Quantification without Quantifiers (or Epsilons):

A New Notation for Quantification in Predicate Logic

Jason Zarri

1. *Introduction*

I believe I've found a way to say everything that one would normally say in predicate logic using quantifiers, but without actually using any, and also without using Hilbert's epsilon operator[1]. Instead of saying that something holds for all x by binding x with a universal quantifier, one can say it by capitalizing all of x's occurrences in a sentence that would, in "standard notation", be in the scope of the quantifier. And instead of saying that something holds for some y by binding y with an existential quantifier, one can say it by leaving all of y's occurrences un-capitalized. Thus, in "variable notation"—so called because it expresses quantification by modifying variables—instead of $\forall x\ (Fx)$ one would write FX, and instead of $\exists y\ (Gy)$ one would write Gy.

2. *Some Problems about Scope and Their Solution*

Immediately, a problem presents itself. In standard notation the order of the quantifiers in a sentence can make a big difference to its meaning. How can one make up for this loss if we discard quantifiers? My solution is to use *scope brackets*, '[' and ']'. For every distinct variable in a sentence there is a pair of scope brackets that determine the scope of the claim involving that variable. Where, in standard notation, a quantifier would be within the scope of another quantifier, in variable notation the scope brackets of the former quantified claim are inside the scope brackets of the latter quantified claim. The brackets should enclose no more than they need to.

Some examples will make this clear. Consider (1) and (2), and their counterparts (1') and (2'):

[1] See "The Epsilon Calculus," http://plato.stanford.edu/entries/epsilon-calculus/.

1. ∀x ∃y (Rxy)	1’. [RX[y]]
2. ∃y ∀x (Rxy)	2’. [R[X]y]

In (1’) the scope brackets for y enclose only y, and the scope brackets for X enclose the entire sentence, indicating that the existential claim is governed by the universal claim. In (2’) the reverse is true, indicating that the universal claim is governed by the existential claim.

Another problem is the scope of quantifiers in relation to the scope of truth-functional connectives. For instance, in standard notation, whether a negation connective is inside or outside the scope of a quantifier can make a very big difference to the meaning of the sentence. In variable notion, this problem can also be solved with scope brackets:

3. ∀x ~ (Fx)	3’. [~ FX]
4. ~ ∀x (Fx)	4’. ~ [FX]

In (3’) the scope brackets for X enclose the negation connective, indicating that the negation is governed by the universal claim. In (4’) the scope brackets for X do not enclose the negation connective, indicating that the negation connective is the main operator of the sentence. These subsequent examples should now be fairly easy to interpret:

5. ∀x ~ (∃y (Rxy))	5’. [~ RX[y]]
6. ~ ∀x ∃y (Rxy)	6’. ~ [RX[y]]

(7) and (8) and their counterparts are, of course, equivalent:

7. ∀x ∀y R(xy)	7’. [RX[Y]]
8. ∀y ∀x R(xy)	8’. [R[X]Y]

These examples involving binary connectives should also be fairly easy to interpret:

9. ∀x (Fx ⊃ ∃y (Gy))	9’. [FX ⊃ [Gy]]
10. ∀x (Fx) ⊃ ∃y (Gy)	10’. [FX] ⊃ [Gy]
11. ∀x (Fx ∨ Gx)	11’. [FX ∨ GX]
12. ∀x (Fx) ∨ ∀x (Gx)	12’. [FX] ∨ [GX]

Furthermore, I would translate:

13. ∃y ∃z (Ryz ⊃ Syz)

as:

13’. [Ry[z ⊃ Syz]]

And:

14. $\exists z\ \exists y(Ryz\ \supset Syz)$
as:
14’. [R[yz ⊃ Sy]z]
I think one can also handle more complex examples, like:
15. $\exists x\ \exists y\ \exists z\ (Rxyz\ \&\ Sxyz)$
My idea is to start by placing scope brackets around the variable with the smallest scope:
Rxy[z & Sxyz]
Then one works outwards, enclosing as little as necessary:
Rx[y[z & Sxyz]]
Finally, the outermost scope brackets, and only they, should enclose all the predicate letters:
15’. [Rx[y[z & Sxyz]]]

How about cases of multiple consecutive occurrences of a variable, like:

16. $\exists x\ (Rxx \supset Sxx)$

Since there is only one variable, we can write:

16’. [Rxx ⊃ Sxx]

Also,

17. $\exists x\ (Rxx) \supset \exists x\ (Sxx)$

would translate as :

17'. [Rxx] ⊃ [Sxx]

Thus, in variable notation it is a rule that, if there are multiple consecutive occurrences of a variable, one puts the scope brackets for that variable around its outermost occurrences. If, as in (16), every occurrence of a variable in a sentence is an occurrence of the same variable, the scope brackets will enclose all the predicate letters in that sentence. And if, as in (17), every occurrence of a variable in a sub-sentence is an occurrence of the same variable, the scope brackets will enclose all the predicate letters in that sub-sentence.

3. *Scope Brackets Refined*

What should we make of this fiendish sentence:

18. $\exists x\ \exists y\ \exists z\ (Rzyx \equiv Rxyz)$

For a first try, we could write:

18'. $[R[[zyx \equiv Rxyz]]]$

By inspecting the translation one can infer that x doesn't have the narrowest scope, for otherwise the first step would yield:

$Rzy[x \equiv Rx]yz$

Similarly, y also cannot have the narrowest scope, for then we would get:

$Rz[yx \equiv Rxy]z$

The only remaining possibility is that z must have the narrowest scope. But which variable has the second narrowest scope? Well, it obviously can't be z. It could be either x or y, both of whose translations would look like:

$R[[zyx \equiv Rxyz]]$

To avoid this difficulty, we could also use parentheses and curly brackets as scope brackets, associating each variable with a distinct kind of bracket. For example, let's use parentheses for x, square brackets for y, and curly brackets for z. We would then have:

18''. $(R[\{zyx \equiv Rxyz\}])$

What, though, if one is dealing with more than three variables? Consider this slight variant of the preceding:

19. $\exists w\ \exists x\ \exists y\ \exists z\ (Rzyxw \equiv Rwxyz)$

One thing we can do is concatenate grouping symbols using dashes to get "double scope brackets," e.g., '(-(' and ')-)', '[-[' and ']-]', and '{-{' and '}-}' which we then associate with the new variables. If we associate the double parentheses with w, we get:

19'. (-(R([{zyxw ≡ Rwxyz}]))-)

A translation of an analogous five-variable case, namely:

20. ∃v ∃w ∃x ∃y ∃z (Rzyxwv ≡ Rvwxyz)

would, associating double square brackets with v, come out as:

20' [-[R(-(([{zyxwv ≡ Rvwxyz}]))-)]-]

Similarly, if we are dealing with more than six variables, we could associate triple scope brackets, e.g., '(-(-(' and ')-)-)', '[-[-[' and ']-]-]', and '{-{-{' and '}-}-}' with the new variables. There is, in principle, no limit to the number of grouping symbols we can concatenate to form compound scope brackets, and hence there is, in principle, no limit to the number of variables we can handle using variable notation.

I should mention that this is not the only way to refine the notion of a scope bracket. One alternative would be to take the variable corresponding to a scope bracket and embed it inside parentheses. For example, one could have '(x(' and ')x)', '(y(' and ')y)', '(z(' and ')z)', and so on. Let's call these variables *marker variables*, since they are used to mark the scope of "real variables," the ones used to express quantified claims. To prevent confusion between real variables and marker variables, we could always surround real variables with bars, as in '|x|', '|y|', and '|z|'. Correspondingly, we can call the new scope brackets *marker parentheses*. To see how this would look in practice, consider this further variant of (18):

18'''. (x(R(y((z(|z||y||x| ≡ R|x||y||z|)z))y))x)

Here it is immediately obvious which scope brackets go with which variables, but the problem is that, at least in my opinion, using marker parentheses makes sentences much harder to write or to read.

4. *Conclusion*

Of what value might variable notation be? For one thing, if it can be made to work, it would demonstrate one possible way of dispensing with quantifiers without any loss of expressive power. For another, I think it would be interesting to try to teach variable notation to students and to determine whether standard notation or variable notation is easier to learn, or whether one notation is easier for students with one kind of learning style and another with another. Our knowledge of logic might be *a priori*, but our knowledge of how best to teach logic is surely empirical.

Bibliography

Avigad, Jeremy and Zach, Richard, "The Epsilon Calculus", *The Stanford Encyclopedia of Philosophy* (Summer 2012 Edition), Edward N. Zalta (ed.), URL = <http://plato.stanford.edu/entries/epsilon-calculus/>.