1. Introduction

Yablo argues that paradox can be generated without self-referential expressions. But he
denies that “self-reference suffices for paradox” (1993: 251). This is obvious if he means that not
all cases of self-reference are paradoxical (e.g., ‘This sentence is a sentence.’). Yet since he
justifies the claim by citing Tarski and Gödel, Yablo likely meant something stronger. An
alternate reading is: Self-reference does not suffice for any pathological sentence, at least in a
semantically open language – that is to say, in a language without semantic terms (‘true’, ‘refers,
‘satisfies’, etc.) which are defined on expressions of that self-same language.\(^1\) This is a standard
view; however, in what follows it is shown to be strictly incorrect. Yet we shall also see that, due
to the nature of the case, we are not forced into any radical revisions of classical logic.

Caveat: It is mostly left implicit in the literature that self-reference is unconstrained in a
semantically open language. As far as I know, Gödel and Tarski never affirm this overtly;
Yablo’s remark is the closest I can find. More often, self-referential devices seem to enjoy a
“presumption of innocence,” as suggested by Tarski’s and Gödel’s work. (Consider also
Henkin’s 1949 completeness proof for first-order logic.) At any rate, I expect that working

\(^1\) The term ‘semantically open language’ is from Tarski (1944). Tarski restricted himself to such languages so to
bypass Liar-like paradoxes, yet this is no longer popular due to alternate approaches by Kripke (1972) and Gupta &
Belnap (1993) (although Kripke’s is patently nonclassical given that it rejects the law of excluded middle). Still, it is
widely assumed that if a language is semantically open, unrestricted self-reference alone is insufficient for
contradiction. That is what I question in this paper.
logicians will recognize the presumption of innocence as orthodox; however, I shall suggest that it requires qualification.

2. An Argument for Absurdity

Consider a classical theory that includes denumerably many constants ‘a₀’, ‘a₁’, ‘a₂’…, interpreted as follows (where $n$ is the numeral for $n$):

(i) ‘a₀’ denotes 0
(ii) For all $n > 0$,┌$a_n$┐ denotes $n – 1$

(Notation: Per Quine 1951, an expression with corner quotes denotes the concatenation of symbols enclosed in the corners, after the replacement of any metavariables. Thus, as concerns ┌$a_n$┐, if ‘$n$’ is replaced with ‘1’, the corner-quoted expression then denotes the concatenation of the symbol ‘a’, followed by a subscripted ‘1’. I.e., it denotes ‘a₁’.) Now as is well-known, if our theory also includes quotation marks, the substitution of co-referring terms within quotation marks will be invalid. Consider that our theory has the consequence that:

$a_0 = a_1$

Thus, given the truism that ‘a₀’ = ‘a₀’, substitution inside quotation would allow us to deduce the absurdity that: ‘a₀’ = ‘a₁’. Such a maneuver is therefore not permitted in classical logic.

What is much the same, a substitutable variable inside quotation marks is sufficient to express an ill-defined function. Thus, consider the expression:

‘x’

---

2 It may be desirable here and with similar sentences to use the notation of Boolos (1995), so to disambiguate which quotation marks are paired together.

3 This connection between substitution into quotes and an ill-defined function was observed already by, e.g., Tarski (1933/1983: 161).
This would normally mention the symbol used as a variable, viz., the (italicized) 24th letter of the alphabet. But suppose instead that the variable inside the quotes is substitutable, so that it is functionally equivalent to a “blank” that can be filled in by any term. Then, the expression expresses an ill-defined function $i(x) = 'x'$, one which yields different outputs for $a_0$ depending on which term for $a_0$ replaces the variable:

\[
\begin{align*}
  i(a_0) &= 'a_0' \\
  i(a_1) &= 'a_1'
\end{align*}
\]

To preclude this, a quoted variable is always regarded as a name rather than a functional expression.\(^4\)

Notwithstanding, my claim is that an analogous ill-defined function can be re-created by exploiting the possibilities for self-reference, even in a semantically open language. Suppose we attempt to define an expression ‘$f$’ in such a way that $\lceil f(a_0)\rceil$ denotes the very constant that occurs inside the parentheses. That is, let us define ‘$f$’ so that it effectively means “the constant that is hereby preceded by ‘$f$’ and followed by ‘$’.” If self-reference could be given free rein, such a thing ought to be permissible and could be formulated as follows:

\[
(*) \quad \text{For all } n, f(a_n) = \lceil a_n \rceil
\]

However, (*) introduces an expression that is ill-defined on input $a_1$ in exactly the same way as $i(x)$. After all, (*) implies that:

\[
\begin{align*}
  (1) & \quad f(a_0) = 'a_0' \\
  (2) & \quad f(a_1) = 'a_1'
\end{align*}
\]

\(^4\) This should not suggest a problem with the function that maps an expression onto its quotation. If $\alpha$ is an expression, $q(\alpha) = \lceil '\alpha' \rceil$ remains perfectly well defined; consider that $q('a_0') = "a_0"$ and $q('a_1') = "a_1"$. (The outputs here are different of course, but so are the inputs.)
So like $i$ from before, $f$ varies between outputting ‘$a_0$’ and ‘$a_1$’ on input $a_0$, depending on which name for the input used. Concurrently, whereas the definition at (*) rules that (2) is true, (2) is provably equivalent to the following absurdity:

$$f(a_1) = 'a_0'$$

The absurdity of (3) is even clearer in noting that, by the transitivity of identity, (2) and (3) imply that ‘$a_0$’ = ‘$a_1$’. And note that the substitution needed to show the equivalence between (2) and (3) is not a substitution inside quotation marks.

Aside: The similarity between $f$ and quotation is not coincidental; $f$ as defined at (*) is a limiting case of the quoting function $i$. Note that if ‘$f$’ was replaced in (*) with the right-quotation mark, and ‘$)$’ were replaced with the left-quotation mark, the result would define an expression ‘‘$a_n$’’ in which the co-referring terms ‘$a_0$’ and ‘$a_1$’ would be interchangeable inside quotation. Yet even though ‘$f$’ bears a kinship to quotation, it simultaneously becomes clear that quotation has a self-referential aspect. A quotation effectively means “the very expression occurring within the single quotes.”

At any rate, since ill-defined functions are to be excluded, the language must not allow such a thing as (*). But contra the usual attitude, this means that self-reference cannot be given free rein in a semantically open language.

3. Discussion

Further elaboration is in order. First, note that the underline in (*) expressing the numeral function is not equivalent to quotation; witness that $0 = 0^2$ whereas ‘$0$’ ≠ ‘$0^2$’. However, ‘$n$’ may appear to be a semantic expression, for $n$ is the numeral that denotes the number $n$. If so, then the language may not be semantically open. However, the pathology ultimately depends on neither
‘\(n\)’, nor the underline, nor quotes, nor corner quotes. It is enough to have a (well-defined) function \(h\) such that \(h(n)\) outputs the \(n\)th constant – that is, \(h(0) = ‘a_0’, h(1) = ‘a_1’, and so on.\(^5\) In lieu of (*), we then can make do with the following:

\[
(**) \quad \text{For all } n, f(a_n) = h(n)
\]

Given that \(a_0 = a_1\), the absurdity at (3) is derivable in a similar fashion.\(^6\)

It might be thought that the problem owes to (*) and (**) quantifying into the subscript position. But while this seems to be sufficient for the problem, I do not believe it is necessary (see the Appendix). Besides, classical logicians otherwise have need of quantification into subscript position. E.g., when we introduced infinitely many constants into the language, we often require a clause such as (ii) above. Quantification into subscript position occurs elsewhere too, as in some presentations of the anti-diagonal function in Cantorian arguments. (This is not to suggest a problem with such presentations; the point instead is that our argument requires only standard formal devices.)

It can be said that ‘\(f\)’ expresses an intensional notion, for its denotation depends on what term for its input is being used. This is noteworthy insofar as it is a new kind of intensionality; it does not owe to a propositional attitude verb, an idiom like ‘so called’, attribute abstraction, a modal operator, or (to repeat) substitution inside quotes.\(^7\) More important, however, is that the definition at (**) clearly uses only formal, nonsemantic terms. This reveals that we can prove an absurdity in a semantically open language simply by exploiting self-reference. Or from another angle, unchecked self-reference enables intensional contexts, even if the language lacks the usual intensional elements.

\(^5\) One could also avoid quotation marks here by instead using descriptions of the constants, such as ‘the first letter of the Latin alphabet subscripted by the first Arabic numeral.’

\(^6\) My thanks to Alexander Pruss for raising the concern about ‘\(n\)’ and suggesting a function like \(h\) in response.

\(^7\) On such cases of intensionality, see Quine (1960, sections 30ff.).
One may object that the preceding shows merely that definitions like (***) are not legitimate. But that is exactly the point: Such definitions must be excluded. Still, the definition is problematic simply because ‘f’ fails to express a well-defined function, and ill-defined functions are already verboten. Agreed – this is why the argument does not have any drastic revisionary implications for classical logic. At the same time, special attention is called for when commonplace formal devices give rise to an ill-defined function. This is how one might regard the expression ‘i’ from section 2. Quotation seems unremarkable at first glance; it is just a convenient way to introduce metalinguistic terms. But substitution inside quotation makes for an ill-defined function, and so, a word of warning is required.

Similarly, self-reference has been ‘presumed innocent’ by the working logician. But unrestricted self-reference suffices for an ill-defined function, even in a semantically open language – and so, an explicit advisory is apt. Put differently, the ban on ill-defined functions implies certain restrictions on self-reference; even though this seem not to be acknowledged.

4. Intensionality with Gödel Numbers

In conversation, Tim Button has worried that if unrestricted self-reference suffices for absurdity, then Peano Arithmetic might be shown unsound via the method of Gödel numbering. For Gödel numbering enables something functionally like self-reference, and unrestricted self-reference now appears sufficient for intensional contexts.

A derivative argument is indeed readily available using Gödel numbers, as I shall explain. But the argument shows only that Gödel numbering in the metalanguage cannot be used without restriction. It hardly implies that the arithmetical object language is defective. (Nothing in Peano Arithmetic functions like self-reference prior to Gödel numbering.) So again, the
general prohibition on ill-defined functions means that there is no revolutionary import here.

Even so, it is worth remarking that the ban on ill-defined functions implies certain restraints on self-referential mechanisms, including Gödel numbering.

The reasoning at issue is the following. Suppose that \( g(\varepsilon) \) is a function that takes in an expression \( \varepsilon \) and outputs its Gödel number. Then, define the symbol ‘c’ as follows:

\[
(†) \quad c(a_n) = g(\lceil a_n \rceil)
\]

This defines \( \lceil c(a_n) \rceil \) in such a way that it denotes the Gödel number for the \( n \)th constant, i.e., the code for the very constant that occurs between ‘c(‘ and ‘)’. Thus:

1. \( c(a_0) = g(’a_0’) \) \[By definition of ‘c’\]
2. \( c(a_1) = g(’a_1’) \) \[By definition of ‘c’\]

And since each expression has its own Gödel number, we know that:

3. \( g(’a_0’) \neq g(’a_1’) \) \[By definition of ‘g’\]

But consider again that:

4. \( a_0 = a_1 \) \[By definition of the constants\]

Then, given the truism that \( c(a_0) = c(a_1) \), the indiscernability of identicals assures us:

5. \( c(a_0) = c(a_1) \) \[From 4\]

However, the earlier observations entail:

6. \( c(a_0) \neq c(a_1) \) \[From 1–3\]

Yet lines 5 and 6 contradict.

Basically, ‘c’ expresses a well-defined function only if no two constants co-refer. Hence, ‘c’ does not express a well-defined function, for our language can very well contain such co-referring constants. The above definition of ‘c’ must therefore be precluded. One might have thought, however, that it is acceptable to define an expression that denotes whichever Gödel
number you please. But as our definition shows, ‘c’ cannot be rigged to denote the Gödel number for the very constant occurring between ‘c(‘ and ‘)’. Otherwise, for some \( n \neq m \), it can turn out that \( a_n = a_m \), even though \( c(a_n) \neq c(a_m) \).

The suggestion that Gödel numbering enables contradiction will provoke anxiety—but again, let me reassure the reader that the foregoing has no radical implications. Granted, since any recursive function is numeralwise expressible in Peano Arithmetic, it may seem that that if contradiction is provable using a Godel numbering function, then contradiction is provable in the arithmetical object language. Yet the Godel numbering operation at (†) is not a well-defined function, much less a recursive one. So again, formal arithmetic is in no danger.

It remains true that self-reference—and the coding analog of self-reference—cannot be completely unrestrained. But this hardly means that every case of such a thing leads to pathology. (An analogy: The intensional expression ‘i’ from section 2 does not show all use of quotation to be illegitimate.) The point is just that if Gödel codes are abused in a certain way, the proviso against ill-defined functions will be violated.

5. Closing Remark

Thus, even Gödel numbering cannot be used without regulations. Again, one can assume that existing formal systems are free of the intensional cases. Even so, it is worth acknowledging how self-referential operations require some restraint, especially since the matter is glossed over by the usual “presumption of innocence.”
Appendix

In conversation, Gabe Dupre emphasizes that self-reference is not needed for the kind of intensionality at issue. After all, instead of (**), the following could be used to define ‘f’ and no self-reference would thereby be implicated:

(††) For all \( n \), \( f(a_n) = n+57 \)

The problem thus arises with other sorts of operations defined by quantifying into the constants’ subscript position. But while this is correct, the point remains that the self-referential operation is sufficient for the pathology. This, again, is contrary to the usual presumption of innocence.

Indeed, the intensionality also results from self-reference absent any subscript-quantification, as I shall now illustrate. But for this, we shall require an approach to definitions that is not always employed, although it is standard in Tarskian semantics.

The relevant approach is one where a functional expression is defined not by using ‘=’ in the object language, as at (*), (**), (†), or (††) but rather by using the denotation-relation in the metalanguage, akin to how the constants were introduced at (i) and (ii). In Tarskian semantics, the function symbols are defined among the metalanguage base clauses; these are then used in connection with recursive clauses to determine the truth-conditions of wff in the object language. Thus, the Tarskian base clauses might include the following, where \( \alpha \) is any metalinguistic term:

\[
\begin{align*}
\Gamma(\alpha)' &\text{ denotes } \Gamma(\alpha) \\
\end{align*}
\]

Given any expression, this outputs the quotation of the expression. E.g., if ‘a’, ‘b’, ‘c’… ‘t’ are constants, then \( q(\text{‘a’}) = \text{‘a’} \), \( q(\text{‘b’}) = \text{‘b’} \), etc. (The function \( q \) is a well-defined function; it is not to be confused with the intensional expression \( i \) from section 2.)

The key observation, then, is that unrestricted self-reference would allow the following to be introduced into the object language. Where \( \tau \) is any singular term (metalinguistic or not):
\[ f'(\tau) \] denotes \( \tau \)

Given a singular term \( \tau \), the concatenation of ‘\( f(' \)’, followed by \( \tau \), followed by ‘\)’, is interpreted to denote \( \tau \) itself. Again, if self-reference were unconstrained, such a definition would be thus far permissible. But suppose that \( a = b \). Then, (‡) implies:

\[
\begin{align*}
  f'(a) &= 'a' \\
  f'(b) &= 'b'
\end{align*}
\]

Even though the input is the same, the output differs depending on which term for the input is used. Unrestrained self-reference is again sufficient by itself for an intensional expression, and in this case, it is made possible without any subscript-quantification.

**Acknowledgments**

I thank Bradley Armour-Garb, John G. Bennett, Matteo Bianchetti, Ethan E. Brauer, Tim Button, David Chalmers, Morgan Davies, Gabe Dupre, Anil Gupta, Thomas Hofweber, Cory Johnson, Yael Loewenstein, William Lycan, David McCarty, Jay Newhard, Bryan Pickel, Alexander Pruss, Michael Resnik, David Ripley, and Keith Simmons for discussion on various versions of this material. I also thank an audience at the 2018 Central Division meeting of the American Philosophical Association, at the 2022 meeting of the Canadian Philosophical Association, and at the 2022 Joint Session of the Aristotelean Society and the Mind Association.
References


