Derivatives Hidden in Word Puzzles

Donald Spector

Dept. of Physics, Hobart and William Smith Colleges, Geneva, NY, USA; spector@hws.edu

Abstract

We consider two closely related word puzzles, and show that each of these is based on an operation that can be interpreted mathematically as a derivative. While the appearance of this mathematical structure in such a context might be surprising, the identification arises quite naturally in both cases, and the resulting derivatives are in the same family as the derivative defined in type theory. We construct the derivatives appropriate to each of these word puzzles, and show how they can be used to frame these puzzles and to motivate some additional related questions.

Introduction

This paper is about a surprising mathematical feature I realized was present when I was playing with a particular word puzzle. There is, of course, a wide range of word puzzles and wordplay that involve manipulating the letters with which words are spelled. For example, while one can always take a word and reverse its spelling (e.g., turning “bridge” into “egdirb”), it is interesting to find words which, when reversed, produce another valid word (e.g., “drawer” into “reward”), or to find palindromes, words which, when reversed, produce the same word (e.g., “level”). Anagrams form another realm, words whose letters can be shuffled to produce new words (e.g., “night” into “thing”), with the added feature sometimes that those anagrams are also synonyms (e.g., “evil” and “vile”). The Wordways archive [5] and Eckler’s *Making the Alphabet Dance* [1] are sources in which to see the range of wordplay puzzles people have engaged in.

The puzzle I happened to be thinking about was based on this question: Can you find a word such that, upon deletion of any letter in that word, the remaining letters still spell a valid word? Such an occurrence is called a Baltimore deletion [4]. (For the examples in this paper, I will work with English words, though the ideas can be employed in other alphabetic languages as well.) Most words do not have this property, but one can find examples. For example, “pear” produces, upon Baltimore deletion, four possible words: ‘pea$\cancel{r}$,’ ‘pe$\cancel{a}$r,’ ‘p$\cancel{e}$ar,’ and ‘$\cancel{p}$ear,’ yielding the words ‘pea,’ ‘per,’ ‘par,’ and ‘ear.’ An alternative exercise is to consider whether each of the possible deletions produces a group of letters that can be anagrammed (rearranged) to form a word; thus, while “gear” upon ordinary deletion would produce ‘gea,’ ‘ger,’ ‘gar,’ and ‘ear,’ not all of which are words, each of these letter groupings can be rearranged to spell a word: ‘age,’ ‘erg,’ ‘rag,’ and ‘ear.’ (Of course, the choice of words in this second case is not necessarily unique; here, the penultimate word could have been left as ‘gar,’ and the last word could just as well be chosen to be ‘are’ or ‘era.’) This permutation-based alternative is called a Baltimore transdeletion [4].

As I will show in this paper, the underlying processes that generate Baltimore deletions and Baltimore transdeletions have the mathematical structure of a derivative. In other words, we might say that the goal of these puzzles is to find words that are differentiable! We will also use our findings to motivate some additional word puzzle questions. Furthermore, the derivatives we will define below (each of the puzzles above is associated with a slightly different derivative) are in the same family as the type theory derivatives defined by McBride [3].

Derivatives

In a first course in calculus, the derivative of a function $f : \mathcal{R} \to \mathcal{R}$ at a point $x \in R$ is defined via

$$\frac{df}{dx} = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}\,. \tag{1}$$

From this definition, one obtains a variety of results, including, most notably for our purposes, the Leibniz rule, namely that, given the product of two functions f and g,

$$\frac{d}{dx}(fg) = \frac{df}{dx}g + f\frac{dg}{dx}\,. \tag{2}$$

However, derivatives can arise in contexts that have nothing to do with limits. Indeed, given operators D and X for which their order of application matters, but with the property that $DX - XD = \mathcal{I}$ (where $\mathcal{I}$ is the identity operator), then D will act as a derivative on expressions involving X, reproducing the familiar power rules, the Leibniz rule, and so forth. This algebraic approach can be used to formulate the derivatives defined below, but in the interests of space, I will focus on demonstrating that the structures defined below satisfy the Leibniz rule. (Some authors refer to an operation that satisfies the Leibniz rule as a *derivation*, but we will keep the term derivative, just as is done in [3].)

In what we discuss below, it will be useful to introduce the notion of a *bag*, also called a *multiset*, with the symbols $\wr$ and $\int$ as delimiters when we wish to denote a bag by listing its elements. A bag is like a set, except one keeps track of multiplicities. Thus, given the bags $\mathcal{M} = \wr a, b, c \int$ and $\mathcal{N} = \wr b, d, e \int$, the union $\mathcal{M} \cup \mathcal{N} = \wr a, b, b, c, d, e \int$. Note that the order of the items in a bag does not matter; $\wr a, b, b \int = \wr b, a, b \int$. For our purposes, the union will play the role of the sum, so we will also write $\mathcal{M} \oplus \mathcal{N} = \mathcal{M} \cup \mathcal{N}$.

The Derivative Underlying Baltimore Deletions

Suppose we have two sequences, $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$. If we were just dealing with sequences, we would define the product of these sequences as their concatenation, $X \cdot Y = x_1x_2\ldots x_m y_1 y_2 \ldots y_n$. However, we are going to be interested in bags of sequences, and so if we have one sequence, we will think of it as the bag $\mathcal{X} = \wr X \int$. Now suppose we have two bags of sequences, $\mathcal{M}$ and $\mathcal{N}$, whose members are $\mathcal{M}_i$ and $\mathcal{N}_j$, respectively. Then the product $\mathcal{M} \circ \mathcal{N}$, which we will term the c-product (due to the underlying connection to concatenations), will be the bag containing all sequences (counting multiplicities, as these are bags, not sets) of the form $\mathcal{M}_i \cdot \mathcal{N}_j$. Thus, $\mathcal{M} \circ \mathcal{N} = \wr \mathcal{M}_i \cdot \mathcal{N}_j | \mathcal{M}_i \in \mathcal{M}$ and $\mathcal{N}_j \in \mathcal{N} \int$.

We are now in a position to define a derivative on bags of sequences. We will label this derivative D_c, and refer to it as the c-derivative. Given a one-sequence bag $\mathcal{M} = \wr M \int$, we define the derivative D_c on $\mathcal{M}$ to be the bag of all sequences obtained by dropping one element of the sequence M. Thus, for example, given the sequence $x_1x_2x_3x_4$,

$$D_c \wr x_1x_2x_3x_4 \int = \wr x_1x_2x_3, x_1x_2x_4, x_1x_3x_4, x_2x_3x_4 \int\,. \tag{3}$$

It is an easy exercise to check that this operation satisfies the Leibniz rule, namely that

$$D_c(\mathcal{M} \circ \mathcal{N}) = (D_c\mathcal{M}) \circ \mathcal{N} \oplus \mathcal{M} \circ D_c\mathcal{N}\,. \tag{4}$$

Note that if $\mathcal{M}$ and $\mathcal{N}$ are one-sequence bags, equation (4) is simply the statement that the deletions of the concatenation of two sequences consists of all the deletions of the first sequence concatenated with the second sequence combined with the first sequence concatenated with all the deletions of the second sequence. Note, too, that the c-product is not commutative, so one must be careful to maintain the ordering in equation (4). (For example, $D_c(\mathcal{M} \circ \mathcal{M}) = (D_c\mathcal{M}) \circ \mathcal{M} \oplus \mathcal{M} \circ D_c\mathcal{M}$, but this cannot be simplified to $2\mathcal{M} \circ D_c\mathcal{M}$; this is not special to the c-product, but arises naturally when products are non-commutative, and thus there is the possibility that an object and its derivative might not commute.)

In other words, the operation we used in Baltimore deletions is the derivative operator on sequences. I would suspect that most puzzlers who play with such word manipulations would have no idea that they were calculating a derivative, but the operation in question is, mathematically, precisely such an operation.

To make this concrete, if we have the sequence “pear,” then

$$D_c \lbag\, pear \rbag = \lbag pea, per, par, ear \,\rbag \ . \tag{5}$$

The goal of the Baltimore deletion puzzle, then, is to find words whose c-derivatives are bags all of whose members are words. We will say that a sequence is cw-differentiable if the sequence and all of the elements of its c-derivative are words. (Formally, we would define a set W to serve as the dictionary, and a word would be any element of this set. One sees that the choice of W is also implicit in these puzzles.)

Of course, this immediately invites new problems. For example, can one find words w_1 and w_2 that are not only themselves cw-differentiable, but for which $w_1 \cdot w_2$ is also cw-differentiable? Are there words that are twice cw-differentiable, which we will take to mean that all the sequences that arise from completing two deletions are words? Notice that a word (sequence) can be twice cw-differentiable but not cw-differentiable if all the double deletions are words, but not all the single deletions are. Note, too, that since we count multiplicities, given an initial sequence all of whose characters are distinct, every element in the bag of second c-derivatives will appear twice.

One could also define a restricted c-derivative, such as a restricted c-derivative that only executes deletions on consonants, or vowels, or some other subset of the letters. Such a restricted derivative will still satisfy the Leibniz rule, but would lead to a broader class of differentiable words (which in turn would produce variations on the original word puzzle).

Finally, we note that the c-derivative bears a formal similarity to the type theory derivative defined by McBride [3]. In that situation, however, one keeps track of the location of the holes left by deletion [2], and the focus is on the derivative of the abstract type, rather than particular instances thereof.

Permutations, a New Derivative, and Baltimore Transdeletions

With the insights of the last section, we can now easily define a derivative appropriate to the second word puzzle introduced above, namely, Baltimore transdeletions. We begin by thinking not about sequences and subsequences, but rather the equivalence classes of all sequences with the same letters occurring with the same multiplicities. Let us denote the equivalence class of, say, the sequence “cab” as $\langle cab \rangle$, and so $\langle cab \rangle = \langle bac \rangle = \langle acb \rangle$. We say an equivalence class is *wordy* if it contains at least one member that is a word; thus $\langle bde \rangle$ is wordy (the letters can be rearranged to spell “bed”), but $\langle bdek \rangle$ is not (there is no English word consisting of precisely these four letters). If X and Y are sequences, then we define the product on equivalence classes in the natural way, such that $\langle X \rangle \bullet \langle Y \rangle = \langle X \cdot Y \rangle$.

We now define a derivative suitable to Baltimore transdeletions, which we call the p-derivative (due to its association with permutations) and denote D_p. Given a sequence X, we define $D_p \lbag \langle X \rangle \rbag$ to be the bag containing all the equivalence classes of terms that arise in $D_c \lbag X \rbag$. Formally,

$$D_p \lbag \langle X \rangle \rbag = \lbag \langle u \rangle | u \in D_c \lbag X \rbag \rbag .$$

Thus, for example,

$$D_p \lbag \langle abcd \rangle \rbag = \lbag \langle abc \rangle, \langle abd \rangle, \langle acd \rangle, \langle bcd \rangle \,\rbag \ .$$

As this example shows, the generalization from the c-derivative to the p-derivative is exactly what one would expect, once one shifts froms sequences to equivalence classes of sequences.

As in the previous case, it is an easy exercise to see that the p-derivative on bags of sequence equivalence classes obeys the Leibniz rule, where the product and sum of bags is as before, although of course the elements of the bags are now equivalence classes of sequences. (This derivative, too, is in the same family as the type theory derivatives of [3].)

The goal of the Baltimore transdeletion puzzle, then, is to find wordy equivalence classes whose p-derivatives are bags all of whose members are wordy. We will say that a sequence is pw-differentiable if the sequence and all of the elements of its p-derivative are wordy. Thus, with Baltimore transdeletions, we again have a wordplay puzzle that is really an exercise in a kind of differentiation. Naturally, too, the p-derivative structure invites the same sorts of questions as the previous case did, such as finding wordy equivalence classes whose products are wordy, or considering multiple transdeletions to get higher p-derivatives.

In this vein, we note that the sequence $\langle payer\rangle$ is an example of an equivalence class that is not pw-differentiable (the equivalence class $\langle aepy\rangle$ is not wordy), but is twice pw-differentiable; the second p-derivative of $\langle payer\rangle$ is the bag that consists of the following equivalence classes (each occurring twice), all of which are wordy: $\langle pay\rangle$, $\langle pae\rangle$, $\langle par\rangle$, $\langle pye\rangle$, $\langle pyr\rangle$, $\langle per\rangle$, $\langle aye\rangle$, $\langle ayr\rangle$, $\langle aer\rangle$, and $\langle yer\rangle$. Note that this phenomenon arises not because of the abstract structure of deletions and transdeletions, but because of the additional constraint that our results be wordy.

Closing Thoughts

In this paper, we identified a hidden mathematical structure inside two standard pieces of wordplay. Both the Baltimore deletion and Baltimore transdeletion rely on operations that can be understood as derivatives. The c- and p-derivatives satisfy the Leibniz rule, and have structural similarities to the type theory derivatives introduced by McBride [3]. The word puzzles associated with Baltimore deletions and Baltimore transdeletions can thus be framed as problems in differentiability, where we include a requirement that the inputs and output are all words (or permutations of words) in order for some input to be considered differentiable.

While it is not the point of this article to explore the wordplay side of these Baltimore-style puzzles, it is worth mentioning that one can construct Baltimore deletion and transdeletion puzzles where instead of sequences whose characters can be deleted, we have sentences whose words can be deleted, and the derivative structure persists. Indeed, one could, in these scenarios, use equivalence classes based on pronunciation, so that homophones (like "write,", "rite," and "right," for example) would be interchangeable, and this would not get in the way of being able to identify a derivative in this generalized context.

Presumably, there are interesting mathematical structures hiding in other word puzzles. If such structures can be identified, it would be intriguing to see what kinds of further wordplay they might inspire. Also, given that the process of deletions that has a long history in wordplay turned out be the key to formulating the zipper technique [2] for working with data structures, one should be alert to possible data structure applications in any new instances of mathematical structures embedded in word puzzles.

References

[1] R. Eckler. *Making the Alphabet Dance*, 1st ed. St. Martins Griffin, 1997.

[2] G. Huet. "The Zipper." *J. Functional Programming*, vol. 7, no. 5, 1997, pp. 549–554.

[3] C. McBride. *The Derivative of a Regular Type is its Type of One-Hole Contexts.* http://strictlypositive.org/diff.pdf

[4] National Puzzler's League. *2008 Edition of the Online Guide to the Engima.* https://enigma.puzzlers.org/guide/remove

[5] Wordways archive. https://digitalcommons.butler.edu/wordways/all_issues.html