Real estate of names

K. Rustan M. Leino

Compaq Systems Research Center,
130 Lytton Ave., Palo Alto, CA 94301, U.S.A.
rustan.leino@compaq.com

(KRML 97, 13 July 2000)

A common convention for writing names (identifiers) in mathematical formulas makes poor use of the real estate on the page occupied by those names. Here is a design principle for using space more efficiently.

The choice of a suitable notation is a vital part of designing clear and convincing presentations of scientific material. Good notation is precise, concise, and suggestive; poor notation may be ambiguous, clumsy, or misleading. Integral to the design of notation is choosing what *names* (identifiers) to use to represent the entities involved. How to typeset formulas is also an important part of the design, for one would like formulas to use the available real estate effectively and efficiently.

Let us examine how some of these design decisions are often made by focusing on a common usage of subscripts. Figure 0 shows two equations from physics [1] and one from computer science [0]. The first of these equations states that the potential difference between points a and b equals the difference between the potentials at these points. (The potential "at a point" is really the potential difference between that point and some implicit point, usually ground.) The second equation states that the angular magnification is minus the ratio of the focal lengths of the objective lens and eyepiece lens. The third equation specifies the lag time of a FIFO synchronizer circuit with N elements in terms of the cycle time and a timing margin.

a) $V_{ab} = V_a - V_b$

b) $m_\theta = -\dfrac{f_{ob}}{f_{ey}}$

c) $t_{cy} + t_m < t_{lag} < N\, t_{cy} - t_m$

Figure 0. Examples of a common usage of names in formulas.

The names in these formulas contain two parts: the *type* (like V for voltage), indicating which attribute is being measured, and the *brand* (like a), indicating where the measurement is obtained. In Figure 0, the types use a larger font than the brands.

By encoding the types as part of the names themselves, it is easy to type-check the formulas. For example, by subtracting a voltage from a voltage in the right-hand side of equation (a), one gets a voltage, which is indeed the type indicated by the name in the left-hand side of the equation.

But the brands are important, too. In fact, they are paramount in Figure 0—for one, they distinguish names with equal types. Unfortunately, the proportion of space allocated to the respective parts of the names in Figure 0 make the types more prominent than the brands. For example, the capital types stand out when looking at equation (a), whereas the subscript brands are less noticeable. Yet it is the brands of this equation that make it what it is. Example (b) demonstrates why being able to identify the brands is important: by confusing the focal lengths of the objective and the eyepiece, one would calculate a vastly different angular magnification. Timing equations like the one in example (c) often occur in a context with even more time variables, but reading off which quantity's time is being denoted requires inspection of the brands, which occupy a regrettably small part of the total real estate of the names.

To show just how ridiculous the frequently-used convention in Figure 0 is, consider the rendition of Euclid's algorithm for computing the greatest common divisor in Figure 1.

A better way to write the names in Figure 0 is to make the brands more visible. For example, Figure 2 shows the same formulas, but with the types contracted and the brands enlarged. I am not arguing that types should be given as prefix superscripts. The point is that the brands of these names ought to be bigger and more noticeable than the types.

In some programming contexts, making type or usage information part of the names of program variables can be quite helpful. The notational convention *Hungarian* [2] (named in reference to Charles Simonyi, who popularized the notation) comes to mind. Figure 3 shows three example variable declarations. The first declaration uses the Hungarian prefix "c" to denote that the integer is a count (in this case a count of warnings, as the brand suggests). The type of the variable name in the second declaration indicates that the variable is a pointer to a character (the "current" character under inspection). The type of the third variable name specifies that the variable contains a pointer to a null-terminated string (the text of a message). In these names, the balance between types and brands seems more well-suited than in Figure 0.

In some contexts where type information is understood or not particularly relevant, the type can be left out of the name altogether. For example, this leads to a nicer rendition of Euclid's algorithm, writing x instead of $\mathbb{Z}_x$ in Figure 1. Similarly, if brand information is understood, it can be omitted. This is often done in Hungarian and in physics. For example, in Einstein's famous equation $E = mc^2$, the brand-less names E and m denote the energy and mass of one (unnamed) object.

I don't have any hard and fast rules for how to write down names. But the moral is this: the most important parts of your names should get the bulk of their real estate.

$$
\begin{array}{l}
\{\ 1 \le \mathbb{Z}_x = \mathbb{Z}_X \ \wedge\ 1 \le \mathbb{Z}_y = \mathbb{Z}_Y\ \} \\
\textbf{do}\ \mathbb{Z}_x < \mathbb{Z}_y \quad \longrightarrow \quad \mathbb{Z}_y := \mathbb{Z}_y - \mathbb{Z}_x \\
\ \square\ \ \mathbb{Z}_y < \mathbb{Z}_x \quad \longrightarrow \quad \mathbb{Z}_x := \mathbb{Z}_x - \mathbb{Z}_y \\
\textbf{od} \\
\{\ \mathbb{Z}_x = \mathbb{Z}_y = \mathbb{Z}_X \,\textbf{gcd}\, \mathbb{Z}_Y\ \}
\end{array}
$$

Figure 1. A rendition of Euclid's algorithm in which the types overwhelm the brands.

a) ${}^{\mathrm{v}}ab = {}^{\mathrm{v}}a - {}^{\mathrm{v}}b$

b) $m_\theta = -\dfrac{{}^{f}ob}{{}^{f}ey}$

c) ${}^{t}cy + {}^{t}m < {}^{t}lag < N\,{}^{t}cy - {}^{t}m$

Figure 2. A different layout of the parts of the names makes it clearer which quantities are being denoted.

```
int cWarnings;
char * pchCurrent;
char * pszMessage;
```

Figure 3. Example declarations of variables in Hungarian.

Acknowledgements

The thoughts I've presented here have been influenced by the attention devoted to the streamlining of mathematical notation and mathematical arguments spearheaded by Edsger W. Dijkstra. This influence was bestowed upon me in a class on "mathematical methodology" I took from Professor Dijkstra in Spring 1989 at UT Austin. I am grateful not just to Dijkstra, but also to the attendees of the *In Pursuit of Simplicity* symposium (honoring Dijkstra, May 2000, at UT Austin) whose positive feedback on the short talk I gave on this subject caused me write these thoughts down. Thanks also to Lyle Ramshaw who provided useful feedback on drafts of this note.

Some related work is done by Tufte [3], who also gives design principles for presentations, but for using graphs to convey data rather than for using mathematical notation to convey concepts.

REFERENCES

0. William J. Dally and John W. Poulton. *Digital Systems Engineering*, page 479. Cambridge University Press, 1998.
1. David Halliday and Robert Resnick. *Physics*, part two, pages 699 and 982. John Wiley & Sons, third edition, 1978.
2. Charles Petzold. *Programming Windows*. Microsoft Press, 1988.
3. Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, 1983.