# Lenses between Multiple Datatypes Part 2

## 0.1 Text Edit Buffer

**Deletion**

If we are going to be able to edit text in a window, we need to be able to delete lines. Let $A$ be the following buffer:

apple
banana
cherry
dog
eclair

and let us be focused on line 3, meaning that $B = \text{cherry}$ and $C = (3, 5)$. If we delete the third line, changing $A$ to

apple
banana
dog
eclair

then we must refocus the $B$ and $C$ views on another line. The following options are reasonable outputs to this operation:

| Description | $B$ | $C$ |
|---|---|---|
| Jump to the beginning | apple | $(1, 4)$ |
| Jump to the end | eclair | $(4, 4)$ |
| Jump to the next line | dog | $(3, 4)$ |
| Jump to the previous line | banana | $(2, 4)$ |
| Jump to the (round $\frac{3}{5}4$)th line | banana | $(2, 4)$ |

While the "next line" and "previous line" options make sure the edit buffer stays in the same general area in the file, the last option allows the edit buffer to scale through the buffer. This effect becomes more prominent if the $B$ buffer holds more than one line.

On the other hand, what if our goal is to delete the line given by $B$ and propagate the update to $A$ and $C$? When we update $B =< \texttt{deleted} >$, we can change $A$ to be the desired

apple
banana
dog
eclair.

But what do we expect for $C$? In a standard text editor, we would expect the cursor to jump to the next line, and then fill the $B$ buffer with that line. But our lenses likely would not accommodate that kind of feedback. Therefore the only possible output for $C$ is a "non-existent" marking, like $(-1, 4)$.

## 0.2 Function Focusing

For this example we have three structures: $A$ contains a function $f : \alpha \to \beta$; $B$ contains an input $b : \alpha$; and $C$ contains some output $c : \beta$. The lens is in sync if $f(b) = c$.

If we change the value of $f$ to $f' : \alpha \to \beta$, we must update $C$ to $f'(b)$ in order to stay in sync.

If we change $B$ to some $b' : \alpha$, we have two options. First, we might update $C$ to contain $f(b')$; this option represents a look-up. Second, we might change $A$ to refer to the function

$$\lambda v.\ \text{if } v = b' \text{ then } c \text{ else } f\ v.$$

Similarly, if we change $C$ to some $c' : \beta$, we should update $A$ to refer to

$$\lambda v.\ \text{if } v = b \text{ then } c' \text{ else } f\ v.$$

## 0.3 Wikipedia Structure

The next example has the property that one of the three structures encompass all the information held by the other two.

Wikipedia articles include the following three parts: $A$, the source which is a list of headings $h_i$, content $c_i$ and authors $a_i$; $B$, the table of contents which consists of headings and authors; and $C$, the article body which consists of headings and content. The layout is summarized as follows:

| Source | Table of Contents | Body |
|---|---|---|
| $h_1, c_1, a_1$ | $h_1, a_1$ | $h_1, c_1$ |
| $h_2, c_2, a_2$ | $h_2, a_2$ | $h_2, c_2$ |
| $h_3, c_3, a_3$ | $h_3, a_3$ | $h_3, c_3$ |

Every update to the source $A$ can be propagated to $B$ and $C$ by simple projections. On the other hand, updates to $B$ can be put into $A$ and then projected from $A$ to $C$; the same holds for updates to $C$. Therefore, every operation by this lens can be seen as a combination of get and putback functions between $A \leftrightarrow B$ and $A \leftrightarrow C$.

## 0.4 Partition

We can view the partition lens perhaps more naturally as a lens between three data structures:

$$A \subset (X+Y)^* \qquad B \subset X^* \qquad C \subset Y^*$$

The behavior is almost exactly the same as the traditional partition problem. Updates to $A$ project to $B$ and $C$ as expected. Updates to $B$ affect $A$ but not $C$, and vice verse. The only caveat is that while the traditional partition can accommodate simultaneous updates to $B$ and $C$, the three-way version can only update $B$ and $C$ independently.

## 0.5 List Concatenation

To motivate list concatenation, suppose we have three data structures: $A$ which records a first name, $B$ which records a last name, and $C$ which is the concatenation of $A$ and $B$. Changes to $A$ and $B$ can easily result in an update to $C$. The interesting point is when an update is made to $C$ which adds a word between the first and last name. Since that insertion may be a first name or a last name, the separation again is ambiguous.

As an example, consider the names Sandy *dos* Santos[1] and Junie *B.* Jones[2]

[1]dos Santos is a popular Portuguese and Spanish last name.

[2]The protagonist of a children's book series, Junie B. gets mad at anyone who refuses to call her by her chosen first name.