as, which in Polish notation appears as a subword a^k for k being the height of the shared wall segment. In rewrite steps below, the redex (the operadic subexpression to be replaced) is outlined by placing parentheses between it and neighbouring non-redex subexpressions; if the parentheses are nested, then the redex is inside the outer parenthesis and outside the inner parenthesis. Note that subexpressions may appear disconnected in the Polish notation encoding, even though they are all connected in the term tree.

12.3.1.1 s_2 to the left of s_1

If s_2 has its bottom in the s_1 left canyon wall, then the s_2 canyon is to the left of the s_1 canyon, and there is some overlap between the right side of the s_2 canyon and the left side of the s_1 canyon. Depending on the extent of this overlap, there are three subcases: the m on top of the s_2 right canyon wall may be part of the s_1 left canyon wall, the a on top of the s_1 left cayon wall may be part of the s_2 right canyon wall, or it may be that neither is part of the other's canyon wall; in the last case, there is a third canyon between the two considered, whose bottom is at an even higher elevation than either of the two.

In the first subcase, the site of the ambiguity has the form

$$\mathsf{m} l_1 \mathsf{m} l_2 \mathsf{a}\llcorner\mathsf{a}^k \mathsf{m}\llcorner l_3 \mathsf{a}\llcorner r_1 r_4 r_2 r_5 r_3 \mathsf{m}\llcorner\llcorner p_3 p_5 p_2 p_4 p_1$$

where $k \in \mathbb{N}$ is the height of the s_2 canyon, $l_2 \in L_k$ is in s_2 matched to $(\mathsf{a}^k, \mathsf{1}) \in R_k$ whereas a^k in s_1 is matched to $(r_2, p_2) \in R_k$, (r_4, p_4), $(r_5, p_5) \in R_1$ are for the bottom and top respectively m in s_2, there is some $i \in \mathbb{N}$ such that $l_1 \in L_i$ and $(r_1, p_1) \in R_i$, and there is some $j \in \mathbb{N}$ such that $l_3 \in L_j$ and $(r_3, p_3) \in R_j$; the height of the s_1 canyon is then $i + 1 + k + 1 + j$. These ambiguities are resolved using a modification s_1' of s_1 that has left canyon wall $l_1 \mathsf{m}\llcorner\mathsf{a}^k \mathsf{a} l_3$ instead of $l_1 \mathsf{m}\llcorner\mathsf{a}^k \mathsf{m}\llcorner l_3$:

$$\begin{aligned}
&(\mathsf{m}l_1\mathsf{m}(l_2\mathsf{a}\text{␣})\mathsf{a}^k\mathsf{m}\text{␣}l_3\mathsf{a}\text{␣}r_1r_4r_2r_5r_3\mathsf{m}\text{␣␣}p_3p_5p_2p_4p_1) \xrightarrow{s_1}\\
&\qquad \mathsf{m}l_1(\mathsf{m}l_2\mathsf{a}\text{␣}\mathsf{a}^k\mathsf{m}\text{␣}(l_3\mathsf{m}\text{␣␣}))r_1r_4r_2r_5r_3\mathsf{a}\text{␣}p_3p_5p_2p_4p_1 \xrightarrow{s_2}\\
&\qquad\qquad \mathsf{m}l_1\mathsf{m}l_2\mathsf{m}\text{␣␣}\mathsf{a}^k\mathsf{a}l_3\mathsf{m}\text{␣␣}r_1r_4r_2r_5r_3\mathsf{a}\text{␣}p_3p_5p_2p_4p_1\\
&\mathsf{m}l_1(\mathsf{m}l_2\mathsf{a}\text{␣}\mathsf{a}^k\mathsf{m}\text{␣}(l_3\mathsf{a}\text{␣}))r_1r_4r_2r_5r_3\mathsf{m}\text{␣␣}p_3p_5p_2p_4p_1 \xrightarrow{s_2}\\
&\qquad (\mathsf{m}l_1\mathsf{m}(l_2\mathsf{m}\text{␣␣})\mathsf{a}^k\mathsf{a}l_3\mathsf{a}\text{␣}r_1r_4r_2r_5r_3\mathsf{m}\text{␣␣}p_3p_5p_2p_4p_1) \xrightarrow{s_1'}\\
&\qquad\qquad \mathsf{m}l_1\mathsf{m}l_2\mathsf{m}\text{␣␣}\mathsf{a}^k\mathsf{a}l_3\mathsf{m}\text{␣␣}r_1r_4r_2r_5r_3\mathsf{a}\text{␣}p_3p_5p_2p_4p_1
\end{aligned}$$

In the second subcase, the site of the ambiguity has the form

$$\mathsf{m}l_1\mathsf{m}l_2l_3l_4\mathsf{a}\text{␣}\mathsf{a}^k\mathsf{a}r_4\mathsf{m}\text{␣␣}p_4r_1r_5r_2\mathsf{m}\text{␣␣}p_2p_5p_1$$

for some $i, j, k \in \mathbb{N}$ such that the s_1 canyon has height $i + 1 + k$ and the s_2 canyon has height $k + 1 + j$; here $l_1 \in L_i$, $l_2 \in L_k$, $l_3 \in L_1$, $l_4 \in L_j$, $(r_4, p_4) \in R_j$, $(r_1, p_1) \in R_i$, $(r_5, p_5) \in R_1$, and $(r_2, p_2) \in R_k$. These ambiguities are resolved using a modification s_2' of s_2 that has right canyon wall $\mathsf{a}^k\mathsf{m}r_4$ instead of $\mathsf{a}^k\mathsf{a}r_4$:

$$\begin{aligned}
&(\mathsf{m}l_1\mathsf{m}(l_2l_3l_4\mathsf{a}\text{␣})\mathsf{a}^k\mathsf{a}(r_4\mathsf{m}\text{␣␣}p_4)r_1r_5r_2\mathsf{m}\text{␣␣}p_2p_5p_1) \xrightarrow{s_1}\\
&\qquad \mathsf{m}l_1(\mathsf{m}l_2l_3l_4\mathsf{a}\text{␣}\mathsf{a}^k\mathsf{m}r_4\mathsf{m}\text{␣␣}p_4\text{␣})r_1r_5r_2\mathsf{a}\text{␣}p_2p_5p_1 \xrightarrow{s_2'}\\
&\qquad\qquad \mathsf{m}l_1\mathsf{m}l_2l_3l_4\mathsf{m}\text{␣␣}\mathsf{a}^k\mathsf{m}r_4\mathsf{a}\text{␣}p_4\text{␣}r_1r_5r_2\mathsf{a}\text{␣}p_2p_5p_1\\
&\mathsf{m}l_1(\mathsf{m}l_2l_3l_4\mathsf{a}\text{␣}\mathsf{a}^k\mathsf{a}r_4\mathsf{m}\text{␣␣}p_4)r_1r_5r_2\mathsf{m}\text{␣␣}p_2p_5p_1 \xrightarrow{s_2}\\
&\qquad (\mathsf{m}l_1\mathsf{m}(l_2l_3l_4\mathsf{m}\text{␣␣})\mathsf{a}^k\mathsf{a}(r_4\mathsf{a}\text{␣}p_4)r_1r_5r_2\mathsf{m}\text{␣␣}p_2p_5p_1) \xrightarrow{s_1}\\
&\qquad\qquad \mathsf{m}l_1\mathsf{m}l_2l_3l_4\mathsf{m}\text{␣␣}\mathsf{a}^k\mathsf{m}r_4\mathsf{a}\text{␣}p_4\text{␣}r_1r_5r_2\mathsf{a}\text{␣}p_2p_5p_1
\end{aligned}$$

In the final subcase, the site of the ambiguity has the form

$$\mathsf{m}l_1\mathsf{m}l_2l_3l_4\mathsf{a}\text{␣}\mathsf{a}^k\mathsf{m}r_4\mathsf{m}\text{␣␣}p_4l_5\mathsf{a}\text{␣}r_1r_6r_2r_3r_5\mathsf{m}\text{␣␣}p_5p_3p_2p_6p_1$$

for some $i, j, k, m \in \mathbb{N}$ such that the s_1 canyon has height $i + 1 + k + 1 + m$ and the s_2 canyon has height $k + 1 + j$; here $l_1 \in L_i$ and $(r_1, p_1) \in R_i$, $l_2 \in L_k$ and $(r_2, p_2) \in R_k$, $l_3 \in L_1$ and $(r_3, p_3) \in R_1$, $l_4 \in L_j$ and $(r_4, p_4) \in R_j$, $l_5 \in L_m$ and $(r_5, p_5) \in R_m$, and finally $(r_6, p_6) \in R_1$. Here the two rules do not change each other's redexes, so the resolution is straightforward.

12.3.1.2 s_1 and s_2 have common bottom

When the bottom of the two rules coincide, we may without loss of generality assume that height of s_2 is greater than or equal to the height of s_1. If the two heights are equal then the rules are in fact equal, and the ambiguity has a trivial resolution. Otherwise the site of the ambiguity has the form

$$\mathsf{m}l_1\mathsf{a}l_2\mathsf{a}\sqcup r_1\mathsf{m}r_2\mathsf{m}\sqcup\sqcup p_2\sqcup p_1$$

where $l_1 \in L_k$, $l_2 \in L_j$, $(r_1, p_1) \in R_k$ and $(r_2, p_2) \in L_j$ for some $k, j \in \mathbb{N}$. These ambiguities are resolved using a truncation s_2' of s_2 that only acts on the canyon $\mathsf{m}l_2\mathsf{a}\sqcup r_2\mathsf{m}\sqcup\sqcup p_2$:

$$\begin{aligned}
&(\mathsf{m}l_1\mathsf{a}(l_2\mathsf{a}\sqcup)r_1\mathsf{m}(r_2\mathsf{m}\sqcup\sqcup p_2)\sqcup p_1) \xrightarrow{s_1} \mathsf{m}l_1(\mathsf{m}l_2\mathsf{a}\sqcup r_2\mathsf{m}\sqcup\sqcup p_2)r_1\mathsf{a}\sqcup p_1 \xrightarrow{s_2'}\\
&\qquad\qquad\qquad\qquad \mathsf{m}l_1\mathsf{m}l_2\mathsf{m}\sqcup\sqcup r_2\mathsf{a}\sqcup p_2r_1\mathsf{a}\sqcup p_1\\
&(\mathsf{m}l_1\mathsf{a}l_2\mathsf{a}\sqcup r_1\mathsf{m}r_2\mathsf{m}\sqcup\sqcup p_2\sqcup p_1) \xrightarrow{s_2} (\mathsf{m}l_1\mathsf{a}(l_2\mathsf{m}\sqcup\sqcup)r_1\mathsf{m}(r_2\mathsf{a}\sqcup p_2)\sqcup p_1)) \xrightarrow{s_1}\\
&\qquad\qquad\qquad\qquad \mathsf{m}l_1\mathsf{m}l_2\mathsf{m}\sqcup\sqcup r_2\mathsf{a}\sqcup p_2r_1\mathsf{a}\sqcup p_1
\end{aligned}$$

12.3.1.3 s_2 to the right of s_1

If s_2 has its bottom in the s_1 right canyon wall, then the s_2 canyon is to the right of the s_1 canyon, and there is some overlap between the left wall of the s_2 canyon and the right wall of the s_1 canyon. Depending on the extent of this overlap, there are three subcases: the a on top of the s_2 left canyon wall may be part of the s_1 right canyon wall, the m on top of the s_1 right cayon wall may be part of the part of the s_2 left canyon wall, or it may be that neither is part of the other's canyon wall; in the last case, there is a third canyon between the two considered, whose bottom is at an even higher elevation than either of the two.

In the first subcase, the site of the ambiguity has the form

$$\mathsf{m}l_1l_2l_3l_4l_5\mathsf{a}\sqcup r_1\mathsf{m}\mathsf{a}^k\mathsf{a}r_5\mathsf{m}\sqcup\sqcup p_5r_3\mathsf{m}\sqcup\sqcup p_3p_1$$

for some $l_1 \in L_i$, $l_2 \in L_1$, $l_3 \in L_k$, $l_4 \in L_1$, $l_5 \in L_j$, $(r_1, p_1) \in R_i$, $(r_3, p_3) \in R_k$, $(r_5, p_5) \in L_j$, and $i, j, k \in \mathbb{N}$; the s_2 canyon has height k whereas the s_1 canyon has height $i + 1 + k + 1 + j$. These ambiguities are resolved using a modification s_1' of s_1 that has right canyon wall $r_1\mathsf{m}\mathsf{a}^k\mathsf{m}r_5$ instead of $r_1\mathsf{m}\mathsf{a}^k\mathsf{a}r_5$:

$$\begin{aligned}
&(\mathsf{m}l_1l_2l_3l_4l_5\mathsf{a}\sqcup r_1\mathsf{m}\mathsf{a}^k\mathsf{a}r_5\mathsf{m}\sqcup\sqcup p_5(r_3\mathsf{m}\sqcup\sqcup p_3)p_1) \xrightarrow{s_1}\\
&\qquad\qquad \mathsf{m}l_1l_2l_3l_4l_5\mathsf{m}\sqcup\sqcup r_1(\mathsf{m}\mathsf{a}^k\mathsf{a}(r_5\mathsf{a}\sqcup p_5)r_3\mathsf{m}\sqcup\sqcup p_3)p_1 \xrightarrow{s_2}\\
&\qquad\qquad\qquad\qquad \mathsf{m}l_1l_2l_3l_4l_5\mathsf{m}\sqcup\sqcup r_1\mathsf{m}\mathsf{a}^k\mathsf{m}r_5\mathsf{a}\sqcup p_5\sqcup r_3\mathsf{a}\sqcup p_3p_1\\
&\mathsf{m}l_1l_2l_3l_4l_5\mathsf{a}\sqcup r_1(\mathsf{m}\mathsf{a}^k\mathsf{a}(r_5\mathsf{m}\sqcup\sqcup p_5)r_3\mathsf{m}\sqcup\sqcup p_3)p_1 \xrightarrow{s_2}\\
&\qquad\qquad (\mathsf{m}l_1l_2l_3l_4l_5\mathsf{a}\sqcup r_1\mathsf{m}\mathsf{a}^k\mathsf{m}r_5\mathsf{m}\sqcup\sqcup p_5\sqcup(r_3\mathsf{a}\sqcup p_3)p_1) \xrightarrow{s_1'}\\
&\qquad\qquad\qquad\qquad \mathsf{m}l_1l_2l_3l_4l_5\mathsf{m}\sqcup\sqcup r_1\mathsf{m}\mathsf{a}^k\mathsf{m}r_5\mathsf{a}\sqcup p_5\sqcup r_3\mathsf{a}\sqcup p_3p_1
\end{aligned}$$

In the second subcase, the site of the ambiguity has the form

$$\mathsf{m}l_1l_2l_3\mathsf{a}_{\lrcorner}r_1\mathsf{m}\mathsf{a}^k\mathsf{m}_{\lrcorner}l_5\mathsf{a}_{\lrcorner}r_3r_4r_5\mathsf{m}_{\sqcup\sqcup}p_5p_4p_3p_1$$

for some $l_1 \in L_i$, $l_2 \in L_1$, $l_3 \in L_k$, $l_5 \in L_j$, $(r_1, p_1) \in R_i$, $(r_3, p_3) \in R_k$, $(r_4, p_4) \in R_1$, $(r_5, p_5) \in R_j$, and $i, j, k \in \mathbb{N}$; the s_1 canyon has height $i + 1 + k$ whereas the s_2 canyon has height $k + 1 + j$. These ambiguities are resolved using a modification s_2' of s_2 that has left canyon wall $\mathsf{a}^k\mathsf{a}l_5$ instead of $\mathsf{a}^k\mathsf{m}_{\lrcorner}l_5$:

$$\begin{aligned}
&(\mathsf{m}l_1l_2l_3\mathsf{a}_{\lrcorner}r_1\mathsf{m}\mathsf{a}^k\mathsf{m}_{\sqcup}(l_5\mathsf{a}_{\sqcup})(r_3r_4r_5\mathsf{m}_{\sqcup\sqcup}p_5p_4p_3)p_1) \xrightarrow{s_1} \\
&\qquad \mathsf{m}l_1l_2l_3\mathsf{m}_{\sqcup\sqcup}r_1(\mathsf{m}\mathsf{a}^k\mathsf{a}l_5\mathsf{a}_{\lrcorner}r_3r_4r_5\mathsf{m}_{\sqcup\sqcup}p_5p_4p_3)p_1 \xrightarrow{s_2'} \\
&\qquad\qquad \mathsf{m}l_1l_2l_3\mathsf{m}_{\sqcup\sqcup}r_1\mathsf{m}\mathsf{a}^k\mathsf{a}l_5\mathsf{m}_{\sqcup\sqcup}r_3r_4r_5\mathsf{a}_{\sqcup}p_5p_4p_3p_1 \\
&\mathsf{m}l_1l_2l_3\mathsf{a}_{\lrcorner}r_1(\mathsf{m}\mathsf{a}^k\mathsf{m}_{\lrcorner}l_5\mathsf{a}_{\lrcorner}r_3r_4r_5\mathsf{m}_{\sqcup\sqcup}p_5p_4p_3)p_1 \xrightarrow{s_2} \\
&\qquad (\mathsf{m}l_1l_2l_3\mathsf{a}_{\lrcorner}r_1\mathsf{m}\mathsf{a}^k\mathsf{m}_{\sqcup}(l_5\mathsf{m}_{\sqcup\sqcup})(r_3r_4r_5\mathsf{a}_{\sqcup}p_5p_4p_3)p_1) \xrightarrow{s_1} \\
&\qquad\qquad \mathsf{m}l_1l_2l_3\mathsf{m}_{\sqcup\sqcup}r_1\mathsf{m}\mathsf{a}^k\mathsf{a}l_5\mathsf{m}_{\sqcup\sqcup}r_3r_4r_5\mathsf{a}_{\sqcup}p_5p_4p_3p_1
\end{aligned}$$

In the third subcase, the ambiguity has the form

$$\mathsf{m}l_1l_2l_3l_4l_5\mathsf{a}_{\lrcorner}r_1\mathsf{m}\mathsf{a}^k\mathsf{m}r_5\mathsf{m}_{\sqcup\sqcup}p_5l_6\mathsf{a}_{\lrcorner}r_3r_4r_6\mathsf{m}_{\sqcup\sqcup}p_6p_4p_3p_1$$

for some $l_1 \in L_i$, $l_2 \in L_1$, $l_3 \in L_k$, $l_4 \in L_1$, $l_5 \in L_j$, $l_6 \in L_m$, $(r_1, p_1) \in R_i$, $(r_3, p_3) \in R_k$, $(r_4, p_4) \in R_1$, $(r_5, p_5) \in R_j$, $(r_6, p_6) \in R_m$, and $i, j, k, m \in \mathbb{N}$; the s_1 canyon has height $i + 1 + k + 1 + j$ and the s_2 canyon has height $k + 1 + m$. Here the rules do not change each other's redexes, so the resolution is straightforward.

12.3.1.4 s_2 on top of s_1

The final case is that the bottom of the s_2 rule is the m on top of the s_1 right canyon wall. Here the site of the ambiguity has the form

$$\mathsf{m}l_1\mathsf{a}_{\lrcorner}r_1\mathsf{m}l_2\mathsf{a}_{\lrcorner}r_2\mathsf{m}_{\sqcup\sqcup}p_2p_1$$

for some $l_1 \in L_j$, $l_2 \in L_k$, $(r_1, p_1) \in R_j$, $(r_2, p_2) \in R_k$, and some $j, k \in \mathbb{N}$; the height of the s_1 canyon is j and the height of the s_2 canyon is k. These ambiguities are resolved using an extended variant s_2' of s_2 which has height $j + 1 + k$, left wall $l_1\mathsf{m}_{\lrcorner}l_2$, and right wall $r_1\mathsf{a}r_2$:

$$(\mathsf{m}l_1\mathsf{a}_{\sqcup}r_1\mathsf{m}(l_2\mathsf{a}_{\sqcup})(r_2\mathsf{m}_{\sqcup\sqcup}p_2)p_1) \xrightarrow{s_1} (\mathsf{m}l_1\mathsf{m}_{\sqcup}l_2\mathsf{a}_{\sqcup}r_1\mathsf{a}r_2\mathsf{m}_{\sqcup\sqcup}p_2p_1) \xrightarrow{s_2'} \\ \mathsf{m}l_1\mathsf{m}_{\sqcup}l_2\mathsf{m}_{\sqcup\sqcup}r_1\mathsf{a}r_2\mathsf{a}_{\sqcup}p_2p_1$$

$$\mathsf{m}l_1\mathsf{a}_{\sqcup}r_1(\mathsf{m}l_2\mathsf{a}_{\sqcup}r_2\mathsf{m}_{\sqcup\sqcup}p_2)p_1 \xrightarrow{s_2} (\mathsf{m}l_1\mathsf{a}_{\sqcup}r_1\mathsf{m}(l_2\mathsf{m}_{\sqcup\sqcup})(r_2\mathsf{a}_{\sqcup}p_2)p_1) \xrightarrow{s_1} \\ \mathsf{m}l_1\mathsf{m}_{\sqcup}l_2\mathsf{m}_{\sqcup\sqcup}r_1\mathsf{a}r_2\mathsf{a}_{\sqcup}p_2p_1$$

The conclusion of these calculations may be summarised as follows.

Lemma 12.1 *The rewrite system S of Definition 12.2 is locally confluent.*

It is somewhat curious that the resolutions are all squares, considering that the resolution of the ambiguity for ordinary associativity is (rather famously) a pentagon. Part of the reason for this is that with ordinary associativity there is a three rewrite steps sequence $\mathsf{m}_{\sqcup}\mathsf{m}_{\sqcup}\mathsf{m}_{\sqcup\sqcup} \to \mathsf{m}_{\sqcup}\mathsf{mm}_{\sqcup\sqcup\sqcup} \to \mathsf{mm}_{\sqcup}\mathsf{m}_{\sqcup\sqcup\sqcup} \to \mathsf{mmm}_{\sqcup\sqcup\sqcup\sqcup}$, but hom-associativity stops at the $\mathsf{mm}_{\sqcup}\mathsf{m}_{\sqcup\sqcup\sqcup}$ stage because there is no a in the position that a third rewrite step would require.

12.3.2 Full Confluence

Besides local confluence, the Diamond Lemma also requires termination of the rewrite system, which is usually established by providing a well-founded partial order relation which is compatible with (i) the rewrite system in question and (ii) composition in the operad; such a relation was given in [5], which basically is a length-lexicographic order, but extended to function symbols of arbitrary arity. First (the "length" part), two terms are unrelated unless the height of input i in one term equals the height of input i in another term, for all inputs; this makes no difference for any of the rewrite rules considered, but is used for establishing compatibility with composition for the second part. Second, the Polish notation words for terms are compared lexicographically, with respect to the alphabet order that makes $\mathsf{m} < \mathsf{a}$ and all other symbols (the inputs) isolated singleton components in the Hasse diagram. Hence for the rules in the rewrite system S the comparisons all come out as

$$\mathsf{m}l\mathsf{a}_{\sqcup}r\mathsf{m}_{\sqcup\sqcup} > \mathsf{m}l\mathsf{m}_{\sqcup\sqcup}r\mathsf{a}_{\sqcup}$$

(left hand side greater than right hand side) because the first position that differs is that on top of the left canyon wall, where the left hand side has an a and the right hand side has an m. Thus by the Diamond Lemma for operads (e.g. as found in [5]), we have the following.

Theorem 12.2 *The rewrite system S for strong hom-associativity is confluent. The strongly hom-associative operad has a set of normal forms consisting of precisely those terms which are irreducible with respect to S.*

Corollary 12.1 *Let $X \not\ni \mathsf{m}, \mathsf{a}$ be a set of constant symbols. Let Z be the set of all Polish notation terms on the alphabet $X \cup \{\mathsf{m}, \mathsf{a}\}$ that are irreducible with respect to the rewrite system S of Definition 12.2. Then the set of all formal linear combinations of elements of Z constitutes the free strongly hom-associative algebra generated by X, with operations*

$$\begin{aligned} \alpha(b) &= \textit{normal form with respect to } S \textit{ of } \mathsf{a}b && \textit{for } b \in Z, \\ \mu(b, c) &= \textit{normal form with respect to } S \textit{ of } \mathsf{m}bc && \textit{for } b, c \in Z. \end{aligned}$$

The rewrite system S is not minimal, because it is not autoreduced: there are plenty of rules whose left hand sides could be reduced by some smaller rule, because they have at least one a–m layer in their canyons. Imposing that restriction already when constructing S is quite doable, but requires coordinating the left and right sides of the canyons, which would further raise the level of technicality in Definition 12.2; better then to start out simple and add complications later.

Conjecture 12.1 *The smallest subset of S which is (i) confluent and (ii) contains the axiom rule* ma␣m␣␣ $\to$ mm␣␣a␣ *is precisely that subset of rules whose right hand sides are irreducible with respect to S, i.e., those rules whose canyons do not have a left wall* a *in the same layer as a right wall* m.

This conjecture is another argument that strong hom-associativity constitutes a natural simplification of (weak) hom-associativity: if all canyon identities are needed to attain confluence, then there is no complete weaker strengthening of weak hom-associativity than the full strong hom-associativity. (Of course, this is still not the same as saying that strong hom-associativity is a logical consequence of weak hom-associativity; the restriction to canyon identity rules is nontrivial, even if it is natural.)

Conjecture 12.2 *The free strongly hom-associative algebra over a field has no zero divisors.*

12.4 A Combinatorial Model

Though encoded as Polish notation terms, the basis Z for the free strongly hom-associative algebra described in Corollary 12.1 is natively a set of trees (in the computer science/formal language sense), so the problem of finding a more direct description of this set is technically one of describing a tree language. Standard tools [1] in formal language theory for addressing such tasks are automata and grammars. Preferably the description should be finitary.

The most notable class of tree languages are the *regular tree languages*, which in [5] were employed to compute initial terms of the Hilbert series of free *weakly* hom-associative algebras (and in particular of the corresponding operad). The major shortcoming of that work was however that since not all rules of a confluent rewrite system were known, it was not possible to take into account more than a finite set of rules, and thus the results could only approximate the Hilbert series (even if it was determined how many terms were correct). Since for the present case of strongly hom-associative algebras the full set of rules (namely the canyon identities) is known, that first obstacle has been overcome. Unfortunately that success also makes it clear that regular tree languages will not suffice.

The problem lies in the restriction that both walls of a canyon must have the same height, and that this height is not bounded; this means there are infinitely many subcases to account for, which contradicts the requirement that a regular tree language can be recognised by a finite automaton. More concretely, a regular tree language on $\{\mathsf{a}, \mathsf{m}\} \cup X$ can be defined as one which is recognised by some deterministic finite bottom–up tree automaton, the core of which is an algebra (in the universal algebra sense) with signature $\{\mathsf{a}, \mathsf{m}\} \cup X$ on some *finite* set S (the set of states). Evaluating a term in this algebra amounts to reading the tree in bottom–up order (from leaves to root), and all information that has been extracted from some branch is encoded into the automaton state, which is an element of S; at every a or m the state is modified to take into account the additional information that became available after reading that far in this branch. In particular, it is at the m at the bottom of a canyon that information from the left branch is combined with information from the right branch, and a decision must be reached on whether this canyon matched any of the canyon identities. To do this, the state must for each height k separately keep track of whether there is an a at that height in the right side of the branch (since that becomes the left wall of a canyon), and similarly for each k keep track of whether there is an m at that height in the left side of the branch, but that is (countably) infinitely many different cases to distinguish whereas regularity requires that the set of states S is finite.

Finite tree automata could handle matching against rift identities just fine, or even rift identities where the height difference is a multiple of some fixed integer, but even so restricted these would be identities that do not hold in familiar hom-associative algebras. Finite tree automata could also handle matching against canyon identities up to any fixed height, but then the corresponding system of rewrite rules is not confluent. Constructing a combinatorial model therefore requires switching to some other class of languages.

Other classes of tree languages are known in the literature, but the desirable property of being able to describe the *complement* of a language—needed here because we seek the set of *irreducible* elements, starting from the rewrite rules which say what is *reducible*—is rare beyond the regular languages. Surprisingly, an alternative can be to look at *word* languages for some encoding of the trees as words, because the literature on formal word languages is larger than that on tree languages. Even here closure under taking the complement is a rare boon, but there is one formalism that does the trick: parsing expression languages.

12.4.1 Parsing Expression Grammars

The class of *parsing expression languages* [2] was invented mostly for practical reasons, but it turned out to have impressive theoretical properties as well, constituting a sweet spot in the space of formal language frameworks. For the reader's convenience, we here give a brief introduction to parsing expressions.

Syntax descriptions in contemporary computing are predominantly given on *Backus–Naur form* (BNF), sometimes with convenience extensions (see e.g. [6]), which technically means they are *context-free grammars* (CFGs). An example of such would be

$$H ::= \mathsf{x} \mid \mathsf{a}H \mid \mathsf{m}HH,$$

which describes the language H of Polish notation hom-algebra terms on $\{\mathsf{x}\}$; the way to read this is "a hom-algebra term is x, or a followed by a hom-algebra term, or m followed by two hom-algebra terms". The letters x, a, and m are here referred to as *terminals* (actual letters in the words being described), whereas H is called a *nonterminal* (representing a set of words) and must be defined using some equation such as that shown; recursion in these equations is common. A *grammar* consists of a set of such equations, together with one nonterminal called the start symbol; the corresponding language is that of all terminal words which can be produced from the start symbol through some sequence of expansions of nonterminals.

Though ubiquous in documentation, the practice of computing does not quite adhere to these grammars; they are often overlaid with additional rules to guide the parser, in ways that cannot be expressed in the grammar alone. One classical example is the "dangling else" problem, which concerns whether an else clause belongs to an inner or an outer if statement; the relevant grammar rule could be written

$$S ::= L{=}R \mid \textbf{if}\ R\ \textbf{then}\ S \mid \textbf{if}\ R\ \textbf{then}\ S\ \textbf{else}\ S$$

and the ambiguity arises for $\textbf{if}\ a\ \textbf{then}\ \textbf{if}\ b\ \textbf{then}\ c = d\ \textbf{else}\ e = f$, where it is not clear whether the else clause belongs to the inner '**if** b' or the outer '**if** a'. Compilers are written to deterministically pick one of these interpretations, but language specifications cannot simply use a CFG to describe which one should be picked.

Parsing expression grammars (PEGs) resolve this ambiguity by replacing the unordered choice '|' of a CFG by the *prioritised choice* operation '/'; the meaning of the parsing expression $A\ /\ B$ is 'first try matching against A at this point, and only if that fails then attempt to match against B'. The intent that an **else** belongs to the closest possible **if** can then be encoded into the grammar rule

$$S ::= L = R\ /\ \textbf{if}\ R\ \textbf{then}\ S\ \textbf{else}\ S\ /\ \textbf{if}\ R\ \textbf{then}\ S$$

since this will cause S to match from the second **if** until the end of the example, which means a match at the first **if** against the second S branch $\textbf{if}\ R\ \textbf{then}\ S\ \textbf{else}\ S$ fails, allowing the third branch $\textbf{if}\ R\ \textbf{then}\ S$ to be tried there and succeed. This determinism in the parsing also provides for another development, namely the use of *packrat parsers* to perform parsing in linear time. Naively, the question of whether some input matches a parsing expression can be answered using a backtracking algorithm—try all variants in sequence, report success when one matches, report failure when none of them do—even if the recursive nature of the grammar rules can lead to this algorithm descending into some very deep recursions. A packrat parser caches the outcomes of all attempted matchings of nonterminals so that each combination of position and nonterminal need only be tried once, and since the number of nonterminals is bounded this provides a linear bound on the runtime of the parser. That is a far tighter complexity bound than one gets for CFGs, which instead have been proved [7] to be tied to the complexity of boolean matrix multiplication (with side of matrix equal to length of input to parse)!

In a packrat parser, it is furthermore easy to support a number of additional features, so PEGs provide these as well. In particular, they provide the lookahead constraint predicate '&' and the negative lookahead predicate '!', which we shall rely upon in what follows. The function of these is that the parsing expression $\&(E)$ matches the empty word $\mathsf{1}$ (so it does not consume any input), but only in a position where the subexpression E would match; $!(E)$ conversely matches the empty word in a position where E would not match. Lookaheads are a common feature in advanced regular expressions (though convenient, they do not change the class of languages that can be described), but when combined with recursion they gain powers beyond what recursion alone can manage. Concretely, the PEG

$$D ::= \mathsf{b}D\mathsf{c} \;/\; \mathsf{1}$$
$$E ::= \mathsf{a}E\mathsf{b} \;/\; \&(D!(T))$$

where T matches any single terminal (and $!(T)$ thus only matches at the end of input) has the property that $E!(\mathsf{b})$ only matches words on the form $\mathsf{a}^n\mathsf{b}^n\mathsf{c}^n$. The terminal D by itself matches an arbitrary number of bs followed by an equal number of cs, in a manner familiar from CFGs. Similarly E without the lookahead constraint matches an arbitrary number of as followed by an equal number of bs, so that when the lookaheads are put in, a match requires equal numbers of as, bs, and cs. This example is important because the language $\{\mathsf{a}^n\mathsf{b}^n\mathsf{c}^n\}_{n=0}^{\infty}$ is (by the so-called Pumping Lemma) one of the classical examples of something which is *not* context-free.

Surprisingly, the lookahead constraints do not extend the class of parsing expression languages; any PEG that employs them can be transformed into another PEG for the same language which only makes use of concatenation, prioritised choice, and recursion [2]. This is another reason to regard the parsing expression languages as a sweet spot in the space of formal language frameworks.

12.4.2 Hom-Associativity

From what was said above, an approach for describing the language of hom-algebra terms that are irreducible with respect to the canyon identities now suggests itself: start with a vanilla recursive grammar for the language of all hom-algebra terms, then insert lookahead constraints in strategic places to exclude anything that contains a canyon with an a on the left side and an m on the right side. The equal height constraint that so wexed regular languages is easy to impose using the recursion technique employed above to get equal exponents in $\mathsf{a}^n\mathsf{b}^n\mathsf{c}^n$. Easy, that is, provided there are explicit letters to count; for simplicity we will therefore first consider a somewhat elaborate encoding of hom-algebra terms as words.

The *clockwise notation* can be explained as walking around the tree form of a term (root is down) in clockwise direction, and recording each node (operation or generator) every time it gets encountered. For an operation p of arity k this means it will produce $k+1$ symbols in the encoding as a word: first a symbol p_0 before the first child (operand), last a symbol p_k after the kth child, and between these $k-1$ symbols $\mathsf{p}_1, \dots, \mathsf{p}_{k-1}$ separating one child from the next. In the case of hom-algebra terms, the encoding can be carried out by defining the hom-algebra operations μ and α as

$$\mu(b,c) = \mathsf{m}_0 b \mathsf{m}_1 c \mathsf{m}_2,$$
$$\alpha(b) = \mathsf{a}_0 b \mathsf{a}_1;$$

for example $\mu\bigl(\alpha(\mathsf{x}), \mu(\mathsf{x},\mathsf{x})\bigr)$ then evaluates to its clockwise notation encoding $\mathsf{m}_0\mathsf{a}_0\mathsf{x}\mathsf{a}_1\mathsf{m}_1\mathsf{m}_0\mathsf{x}\mathsf{m}_1\mathsf{x}\mathsf{m}_2\mathsf{m}_2$. In this encoding, the concept of being a canyon is captured by the rule

$$C ::= (\mathsf{m}_2/\mathsf{a}_1)C(\mathsf{m}_0/\mathsf{a}_0)\ /\ \mathsf{m}_1$$

(a canyon is either a left wall segment followed by a canyon followed by a right wall segment, or a canyon bottom) and the vanilla language of hom-algebra terms on X becomes

$$H ::= X\ /\ \mathsf{a}_0 H \mathsf{a}_1\ /\ \mathsf{m}_0 H \mathsf{m}_1 H \mathsf{m}_2.$$

Combining these, we get

$$I ::= X\ /\ \mathsf{a}_0 I \mathsf{a}_1 !(C\mathsf{m}_0)\ /\ \mathsf{m}_0 I \mathsf{m}_1 I \mathsf{m}_2 \qquad (12.10)$$

as a parsing expression rule for the language I of clockwise notation hom-algebra terms which are irreducible with respect to the canyon identities. This may be summarised as a theorem.

Theorem 12.3 *There is a parsing expression language encoding a basis of the free strongly-hom associative algebra.*

Why use this spacious clockwise encoding of hom-algebra terms, though—wouldn't the more compact Polish notation or reverse Polish notation be a more familiar choice? Indeed they would, but it is not clear that the canyon identities in that case can be described using parsing expressions. For Polish notation, lookaheads to avoid a match against a canyon identity have to be imposed at the m at the canyon bottom; beyond that point, there is no longer enough information in the unread part of the input to correlate height in left canyon wall to height in right canyon wall. The problem with this is that all canyon heights have to be checked at the same point, which is an infinite number of lookaheads. It *is* possible to adapt to the canyon height actually at hand, for example

$$\begin{aligned} H &::= X \mathbin{/} \mathsf{a}H \mathbin{/} \mathsf{m}HH, \\ G &::= (\mathsf{a}/\mathsf{m}H)G(\mathsf{m}/\mathsf{a}) \mathbin{/} \mathsf{a}H \end{aligned} \tag{12.11}$$

will have the G in $\mathsf{m}!(G\mathsf{m})HH$ recursing k times, for the maximal k such that there is an a at that height in the left wall and also a right wall of height k, to a net effect of preventing match against the height k canyon identity; backtracking would start out with k being the height of the topmost a in that left wall, and then decreasing as long as the right wall has height lower than k. Unfortunately this does nothing to guard against matches for canyon identities of lower height; as soon as one match is found for G, then that is the only one considered. Nor would it be possible to cater for the lower heights by correlating the initial and final terminals contributed by expanding G, because these are in general at different heights; if an expansion of G to $\mathsf{a}G\mathsf{m}$ contributes the a to the left wall at height i, then the right wall m will be at height $k - 1 - i$, so this a–m combination need not contradict irreducibility.

For the reverse Polish notation, the situation is different in that lookaheads instead would have to be placed at the a on top of the left canyon wall; thus at most one canyon identity would have to be matched against per position in the input, which avoids the main complication facing a parsing expression grammar for the Polish case. On the other hand, it is now no longer clear how to locate the corresponding position in the right wall. In more detail, one must first observe that the trivial reversion

$$R ::= X \mathbin{/} R\mathsf{a} \mathbin{/} RR\mathsf{m}$$

of (12.11) does not yield a parsing expression for reverse Polish hom-algebra terms, since it becomes left-recursive (and thus not well-formed: in order to determine whether R matches at a particular position, one might first have to determine whether R matches at that position!) To make a PEG for reverse Polish notation, one should instead focus on the language S of arity 1 suffixes of reverse Polish words, as that permits

$$\begin{aligned} S &::= \mathsf{a} \mathbin{/} XS^{*}\mathsf{m}, \\ R &::= XS^{*} \end{aligned}$$

as a well-formed grammar to the same end. However the naive insertion of canyon lookaheads

$$Y ::= (\mathsf{a}/\mathsf{m})Y(\mathsf{a}/R\mathsf{m}) \ / \ RR\mathsf{m}$$ (canyon, plus everything attached to the right wall, not including bottom),

$$Q ::= \mathsf{a}!(Y\mathsf{m}) \ / \ XQ^*\mathsf{m}$$ (like S, but with lookaheads)

does not make XQ^* a parsing expression for only irreducible terms in reverse Polish notation; the catch is that the branch $RR\mathsf{m}$ of Y which is supposed to match the subterm on top of the right canyon wall is too greedy. Rather than stopping at the right height, the S^* in the first R will consume the entire right wall of the canyon, after which matching Y fails and the lookahead constraint achieves nothing. Support for repetition that is just greedy enough to allow a later part of the expression to match is not something that parsing expressions provide.

Conjecture 12.3 *The word languages in Polish notation and reverse Polish notation, for hom-algebra terms that are irreducible with respect to the canyon identities, are not parsing expression languages.*

For enumeration, the more verbose nature of the clockwise notation is not a problem. What does constitute a challenge is that parsing expressions is an *analytical* language framework—it tests concrete words for belonging to a language—rather than a *generative* framework, as it is the generative aspect of regular languages that one makes use of when translating regular expressions to Hilbert series. Presentation does however matter; the use of lookaheads in (12.10) makes it unfeasible to apply generative-style reasoning to this grammar, but the equivalent lookahead-free PEG could be more tractable. Whereas there is an algorithm for eliminating lookaheads from PEGs, there cannot by general computability theory be an algorithm for extracting the Hilbert series for an arbitrary parsing expression language from a lookahead-free PEG for it, though nothing prevents extracting the Hilbert series from any particular such PEG.

References

1. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Löding, C., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications (web book), version of November 18, 2008. http://tata.gforge.inria.fr/
2. Ford, B.: Parsing expression grammars: a recognition-based syntactic foundation. In: Jones, N.D., Leroy, X. (eds.) Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2004, Venice, Italy, January 14–16, 111–122. ACM (2004)
3. Guo, L., Zhang, B., Zheng, S.: Universal enveloping algebras and Poincaré-Birkhoff-Witt theorem for involutive Hom-Lie algebras. J. Lie Theory **28**(3), 739–759 (2018)
4. Hartwig, J.T., Larsson, D., Silvestrov, S.D.: Deformations of Lie algebras using σ-derivations. J. Algebra **295**, 314–361 (2006)

5. Hellström, L., Makhlouf, A., Silvestrov, S.D.: Universal algebra applied to hom-associative algebras, and more. In: Makhlouf, A., Paal, E., Silvestrov, S., Stolin, A. (eds.), Algebra, Geometry and Mathematical Physics, Springer Proceedings in Mathematics and Statistics, vol. 85, 157–199 (2014)
6. International Standards Organization. Syntactic metalanguage – Extended BNF, 1996. ISO/IEC 14977
7. Lee, L.: Fast context-free grammar parsing requires fast Boolean matrix multiplication. J. ACM **49**(1), 1–15 (2002)
8. Lyndon, R.C.: On Burnside's problem. Trans. Am. Math. Soc. **77**, 202–215 (1954)
9. Makhlouf, A., Silvestrov, S.D.: Hom-algebra structures. J. Gen. Lie Theory Appl. **2**(No. 2), 51–64 (2008)
10. Shirshov, A.I.: Subalgebras of free Lie algebras. (Russian) Mat. Sbornik N.S. **33**(75), 441–452 (1953)
11. Shirshov, A.I.: Some algorithm problems for Lie algebras. (Russian) Sibirsk. Mat. Ž. **3**, 292–296 (1962)
12. Yau, D.: Enveloping algebra of Hom-Lie algebras. J. Gen. Lie Theory Appl. **2**, 95–108 (2008)
13. Zheng, S., Guo, L.: Free involutive Hom-semigroups and Hom-associative algebras. Front. Math. China **11**(2), 497–508 (2016)