THE JOURNAL OF PHILOSOPHY

VOLUME LXXXIX, NO. 5, MAY 1992

POINTERS TO TRUTH*

Consider the following:

Jack: What I am saying at this very moment is nonsense.
Jill: Yes, what you have just said is nonsense.

Apparently, Jack spoke nonsense and Jill spoke to the point. But Jack and Jill seem to have asserted the same thing: that Jack spoke nonsense. Wherefore the difference?

To avoid the vagueness and the conflicting intuitions that go with 'nonsense', let us replace it by 'not true' and recast the puzzle as follows:

line 1: The sentence on line 1 is not true.
line 2: The sentence on line 1 is not true.

If we try to evaluate the sentence on line 1 we find ourselves going in an unending cycle. For this reason alone we may conclude that it is not a true sentence. Moreover, we are driven to this conclusion by an elementary reasoning: if the sentence on line 1 is true then what it asserts is true, but what it asserts is that the sentence on line 1 is *not* true. Consequently, that sentence is not true. But when we write this true conclusion on line 2 we find ourselves repeating the very same sentence. It seems that we are unable to deny the truth of the sentence on line 1 without asserting it at the same time.

* The paper's basic ideas were presented in October 1985, at the Bar-Hillel memorial conference held at Boston University. Further developments were reported in various lectures and invited talks at: Harvard, Princeton, Stanford, UCLA, UC Irvine, CSLI, the Pacific APA meeting, the University of Texas/Austin, and the 1991 ASL meeting. The technical core was presented in the 1988 TARK meeting and published in the proceedings (cf. fn. 3). I have benefited from many reactions. From the many to whom thanks are due I shall only mention David Kaplan, Rohit Parikh (who found a bug in one of the earlier proofs), Charles Parsons, Hilary Putnam, and Brian Skyrms.

0022-362X/92/8905/223–61

Essentially, the puzzle is a variant of what Bas van Fraassen[1] has named "the strengthened liar"—called by some "the metaliar"—which I shall simplify here to *the strong liar*. We might recall at this point that the usual version of the liar paradox consists in the fact that the sentence on line 1, or some analog of it, is neither true nor false, because either assumption leads to contradiction. That version can be resolved by admitting gaps, i.e., by giving up the rule that every sentence must be either true or false. The present paradox is an altogether different kettle of fish. It consists in our seeming inability to express a true conclusion—namely, that a certain sentence is not true—without repeating the very same sentence whose truth we deny, or one equivalent to it.

It is known that the semantic paradoxes can be reconstructed in various modal frameworks (cf. Richard Montague[2]). Indeed, using a knowledge predicate we get an analogous puzzle:

line 1: What is written on line 1 is not known by Jill to be true.

Regarding the sentence, Jill concludes that, indeed, she does not know it to be true. (If she knows it to be true, then it is true; but, given what the sentence says, this implies that she does *not* know it to be true, hence—a contradiction.) Jill writes her conclusion on line 2:

line 2: What is written on line 1 is not known by Jill to be true.

So Jill knows the truth of what is written on line 2, because she has just deduced it, but not of what is written on line 1. A similar puzzle can be made with 'known' replaced by 'believed'. With 'necessary' we get the following version:

line 1: The sentence on line 1 is not necessarily true.

line 2: The sentence on line 1 is not necessarily true.

Again, the line-1 sentence is not necessarily true; because if it were, it would *not* be true and, a fortiori, not necessarily true. This conclusion *is* necessarily true, because we have just proved it. Writing it on line 2, we get a necessarily true sentence, which merely repeats the not necessarily true sentence on line 1.

The moral of all these puzzles is simple. In situations of this nature we should assign truth values not to sentence types but to their

[1] "Presumption, Implication, and Self-reference," this JOURNAL, LXV, 5 (March 7, 1968): 136–52.

[2] "Syntactical Treatments of Modality, with Corollaries on Reflection Principles and Finite Axiomatizability," *Acta Philosophica Fennica,* XVI (1963): 153–67.

tokens. Having concluded that the line-1 sentence is not true, we state the conclusion by displaying another token of the very same sentence. The second token, on line 2, expresses something altogether different from what is expressed, if anything, by the token on line 1. The people who agreed with Jill in the Jack and Jill exchange had no difficulty in grasping this point.

In this respect, modal predicates like 'know', 'believe', and 'necessary' are in the same boat with 'true' and the same remedy is required, namely, to allow different tokens of the same sentence to have different meanings, or—if you want—to express different statements, or different propositions. One should expect the formalisms to differ according to the predicate in question, but the same general framework will underlie them. In the present work, I set up the formalism for truth, thereby providing also the basic framework for the various modalities.

Following the intuition just given, we can see that in general what a token expresses depends on (1) what it *says,* i.e., on the sentence type and (2) on the whole network: on the tokens to which the sentence refers and on the tokens to which *they* in their turn refer, etc. This is what distinguishes the self-referential sentence token on line 1 from its non-self-referential brother on line 2.

I shall propose a formal setup for specifying networks of tokens and an algorithm for assigning truth values. It assigns the sentence token on line 1 the value *GAP*, that on line 2 the value *T* (True), and it yields the same kind of intuitive results in general.

The following "Preview" contains a rather nontechnical presentation of the method, with an illustrative example. It also contains philosophical points and clarifications concerning the proposal's significance, some of which are elaborated in later sections. In section II—where the concept of a *black hole* is defined—a wider perspective is given, which clarifies the proposal's place and motives, within the frameworks of natural-language semantics, and in the philosophy of mathematics. Section III is devoted to the issues of indexicality, levels of truth, and the import of a non-Tarskian semantics. Section IV discusses some points concerning pointers and propositions, and comments on some aspects of the Barwise-Etchemendy work. In section V, previous accounts of the strong liar are discussed: those of Charles Parsons, Tyler Burge, and Brian Skyrms.

I. PREVIEW

I base the formalism on three truth values *T*, *F*, and *GAP*. I shall refer to the first two as the *standard truth values*. The third value, *GAP*, signifies more than mere absence of a standard value. It signifies *recognized failure*. The idea is that we assign tokens, or—as we

shall later see—pointers, the value *GAP* when, in the course of the evaluation, we conclude that they must fail, i.e., that the rules will not enable us to assign them a standard value. We can then go on and use other, "uninfected" tokens to assert of a previously failed token that it is not true, or not false, or that it is neither true nor false.

Those acquainted with Kleene's three-valued logic will be quick to match *GAP* with Kleene's *u*, which stands for 'undefined'. *GAP* is, however, an active value, not a mere "undefined"; an assignment of it can serve as a basis for further assignments of *T*, or of *F*. By making *GAP* an active value we can construct on top of the gap instead falling into it.

Now, the concept of a token is too narrow for the purpose of a general framework. For we might want to refer to sentences without having them displayed somewhere as tokens. Suppose that there is just one sentence on line *n*, then we can use 'the negation of the sentence on line *n*' in order to refer to the negation of that sentence, independently of there being a token of that negation. And, in general, we might speak about the sentence obtained from that on line *n* by any syntactic manipulation, e.g., the substitution of some term by another. Therefore, I use a more general concept, that of a pointer. *A pointer is any object that is used to point to some object.* A token constitutes a special kind of pointer, *it points to the type of which it is a token.*

Pointing is determined by a variety of conventions. Being a pointer is not something intrinsic in an object, but amounts solely to its functioning in a certain role. Any object can serve. If desired, we can regard descriptions as pointers to the described objects. And one can set up a formalism whereby every sentence type is a pointer to itself. The solution to the semantic paradoxes requires, however, that we have many pointers to the same sentence type, so that when a pointer fails another is available. Any object can be pointed to. But since in this paper only pointers to sentence types are needed, I shall reserve 'pointer' for this case, unless the context indicates otherwise.

I find it is useful not to enter, at least at this stage, into the nature of pointers, but to take them as primitives. Thus, I assume a language, a set of objects called "pointers," which have names in that language, and some fixed arbitrary function that correlates with each pointer a sentence (of that very same language) to which it points. A pointer whose name occurs in some sentence can point to any sentence whatsoever (including the sentence containing its

name). All possibilities of direct or indirect self-reference are thus covered.

Only the following additional structural elements are needed for the system. We have to correlate, with each pointer to a compound sentence, pointers that point to its sentential components. Thus, for any pointer p pointing to $A*B$, where $*$ is a sentential connective, there are pointers $p1$ and $p2$, which point, respectively, to A and to B. And if p points to the negation $\neg A$, then $p1$ points to A (and we can, in this case, stipulate that $p2 = p1$). I call $p1$ and $p2$ the *derived pointers* of p. If the language has quantifiers, we also correlate with every p that points to $\forall v A(v)$, or to $\exists v A(v)$, and with every constant term t, a derived pointer $p|t$, which points to $A(t)$ (the result of substituting t for v in $A(v)$).

Note that, if p is a token, then $p1$ and $p2$ can be simply identified with the corresponding subtokens. But with quantifiers, we must go beyond tokens, for in general we cannot identify $p|t$ with a token. For our present purposes, as long as quantifiers are not included, we can take our pointers to be just tokens. The reader may, for the sake of convenience, assume this to be the case. The extension to the general case is straightforward.

The upshot of this approach is a new kind of semantics in which truth values are assigned to pointers and the usual recursive definition of truth is replaced by a set of rules for evaluating networks. Here is an informal description of the algorithm, illustrated by a simple example. This, I think, should give an adequate idea of the method, providing thereby a basis for the discussion that follows. (A full mathematical presentation is planned for a separate companion paper. A concise version containing the formal definitions and the statements, without proofs, of the main theorems can be found in my TARK paper.[3])

Consider the above mentioned system of language and pointers. Assume that the language (based on predicates and, possibly, function symbols) is interpreted, except for two, so-called *truth predicates* (or *semantic predicates*): $Tr(\ )$ for 'true' and $Fa(\ )$ for 'false'. These two predicates take pointer names as arguments. Assuming that p, q, r, etc., are pointers, and that the pointer names used in this discussion are also their names in the formal language, we get sentences of the form $Tr(p)$, $Fa(q)$, etc., where p, q, etc., are pointers to sentences of this very same language.

[3] "Operational Pointer Semantics: Solution to Self-referential Puzzles I," *Theoretical Aspects of Reasoning about Knowledge* [hereafter TARK], M. Vardi, ed. (Los Angeles: Morgan Kauffman, 1988), pp. 43–59.

The goal is an assignment of truth values to all the pointers. It is achieved by an *algorithm*—also called the *evaluation procedure*—which is based on a collection of rules for assigning truth values. They are naturally divided into three groups. The first, the *standard rules* group, consists of rules that correspond to the traditional rules for assigning standard values; they include also rules for the truth predicates.

Standard rules. Pointers to sentences of the form $P(t_1, \ldots, t_n)$, where P is a nonsemantic predicate, get their truth values in the usual way, via the interpretation of the predicates and the denoting terms. Such sentences are called *basic*. Evidently, different pointers to the same basic sentence get the same truth value. It is a standard value determined by sentence type alone. And this holds as long as only standard rules are employed.

Now, if p points to a nonbasic sentence, then, in the usual order of things, its value is determined by the values of other pointers, which, so to speak, are *called directly by* p. Thus, if p points to the disjunction $A \vee B$, then the pointers it calls directly are its derived pointers $p1$ and $p2$ (which, recall, point to A and to B). If one of them gets the value T, so does p (no matter whether the other has been assigned a value, or what that value is). If both get F, p gets F as well. The standard rule for negation, i.e., the "toggling" of the standard truth value, should be obvious. Other connectives are treated by expressing them in terms of $\vee$ and $\neg$.

A pointer, p, to a quantified sentence, calls directly each of the pointers $p|t$ to the substitution instances. The rules for quantified sentences are obtained by treating them as (possibly infinite) disjunctions or conjunctions. For the sake of convenience, I assume that every object in the interpretation's universe is named by some term and that quantifiers are interpreted substitutionally. This is not essential. Quantifiers can be interpreted referentially, at the cost of adding some obvious items.[4]

Finally, a pointer to $Tr(q)$, or to $Fa(q)$, calls directly q; if q gets a standard value, then p gets, in the first case, the same value; in the second case, the opposite value. These are the standard rules for the truth predicates.

The essential part of the algorithm, by which it enhances classical logic, is constituted by two additional groups: the *gap rules* determine the cases of failure, where *GAP* is assigned. The *jump rules* determine assignments of standard values, which are based on pre-

[4] Namely, pointers would point to well-formed formulas (not only to sentences) and truth values would be given to pairs consisting of pointers and assignments of objects to variables.

vious failures. They enable us to use new pointers in order to make semantic assertions about previously failed ones. Before I go into these two groups, here is a simple illustrative example.

Let John, Joan, Jack, and Jill make the following statements:

John: What Jill says is not true and what Joan says is not true.
Jill: What Jack says is true.
Joan: What Jill says is not true.
Jack: Either McX's conjecture is true or what Jill says is false.

For simplicity, we can use here each speaker as a pointer to his uttered sentence. The resulting network is represented in figure 1. The arrows between pointers (the nodes) do not represent pointing, which is a relation between pointers and sentence types, but the above mentioned direct calls between pointers. McX's conjecture, represented as a box, is not spelled out. It might involve a whole network of its own; but I assume, for the purpose of illustration, that it does not refer, directly or indirectly, to any of the four utterances.

Actually, the diagram has been simplified by merging certain arrows and hiding certain derived pointers; e.g., the arrow from John to Jill should be split into an arrow from John to John1 (which points to the first conjunct of what John says), an arrow from John1 to John11 (which points to 'What Jill says is true'), and an arrow from John11 to Jill. Note that the pointed-to sentence types can be read from the diagram, though they are not explicitly displayed.

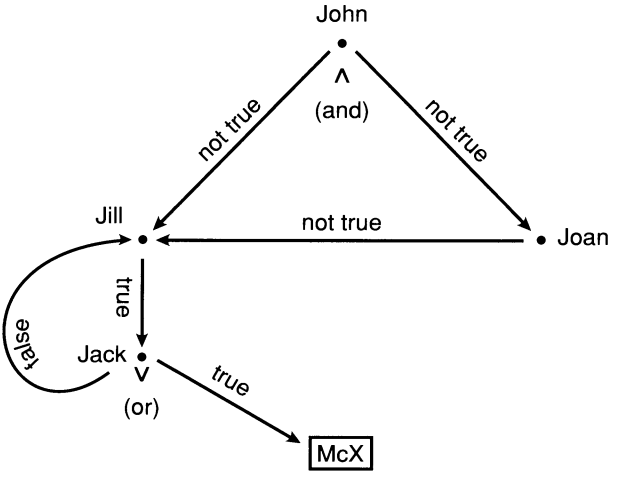

Figure 1

The algorithm is applied recursively. Assume that it results in assigning *F* to McX. This leaves Jack and Jill in a closed unresolved loop. Both get at this stage the value *GAP*. Then Joan, who makes an assertion about Jill and does not belong to the loop, gets a standard value (here a jump rule is employed). Since her assertion is true, she gets *T*. In the same way, the first conjunct in John's assertion gets *T*. But the second conjunct (which asserts the nontruth of Joan) gets *F*, hence John gets *F*. Had McX got *T*, Jack and Jill would have got *T*, while John and Joan would have got *F*.

In the case of the two-line puzzle the network is:

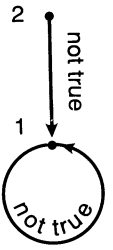

Figure 2

For '*i*' read 'the sentence token on line '*i*'. Right at the beginning we get a closed loop consisting of the pointer 1. It gets the value *GAP*. Then 2 gets the value *T*.

In the four-speaker example, the standard rules for *Tr* and for negation are used for determining the value of John2 (which points to the second conjunct of what John says). Also, the standard rule for conjunctions is used for determining the value of John. The additional rule groups are as follows.

Gap rules. The most important of these is the *closed loop rule,* which states the following: if, in the course of applying the evaluation procedure, a closed unevaluated loop forms and none of its members can be assigned a standard value by any of the rules, then *all* of its members are assigned *GAP* in a single evaluation step.

[A *closed unevaluated loop* is a set, say *L*, of unevaluated pointers (i.e., which have not been assigned truth values), satisfying the following two conditions: (1) unevaluated pointers called directly by any member of *L* also belong to *L*; (2) from each member of *L* one can reach any other member of *L* by one, or more, finitely many direct calls; i.e., by a sequence $p_1, \ldots, p_n$, where $n > 1$, the p_is are in *L* and each calls directly its successor.]

In the four-speaker example, Jack and Jill get their value via the closed loop rule. And in the two-line example this rule applies to the token on line 1, which forms by itself a closed unevaluated loop. In both examples the loops are simple cycles. But, in general, the closed unevaluated loop can be as complex as you can imagine and might consist of any number, finite or infinite, of pointers.

There is also a *simple gap rule* that states: a pointer is assigned *GAP* if all pointers called directly by it have been assigned values, but the pointer cannot be assigned a standard value by any of the rules.[5]

> *Jump rules.* Assume that q points either to $Tr(p)$ or to $Fa(p)$, and that p, but not q, has already been assigned *GAP*. Then the jump rules (for *Tr* and for *Fa*) assign q the value *F*.

To see how the jump rules work assume that an unevaluated r points to $\neg Tr(p)$ and that p has been assigned *GAP*. Then, by the jump rule for *Tr*, $r1$ (which points to $Tr(p)$) gets *F*; and, by the standard negation rule, r gets *T*. Thus, the jump rules provide for the possibility of successfully asserting that a given *GAP* pointer is not true, or that it is not false. *Jump* signifies here true ascent in the metalinguistic hierarchy. In our example, a jump rule (for *Tr*) is needed in order to assign Joan and John1 their values. It is also invoked in assigning a value to the line-2 token in the two-line puzzle.

To sum up, the assignment of *GAP* signifies recognized failure: we decide that the pointers in question fail to evaluate to a standard value. The grounds for such a decision are given in the gap rules. The assignment of *GAP* can serve as a basis for further assignments

[5] Given the standard rules, the effect of this rule is to enforce Kleene's strong three-valued truth tables. A pointer p to a disjunction $A \vee B$ is assigned *GAP* if $p1$ and $p2$ have been assigned values that do not enable us to assign, by some other rule, a standard value to p; this is the case where one of $p1$, $p2$ is assigned *GAP* and the other is assigned either *GAP* or *F*. In other variants, however, based on a different set of standard rules (e.g., using supervaluations), the simple gap rule does not imply Kleene's strong truth tables.

The gap rules determine the assignments of *GAP* that signify recognized failures. But *GAP* can be assigned also by an additional so-called *give-up* rule: if at some stage we are left with unevaluated pointers to which none of the other rules applies, then all of them are assigned *GAP*. Gaps assigned by this rule do not constitute *recognized* failures, but are rather like Kripke's "undefined" *u*. It can be shown that in all networks of finitary type, i.e., where every pointer calls (directly or indirectly) finitely many pointers, the give-up rule is never employed. The aim of more sophisticated versions of the algorithm is to avoid, as far as possible, the employment of such a rule in nonfinitary networks.

of standard values via the jump rules. What is inexpressible in the usual denotational semantics is thus expressible through network evaluation.

Various mathematical results, which I shall not go into here, establish the desirable properties of the evaluation procedure. I shall only describe informally two of the basic results, whose significance, I hope, can be seen:

> *Order independence.* The rules can be applied in any order, to whatever pointers; the (possibly transfinite) sequence of applications results in a unique total valuation which does not depend on the order of applications.[6]
>
> *Limited pointer dissent* (abbreviated as LPD). Two pointers to the same sentence are assigned different values only if one of the pointers is assigned *GAP*. This is also true if the pointed-to sentences are logically equivalent.

The evaluation procedure, also called an "algorithm," has indeed the form of an algorithm: an exact prescription for applying a well-defined set of rules. But this "algorithmic" form should not disguise its true potential. It applies to all networks, including infinite ones of arbitrary cardinality, and it extends both the system of Alfred Tarski and that of Saul Kripke.[7] Each of these is obtained by deleting from my algorithm certain rules: Tarski's truth definition is what we get if we employ only the standard rules for basic sentences, connectives, and quantifiers. Kripke's system is what we get if we employ only the standard rules—including the standard rules for the truth predicates. (In either case, pointers remaining at the end unevaluated are assigned *GAP*, or "undefined.") It can be shown that, if the jump rules are not employed, then a pointer's value depends only on the sentence it points to; truth values would be thus determined by sentence *types* alone.

Note that, in Kripke's construction, both tokens in the two-line puzzle constitute gaps (i.e., remain unevaluated), as do the utter-

[6] A rule application extends a given partial valuation. If only the standard rules are used, then these extensions constitute monotone operators and the order-independence claim follows easily by a fixed-point argument. With the gap and jump rules, however, the operators are not monotone. The claim is nonetheless true, but the proof is much more involved.

[7] "Outline of a Theory of Truth," this JOURNAL, LXXII, 19 (November 6, 1975): 690–716. Repr. in *Recent Essays on Truth and the Liar Paradox,* Richard Martin, ed. (New York: Oxford, 1984). The essays cited in fns. 8, 9, 12, 27, and 64 are also reprinted in this collection.

ances of the four speakers. In the variants of Anil Gupta[8] and Hans Herzberger[9] they are assigned oscillating values. The two major points of difference between my proposal and all the previous ones are, first, the use of pointers and, second, the gap and jump rules.

Recovering the metalanguage hierarchy. There is a precise definition, based on the evaluation procedure, which correlates with each pointer an ordinal number expressing an intuitive notion of "metalinguistic level." The pointers can therefore be stratified, according to their levels, so as to reflect a Tarski-like hierarchy.

Roughly speaking, a pointer's level is the number of *nested applications of jump rules* needed in order to evaluate it. In the four-speaker example, if α is the level of McX, then also Jack and Jill are on level α, whereas John and Joan are on level $\alpha + 1$. We can also refine each of our levels so as to reflect a minor, or local, subhierarchy within it. Roughly speaking, the minor subhierarchy reflects the height of nested applications of the *standard rules for the truth predicates,* which are necessary—once the major level has been reached.

Unlike Tarski's hierarchy, we have a single language with a single truth predicate and the stratification is effected after the fact, i.e., after the assignment of truth values. In this respect it is like the hierarchy obtained via Kripke's construction; as he aptly puts it, the pointers (in his case—sentences) seek their own levels. But his whole hierarchy of grounded sentences corresponds (exactly) to our local subhierarchy of level 0. Real climbing starts where Kripke leaves off. All our nonzero levels are in his model gaps ("undefined").

The construing of truth as a predicate over *pointers* rather than types is related to the metalinguistic hierarchy as follows. Call pointers *dissenting* if they point to the same sentence and have different truth values. We know, by LPD, that of any two dissenting pointers one is assigned *GAP*. It turns out that the non-*GAP* pointer is one level higher in the hierarchy:

> *Higher level dissent* (abbreviated as HLD). For every two dissenting pointers, p_1 and p_2, if p_1 is the *GAP* pointer then $l(p_2) = l(p_1) + 1$, where $l(p_i)$ is the level of p_i.

The significance of LPD and HLD will be discussed in section II. Before proceeding let me clarify certain points, answering thereby

[8] "Truth and Paradox," *Journal of Philosophical Logic,* XI (1982): 1–60.
[9] "Notes on Naive Semantics," *Journal of Philosophical Logic,* XI (1982): 61–102.

possible questions and allaying, perhaps, certain misgivings that might have arisen. Some of these points will be reconsidered, at greater detail, in later sections.

What pointers express. Some might find the idea of assigning truth values to pointers strange; the phrase 'that pointer is true' may sound bizarre. Truth and falsity, so it is argued, have to do with what we *say*. Surely, it is inappropriate to endow some arbitrary objects, used for pointing to sentences, with truth or falsity. To argue thus is to misunderstand my proposal. The phrasing is not essential. It is not strange to speak of the truth or falsity of sentences, meaning thereby sentence tokens; and if it comes to pointers, one could speak of a pointer pointing to a truth, or to a falsity, or indicating, or failing to indicate one. Nothing marks a pointer as such except its function within a global arrangement. But this very function makes it a tool for expression, hence—something on which truth values can be hanged.

As far as the ontology of statements, propositions, meanings, etc., is concerned, my proposal is neutral. A nominalist who bars intensional entities can use something like the evaluation procedure as part of the account of the functioning of language. But the proposal is compatible with seeing pointers (in the contexts of networks) as means of expressing propositions, or statements.[10] On a pretheoretic level, it is not difficult to make sense of "the proposition expressed," or "the statement made" by means of a pointer. For example, what is expressed by the sentence token on line 2 is simply that the sentence token on line 1 is not true, which is to say: *is not assigned the value T by the evaluation procedure.* Although the token on line 1 is of the same sentence type, it fails, because of the loop, to express this. Similarly, in the Jack and Jill exchange, Jack says of himself that he does not express a proposition, but *in this very saying no proposition is expressed* (at least not the proposition indicated by his words, at their face value.) What Jack seems—but has failed—to express is successfully expressed by Jill. If only sentence types are considered, Jack and Jill appear to have stated the same thing. Actually they did not. The source of the illusion is that they employed sentence types that are, in the given conversational context, logically equivalent.

The following can perhaps add clarification. Two intuitive theses about truth lead to the *strong liar:*

[10] Cf. fn. 55 concerning 'propositions' and 'statements'.

(T1) If the sentence 'x is not true' is true, then x is not true.

(T2) If x is not true, then the sentence 'x is not true' is true.

Here 'x' ranges over sentence types and quotes are interpreted as Quine's semiquotes (i.e., the quoted sentence, which is x's value, is to replace 'x'). We get the strong liar when x is, or is equivalent to, 'x is not true'. But if we construe 'true' as a predicate of pointers, then 'x' should range over pointers and (T1) and (T2) should be modified by replacing 'the sentence' by 'any pointer to the sentence'. We can still maintain (T1) in its new form, but we have to qualify (T2) by rewriting it as:

(T2*) If x is not true then any *unfailed* pointer to 'x is not true' is true.

The paradox is thus blocked by allowing pointers—in particular, tokens—to fail, and by giving up (T2) for failed tokens of 'x is not true'. If we employ the analogous strategy for the case of types, and x is a liar sentence, then it is the *type* 'x is not true' which fails. Thus, we cannot use it to say that x is not true. Further observations concerning pointers and propositions are made in section IV.

An essentially non-Tarskian semantics. In Tarskian semantics, truth is a property of sentence types; tokens as such are not considered. That tokens play a central role in natural language, due to the abundance of indexicals and demonstratives (words like 'I', 'you', 'now', 'this', 'that') is, of course, very old news. These terms, having highly variable context-dependent denotations, cause truth value to vary according to token.

But it would seem that the Tarskian framework can be adjusted, or rather supplemented, so as to handle these phenomena. Having replaced the items of varying denotation by others, whose denotations—within the considered discourse—are fixed, we can apply Tarskian semantics. To use an admittedly simple example, transform 'He is tall' to 'X is tall', where 'X' is an appropriate context-insensitive name or description of the indicated person, and evaluate the result as a sentence *type*. If needed, a further reduction can be used to eliminate the temporal indexicality of 'is'. The truth predicate that is defined over tokens is thus reduced to a predicate defined à la Tarski over types.

The present situation is altogether different. The predicate defined by the evaluation procedure is *irreducibly over tokens,* or—

more generally—over pointers.[11] Here, as I shall show, differences in truth value are not traceable to terms that behave like indexicals, or have ambiguous denotations, which occur in the sentences.

Attempts, like Tyler Burge's,[12] to resolve the semantic paradoxes by construing 'true' as an indexical cannot but fail to achieve that goal. Indeed, as the analysis will show, Burge's account puts all the burden of work on a rather obscure mechanism of implicatures, leaving indexicality idle. (The general argument concerning indexicality is laid out in section III; the more specific discussion of Burge's work is in section V.)

Much more significant than the failure of indexicality is the change from compositional semantics, whereby the meaning of a sentence is "composed" of the meaning of its parts, to a radically different one, in which tokens, or pointers, are evaluated globally. The present proposal shows that this essentially different picture can be embodied in a formalism as precise as Tarski's truth definition for formal languages.

An argument propounded by Donald Davidson has it that the ability of finite beings—such as humans—to use a potentially infinite number of sentences is to be explained by assuming a finite number of rules,[13] whereby meanings of compound expressions are determined by the meanings of their components. A setup of this nature resembles Tarski's truth definition. That something of this kind underlies a great deal of the functioning of language cannot be doubted. But it is equally true that a great deal lies beyond it. If human ability for controlling a potential infinity is to be modeled by a formal device—a plausible working assumption, though far from an axiom—then the appropriate concept is that of an *algorithm.* A finitely axiomatized compositional semantics amounts to a rather special kind of algorithm. Appealing as the model is, it is more than plausible that the processing of language, on the level of meaning, involves also algorithms of quite a different nature. I shall say more on this score in section III.

[11] Formally, one can of course redefine "type" by stipulating that tokens that get different truth values (e.g., those on lines 1 and 2) are to be considered of different type. Isaac Levi observed that such a stipulation might not be bizarre, given that the token-type distinction is not always clear-cut and that homonyms might be considered as different types, because of their different meanings. But if we do this, the very definition of "type" presupposes the assignment of truth values to tokens; in this sense, the truth predicate is still irreducibly over tokens.

[12] "Semantical Paradox," this JOURNAL, LXXVI, 4 (April 1979): 169–98.

[13] On the syntactical level, Noam Chomsky had argued in this way for an innate language-processing mechanism.

Truth, falsity, and GAP. In the pointer formalism, the predicates representing truth and falsity figure as two primitives. The evaluation rules relate truth and falsity, through negation, as follows. *A pointer to the negation* ¬A gets the value T, or the value F, iff the derived pointer to A gets the opposite value. Thus, our concept of falsity accords with the well-known thesis, according to which a sentence is false iff its negation is true. It is possible to set up an equivalent formalism that uses only *Tr*, or only *Fa*, as a primitive.[14]

Let 'false$_1$' denote falsity in the sense employed here. Another way of relating falsity to truth is given by 'false$_2$'. *A sentence is false$_2$ just when it is not true.*[15] Consequently, every sentence is either true or false$_2$. When one reads 'false' as 'false$_2$' there is a miraculous disappearance of truth-value gaps. Actually, they are there, thinly disguised and easy to uncover. A sentence is a gap if and only if both it and its negation are false$_2$.

As the last equivalence suggests, easy translations are available from one 'false$_i$' terminology to the other, making the two equivalent for purposes of expression. The present formalism can be recast in terms of 'false$_2$', at some cost of additional structure.[16]

The false$_2$ interpretation might remind one of cases, arising in Bertrand Russell's theory of descriptions, where sentences and their "negations" come out false; e.g., (i) 'The present king of France is bald' and (ii) 'The present king of France is not bald'.[17] But the similarity is only apparent. The falsity of (ii) is caused by the narrow

[14] We can define the evaluation procedure for a subsystem based on a single truth predicate. But to have the expressive power of our present system, we need a way to express both $Tr(p)$ and $Fa(p)$; e.g., both are needed in order to say that p is a gap ($\neg Tr(p) \wedge \neg Fa(p)$). This requires an additional structural item for pointers: a negation operant that associates with every pointer p a pointer, say $neg(p)$, to the negation of the sentence pointed to by p. We can then translate $Fa(p)$ into $Tr(neg(p))$. (We should stipulate that $neg(p)1 = p$ and that if p points to a negation then $neg(p1) = p$). Note that, in general, $neg(p)$ will not be a token, even if p is. The present formalism avoids the need for a negation operant; hence, unless we consider quantifiers, we can assume that all our pointers are just tokens.

[15] Such a reading of 'false', for the case of certain atomic sentences, was suggested in David Kaplan's "What is Russell's Theory of Descriptions?" *Physics, Logic, and History,* W. Yourgrau and A. D. Beck, eds. (New York: Plenum, 1970). The same reading also turns up in Jon Barwise and John Etchemendy's *The Liar: An Essay in Truth and Circularity* (New York: Oxford, 1987). But the issues there, to be discussed in section IV, are complicated by the treatment of negation.

[16] We shall need a negation operant for pointers; cf. fn. 14.

[17] I am ignoring here the more radical Russellian position according to which definite descriptions are incomplete symbols to be eliminated, in the final account, altogether.

scope of the negation. True negation—one that applies to the whole sentence—*does* convert falsity to truth: (iii) 'It is not the case that the present king of France is bald' comes out true. Much of the appeal of Russell's theory is due to its ability to distinguish between local negation, as exemplified in (ii), and true negation, as exemplified in (iii). There are no gaps here. Acceptable or not, Russell's theory rests squarely within the framework of classical two-valued logic.

The present situation is altogether different. The nontruth of the line-1 sentence (in the two-line puzzle) does not hinge on the restricted scope of 'not'. Making that scope as wide and as unambiguous as possible does not make the slightest difference. Consider for example:

line 1: *It is not the case that the sentence on line 1 is true.*

To classify this last sentence as false means to classify as false both a sentence x and 'it is not the case that x'.[18] And while it is conceivable for both x and 'it is not the case that x' to fail, it is inconceivable that both be false—unless 'false' is just another mark for failing. That is, 'is false' is to be read here as *is nontrue,* or—still better—*fails to be true,* or *fails to express a truth.* To what extent this subsumption of failure under falsity reflects ordinary usage is a question for the lexicographer.

A two-fold classification of all sentences into "true" and "false" may serve to make a philosophical point.[19] But it should be clear that by no means does it signify a two-valued logic. There is more to two-valued logic than there being just two "truth values." The two values should underly the semantics of the sentential connectives. That they should work well with negation is the minimum requirement.

To sum up, the construing of 'false' as 'false_2' has no bearing on the existence of gaps, but is merely a terminological grouping of falsity (i.e., truth of the negation) together with other kinds of failures. There is a moral in this concerning propositions. Either we say that liar-type sentences express no propositions, or we make place for propositions that constitute truth-value gaps.

The liar and the truth teller. The algorithm described so far treats the truth teller (a sentence asserting its own truth, e.g., 'What I am

[18] Take as x the sentence 'the sentence on line 1 is true'. If the sentence on line 1 is false, so is x and so is 'it is not the case that x'.

[19] As was done, for example, by Michael Dummett, in "Truth," *Proceedings of the Aristotelian Society,* LIX (1959): 141–62.

saying now is true') in the same way as the liar; both get *GAP*. Our framework does, however, provide means for distinguishing between the two. The distinction is made by considering consistent assignment of truth values, or what I call in the technical paper *self-supporting valuations*.[20] There is a total self-supporting valuation that assigns the truth teller (token) the value *T*, and there is one that assigns it *F*. But every self-supporting total valuation must assign the liar (token) the value *GAP*.

Different variants of the algorithm. The present variant of the algorithm leads to a truth-value assignment that conforms to Kleene's three-valued truth tables (cf. fn. 5). Some may, however, prefer to assign *T* to all pointers to tautologies. This can be achieved by using a variant in which the set of standard rules is replaced by another set, based on *supervaluations*. In the usual supervaluation method—which applies to sentence types—a sentence is given a standard value whenever it gets this value in all possible complete standard extensions of the partial valuation. This idea can be adopted to assignments of values to pointers; I defer details to the technical paper.

It is noteworthy that, to get the supervaluation variant, only the standard rules are changed. The gap and jump rules remain the same.[21] In other variants as well, changes can be done "locally" within this or that group of rules. This illustrates the modular character of these groups.

More sophisticated variants, not to be presented here, incorporate additional gap rules. These rules introduce grounds for discriminating between pointers. They enable early assignments of *GAP* to some of the pointers, thereby salvaging the remaining ones that would have otherwise been all members of the same closed loop. The total effect is to reduce the assignment of *GAP* and to make more pointers available as means of expression. Such variants aim at achieving higher degrees of self-contained systems for some highly expressive languages—a goal that will become clearer in the next section.

II. BLACK HOLES

Suppose that we assign truth values to sentence types. Consider *any* notion of sentence equivalence, which preserves truth value and

[20] Roughly speaking, a self-supporting total valuation is one that is closed under rule applications: applying to it the rules of the algorithm yields the same valuation.

[21] Changing the standard rules can affect the applicability of the gap rules, because a gap rule can apply only if other rules cannot. The gap rules themselves, however, need no change.

which satisfies the following two mild conditions. (i) Each sentence is equivalent to itself. (ii) If '. . .' and '——' are terms referring to equivalent sentences, then '. . . is not true' is equivalent to '—— is not true'.[22] Let us now extend the original two-line text as follows:

line 1: The sentence on line 1 is not true.

line 2: The sentence on line 1 is not true.

line 3: The sentence on line 2 is not true.

line 4: The sentence on line 3 is not true.

⋮

line n + 1: The sentence on line n is not true.

The sentences on the first two lines are the same, hence—equivalent. And, by (ii), the equivalence of the sentences on lines $k - 1$ and k implies that of the sentences on lines k and $k + 1$. Consequently, all the sentences are equivalent. Since the sentence on line 1 is not true, the sentence on line 3 is not true either, but what I have *just stated* is not true, because it is the sentence on line 4 (or an obviously equivalent reformulation of it), and also *this* last statement of mine is not true, because it is the sentence on line 5, etc. None of these sentences can be successfully asserted, because none of them is true; but again I find myself slipping into nontruth: what I have *just* said is not true for it obviously includes the conjunction of these very same sentences; and also *this* last assertion is not true, and so on ad infinitum.

The innocent looking gap has developed into a black hole that sucks into itself every sentence asserting that some sentence in the hole is not true.[23] Formally, a hole hierarchy is definable as follows: S is a *0-hole* if it is a gap and, by induction: S is an $n + 1$-*hole* if it is an n-hole and 'S is true' and 'S is false' are gaps. Then S is a **black hole** if it is an n-hole for all n. (Here and elsewhere, quotes are to be read as semiquotes, whenever this is obvious from the text.) We shall

[22] That some such minimal equivalence notion should obtain is inherent in the very assignment of truth values to sentence types. The giving up of (ii) would signify that something other than sentence type is involved in determining truth value.

[23] So far we assumed an equivalence satisfying the above mentioned conditions, and the following rules that imply that the line-1 sentence is a gap. (1) If 'S is true' is true then S is true. (2) If 'S is not true' is true then S is not true. (3) If the negation of a sentence is false then the sentence is true. We shall also assume, in the rest of the discussion, that if the negation of a sentence is true then the sentence is false and the equivalence of 'S is false' and 'not-S is true'.

use '*hole*', unqualified, to denote a 1-hole. Thus, we cannot assert of a hole that it is not true, or not false, or a gap (neither true nor false), without this second assertion being itself a gap.

Of course, syntactical properties can be asserted of *any* sentence. But no information about the truth value of a hole can be conveyed directly. And as for black holes, they are semantic untouchables. No semantic information about them can be conveyed in any way, be it as indirect and across as many layers as one may wish.

Actually, the arguments produced so far in the case of our example imply only that the sentence on line 1 is a "semiblack hole"; we have only iterated '. . . is not true' but not '. . . is not false'.[24] This is, however, bad enough. And it becomes a fully fledged black hole, under some additional very natural assumptions. In particular, it is easily seen that *every* gap is a black hole if we assume that *S* and '*S* is true' are equivalent, under an equivalence relation satisfying Leibniz's substitutivity law.[25] (Note that this is *not* the Tarski biconditional: '*S*' is true $\leftrightarrow$ S, but only the requirement that '*S* is true' and *S* should get the same truth value, including *GAP*.) It is difficult to see how this equivalence can be avoided if truth values depend only on sentence types. Indeed, it holds in the system of Kripke (*op. cit.;* discovered independently around 1974 by Walther Kindt[26]), in its Martin-Woodruff[27] forerunner and in the variants of Gupta (*op. cit.*) and of Herzberger (*op. cit.*). Every truth-value gap is a black hole in these systems.

When truth values depend only on types, black holes are thus bound to occur on the most elementary levels of language. (Think how little is needed to produce sentences like the sentence on line 1.) Within such a system the little game I have just played, of repeatedly refuting myself by trying to assert the "unassertable," signifies a genuine metaphysical mystery. And yet the trick is quite transparent and we, Jack and Jill included, have no trouble at all asserting

[24] Our minimal assumptions (cf. *ibid.*) do not imply that 'The sentence on line 1 is false' is a gap. This can be shown by constructing a model—to be sure, a highly pathological one—satisfying the minimal assumptions, in which this last sentence is false.

[25] That is to say, the equivalence of S_1 and S_2 entails that of 'S_1 is true' and 'S_2 is true' and similarly for 'false'. Actually, for the sentence on line n to be a black hole a weaker assumption than the equivalence of *S* and '*S* is true' will suffice. E.g., for $n = 2$ the assumption is that '*S* is true' is equivalent to '*S* is true' is true'. For larger n one iterates 'is true' accordingly; the assumption becomes weaker with each iteration.

[26] "Introduction of the Truth Predicates into First-order Languages," in *Formal Semantics and Pragmatics for Natural Languages,* Guenthner and Schmidt, eds. (Cambridge: Reidel, 1979).

[27] "On Representing 'True-in-L' in L," *Philosophia,* v (1975): 213–7.

this sort of "unassertable" by employing another token of the very same troublesome sentence. Language usage involves a mechanism for solving the black-hole dilemma. And if our goal is to model natural language, or a basic aspect of it, then the elimination of black holes—at least those of the simpler varieties—should be an overriding concern. Failure in this respect means that we have failed to capture something very essential to the functioning of language.

Although Kripke notices the failure, he plays it down, not giving the issue the central place it deserves.[28] "The sense [so he writes] in which we can say, in a natural language, that a liar sentence is not true must be thought of as associated with a later stage in the development of natural language, one in which the speakers reflect on the generation process leading to the minimal fixed point."

Indeed, the nontruth of the sentence on line 1 hinges on a sort of metareasoning. But this does not place the assertion on line 2 at a "later stage in the development of natural language." Certainly not at a stage that comes after the construction of the minimal fixed point, an inductive process that moves through all recursive ordinals! The fact that, after carrying out this transfinite construction, we are still left with the elementary problem of making sense of what Jack and Jill said shows that, far from having completed a home run, we are still at first base.

The basic (and simplest) version of my proposal eliminates the holes in all situations of finitary type—those in which every pointer calls only finitely many pointers.[29] To this type of network, which I call *locally finite*, belong all the tangled loops that can arise when finitely many people make assertions about each other, provided that altogether finitely many assertions are involved. In particular, it contains the examples discussed in the literature. The absence of holes is formally expressed by the following result, which holds for locally finite networks under quite general conditions:[30]

> If p has the value *GAP*, then there is a pointer q to the sentence $\neg Tr(p) \wedge \neg Fa(p)$ (i.e., the sentence asserting that p is a gap), such that q has the value T.

We can thus successfully assert, of any gap, that it is a gap. Another result, which applies under the same conditions, states that

[28] This point has been nicely put by Burge (*op. cit.*).

[29] A pointer p calls a pointer q if one can reach q from p by one or more, finitely many, direct calls; cf. the "Preview" where the algorithm is described.

[30] The required condition is that there are infinitely many independent pointers to each sentence type, where "independent pointers" are those which have no common subpointers, a subpointer being the natural generalization of "subtoken."

every sentence has a pointer to it that has a standard (T or F) value. (Recall that, by LPD, pointers to the same sentence, which get standard values, always agree in value.)

In certain situations of infinitary type, when the language is sufficiently expressive, holes, even black holes, appear. Of these, some are eliminable by more sophisticated versions of the algorithm and it appears that further improvements are possible. It is far from clear how far one can go toward the elimination of holes. In the context of mathematical languages, the elimination of holes will bring us near to what one might consider a "universal language"—a self-contained system, one that includes its own semantics. For this reason alone it is highly plausible that rich languages must have holes. Note, however, that we have no formal proof of this. The standard argument from the liar is no longer valid for setups of the kind proposed here. But perhaps the question of proof should not even be raised at the present stage, since it is not clear what the present proposal might, in further developments, encompass.

In the broader perspectives of natural language "hole-like" phenomena are bound to occur. Some discourses take us to the edge of meaning, whereby we are tempted to express the inexpressible or to think the unthinkable. Holes are perhaps the inevitable price for a powerful language capable of evolving. Indeed, some situations can be described only by saying that we have changed the language, or that we have altered the very fabric of our conceptual framework. But I do not think that the Jack and Jill exchange calls for such a description. Here we can at least tidy up the more elementary levels of language.

A comparison with set theory comes naturally to mind. The analogy between the semantic paradoxes and those of set theory is well-known. Russell grouped them together and proposed the same kind of treatment.[31] His solution is based on a rigid predetermined classification of sets, which makes room for quantifying over sets of the same type, but not over sets in general. The hierarchy is incorporated into language itself. Axiomatic set theory, say ZF, has disposed of the built-in hierarchy, replacing the aggregate of concepts *set-of-type-i* by a single concept of set.

Russell's hierarchy of sets parallels his hierarchy of propositions, of which Tarski's hierarchy of metalanguages is a close relative. The

[31] More recently, the analogy has been emphasized by Parsons. Both types of paradoxes have been discussed by me as cases of what I called an infinity paradox: a contradiction that results from an attempt to construct self-containing systems. Cf. my "Paradoxes of Infinity and Self-application," *Erkenntnis,* XX (1983): 131–55.

representation of Tarski's hierarchy in a formal system yields an aggregate of truth predicates $true_i$, where i ranges over some list fixed in advance, but no single *true*.

In mathematical setups the problem is not acute. First, because 'true' is used there in a restricted way, which bars self-reference. Second, because, working inside set theory, we can assign truth values to mathematical statements—as long as these statements are about structures that are *sets*. A closer look at the second point shows that the metalinguistic hierarchy has been subsumed under the hierarchy of sets (the one obtained by iterating the power-set operation). Problems resurge when the structures in question are no longer sets; in particular, when truth in "Cantor's universe" is considered. In this respect, axiomatic set theory has not solved the problems arising out of the set-theoretic paradoxes. It has only shifted them to the theory's fringes, where they are of no concern, except to set theoreticians who investigate the theory's borders (for a detailed discussion, cf. *ibid.*).

This perspective marks a clear goal for the semantics of 'true'. A semantics of formal languages—containing their own truth predicate—should do for the concept of truth what set theory, such as ZF, does for the concept of set. By minimizing the black-hole phenomena we are shifting the problematic points, where the system's inadequacies show, to the less accessible parts of the system.

Whereas set theory is informed by mathematical thought, pointer semantics draws directly on insights derived from natural language. (Mathematics itself, however, is an outgrowth of thought manifested in ordinary language.) Yet it results in systems that are as formally precise as any piece of mathematics. The present work can be seen as a first step that eliminates black holes at the basic linguistic level. As mentioned earlier, black holes reappear, in highly expressive languages. These are cases in which the pointing relation itself is describable in the language and Gödel's techniques are available. Attempts to eliminate such black holes lead to more sophisticated versions of the algorithm and to some intricate, interesting questions. Thus, the formalization of mechanisms inherent in linguistic practice also gives rise to foundational questions of logic.

III. TOKENS, TYPES, AND COMPOSITIONAL SEMANTICS

Let me start with a commonplace observation. When a child learns a language he encounters first of all tokens of linguistic entities. The child acquires the language by learning the effects of token display and by participating in the token-display exchange. This observation does not prove that tokens are prior (in some nontrivial sense) to

types; for the ability to learn the language is preconditioned by the ability to recognize token similarity and, thereby, to classify tokens according to type. The observation is made in order to underline the obvious fact that there are many ways of passing information by means of an interactive game-like process. A useful description of the game must, inevitably, ignore through abstraction various aspects; otherwise, we shall be submerged under sheer detail.[32] Of course, we may ignore only inasmuch as it does not restrict us seriously in accounting for what is going on. There are thus two conflicting requirements: that the account we give be sufficiently simple, and that it be adequate for handling a broad enough class of applications. Usually there is trade-off between the two.

The fixing on sentence types, as on the basic truth-carrier units, is a natural move. It leads to an enormous simplification, and it clears the way for compositional semantics: a semantics whereby linguistic entities are analyzed as complexes whose meanings derive in a systematic way from that of their components, via certain—relatively few—recursive rules. The truth value of a sentence is thus determined recursively; the context in which the sentence occurs can be completely ignored.

A description of this kind fits accurately formal languages constructed by logicians; which is no wonder, since it is the very principle on which such languages are constructed. It does not fit natural languages, where context dependencies abound, due to indexicals, demonstratives, and ambiguous terms. Nonetheless, it reflects a very essential aspect of natural language. For if the dependency of truth value on context can be traced to the shifting denotations of certain context-sensitive terms, then once these denotations have been determined—how, is another matter—we can derive truth values by means of the above mentioned recursive rules.[33]

Call two tokens *dissenting* if they are tokens of the same sentence type, but are accorded different truth values. Then the account just outlined amounts to the following. If p and q are dissenting tokens, then the sentence type should contain terms, say $t_1, t_2, \ldots, t_n$, whose references differ on the two occasions. If we are to replace each t_i by a suitable t_i', which denotes on both occasions what t_i denotes in the context of p, the resulting tokens, say p' and q', would

[32] On a more fundamental level, the very concept of description means that many details have been ignored.

[33] Such a view underlies Davidson's semantic program.

have the same truth value, namely, that of p. We can call such an elimination of context dependency *local disambiguation.*[34]

For example, when 'I am tall' is uttered by two speakers, only one of whom is tall, the conflict in truth value is traceable to the different denotations of 'I'. The dissent can be resolved by using proper names. If needed, a name can be introduced by stipulation: "Let S1 be the first speaker." In the counterfactual situation, where the two speakers use 'S1' instead of 'I', the two utterances have the same truth value: that of the first speaker's original utterance.[35] Or consider two tokens of 'tanks are never made of wood', one in a treatise on weapons, the other in an essay on storing methods. The difference in truth value is due to 'tanks', and is eliminable by substituting the term by 'armored vehicles on caterpillar treads'.[36]

The picture is radically different when every term contained in the sentence type has the same denotation, on both occurrences in two dissenting tokens. Here, a truth-value difference cannot be reduced to denotational shifts of sentence parts. It is accounted only by the token's role as a unit within some global arrangement. And such, I hope to show, are the cases deriving from the self-referential usage of 'true'. The argument for the failure of indexicality shows, in fact, the failure of the usual compositional semantics.[37]

First, I take it that in the two-line puzzle, and in similar cases, we *do* have token dissent. We do want to say that the sentence on line 1 is not true but the one on line 2 is; we do side with Jill against Jack.

[34] The "context insensitivity" of the terms t_i' is not necessarily absolute. It only means that they have the same denotations throughout some wider context, which includes that of p and that of q.

The terms t_i can be either noun phrases or predicates. I assume here, for the sake of simplicity, extensional contexts. But this is not essential. Local disambiguation makes sense in intensional settings, provided that we apply some stronger equivalence notion. The term t_i' should be intensionally equivalent to t_i-in-the-context-of-p.

[35] Instead of changing, counterfactually, the original utterances, we can let the two speakers reutter the sentence, with 'I' replaced by 'S1'; the natural assumption being that, in the given circumstance, an utterance and its repetition by the same speaker are equivalent. This strategy does not work if the indexical is time-sensitive, unless the repetition is sufficiently close so as to make the time difference insignificant.

[36] In lumping together indexicals with cases of ambiguity, I am not glossing over what distinguishes the first. The "ambiguity" of 'I' means that its denotation shifts according to context. Because the shift is systematic, 'I' can be regarded, in one sense of 'meaning', as having a fixed meaning: it always denotes the speaker. Where no such systematic account is available, we must rest content with what informal pragmatics will provide.

[37] The qualifier 'usual' is added here, in order not to rule out other types of compositionality. For example, one may consider composition of networks.

Or, to use Buridan's example,[38] if Plato says at time *t*: "What Aristotle says at time *t* is not true," and Aristotle says, at the same time, the same about Plato, then both sayings are not true; but if a third party were to assert at that time that what Plato (or what Aristotle) says is not true, *his* saying would be true.

Evidently, 'The sentence on line 1' in the two-line puzzle refers in both tokens to the same object, be this sentence type or sentence token. The only candidate whose meaning, or denotation, might have changed is 'true'. But note that the argument for the nontruth of the line-1 sentence employs 'true' in the very same sense of the 'true' occurring on line 1.[39] The argument simply shows that the sentence on line 1 does not fall under what 'true' *in this very sentence token* denotes. And if *this* is what we write on line 2, then the 'true' on the second line has the same denotation as that on the first. (Indeed, if the 'true' on line 2 is construed as being on a higher level, then the sentence on line 2 is false!) The difference in truth value is not caused by some shift involving the 'true' that occurs in the sentence, because no shift takes place.

Interestingly, the analysis just offered accords with that given by Burge in his discussion of a similar puzzle (*op. cit.*). Phrased in the present terms, his conclusion is that the tokens of 'true' on the two lines should be construed as standing for the same truth concept. The difference in truth values is traced by him to different implicatures involved in evaluating the sentence. (As will be shown in section V, some unspecified implicature mechanism is implicitly presupposed by Burge throughout, while his indexicality account does no actual work.)

Although I find the foregoing argument sufficient for my purpose, I shall supplement it by another, which is, I think, of independent interest. Again, I consider the two-line puzzle. I shall not appeal to how we derive the conclusion about the line-1 sentence, however, but proceed in a more general way. Let the 'true' on line 1 be changed to a nonindexical 'true*' that denotes once and for all what 'true' in the original context denotes.[40] We thus freeze 'true' in its line-1 context. The freezing, one feels, should not affect anything

[38] Quoted in Burge, op. cit.

[39] The argument rests on (T1). If '*x* is not true' is true, than *x* is not true. (And in this phrasing the various occurrences of 'true' denote the same truth concept, whatever that may be.) The familiar argument now runs thus. If the line-1 sentence falls under what 'true' in the line-1 context denotes, then by (T1) that sentence does not fall under what 'true' in the line-1 context denotes.

[40] Either the change is effected by erasing and rewriting, or we imagine a counterfactual situation in which the second version is written instead of the first.

that has to do with truth. Saying, while addressing Jack, 'you are tall' and saying 'Jack is tall' come to the same.

There is yet a subtle complication: By changing 'true' to 'true*' we have altered the reference of 'the sentence on line 1'. There is thus the additional change in what that noun phrase denotes. Nonetheless, as far as truth or nontruth is concerned, this added difference should not matter. It might have, had the sentence said something about lexical properties. Changing

> You are tall and this last utterance of mine is short.

said by Jill to Jack, to

> Jack Horatio MacWiltington the Third is tall and this last utterance of mine is short.

makes a difference. But with 'true' instead of 'short', the two come to the same.[41]

Since the original line-1 sentence is not true, its frozen version is not true as well. And since we can assume that truth* satisfies (T1) of the "Preview," this being a minimal requirement of any truth concept, it obviously follows (cf. fn. 39) that the frozen version is not true*. Having written this last conclusion on line 2, we get another token of the frozen version, this time a true one. We have eliminated the indexical context dependency, but we get back two dissenting tokens. Local disambiguation fails.

Thus, indexicality, or anything like it, cannot account for truth-value changes accruing through the self-referential usage of 'true'. This is not to say that the picture of truth as involving meaning variations is altogether wrong. Something like truth-level shift does take place. And it is important to disentangle this true aspect from the wrong idea that 'true' is indexical.

We can, in fact, view the judging of the sentence on line 2 as taking place on a higher level than the judging of the sentence on line 1. But the shift to higher level is determined directly by the place of the token in the network, not by the mediation of context-

[41] The analogy is not altogether complete. With the changes in two separate conjuncts: 'you are tall' and 'this last utterance of mine is true', we can argue formally by considering first one, then the other. Such a procedure is not at hand when the changes are fused in the same atomic sentence. But this cannot provide any ground for taking the change of 'true' to 'true*' as a cause for truth-value change. Whoever has misgivings on this score may consider a variant of the liar in which the sentence is, so to speak, split: on line 0 we have, 'The sentence on line 1 is true', and on line 1, 'The sentence on line 0 is not true'. Can the freezing of 'true' on line 1 make a difference? I cannot see that it should matter for the sentence on line 0 and, by consequence, for the sentence on line 1 either. And neither should an additional freezing of 'true' on line 0.

sensitive terms in the sentence, because all the terms retain their original meanings. We have seen that the evaluation procedure provides a formal reconstruction of the idea of levels. The line-2 token is seen to be one level higher, because its evaluation takes one application of a jump rule, while no jump rule is needed for evaluating the first. If one wishes, one can get, as a corollary of the level assignment to pointers, a level assignment to 'true'. The prescription is simple. Assign to the occurrence of 'true' in a token of 'true(p)' the ordinal number assigned to p; for this is the level that the evaluation of p reaches. In particular, the 'true' on both lines is assigned level 0—the number assigned to the line-1 token. But were we to write on a third line 'the sentence on line 2 is true' (or 'not true'), *that* 'true' would get level 1.

Occasionally, truth levels are indicated by attaching subscripts to 'true'. This would be harmless, provided that we are not misled to see in these subscripts, or indices (as they are also called), some form of indexicality. The indices play no role in determining truth value. They come after the fact, as an annotation to a finished process, an underlining of certain aspects of the evaluation.

While the reference of 'I' is picked through the indexical mechanism from the multitude of humans, nothing of the kind obtains in the case of 'true'. There is no preexisting multitude, or hierarchy, of truth concepts, out of which we pick the concept we invoke. The hierarchy is created by the evaluation process itself, by the very act of judging, by the linguistic exchange.

The functioning of 'true' exemplifies that aspect of linguistic activity whereby sentence tokens function as the units of the exchange. The meaning of a token is not reducible to what the sentence says, even when the denotations of its terms have been fixed. It derives also from the token's particular place in a global setup.

The principle, under other forms and on different scales, is quite common. An event, an assertion, a work of art might mean whatever it means by virtue of its place in a global system of similar events, acts, or works. This is particularly true in the artistic domain. Duchamp's famous urinal refers to the tradition of art exhibitions. The reference, though not formal and not explicit, is there. For the place of that particular display within the historic network of art is what endows Duchamp's exhibit with artistic meaning. Moreover, it would be erroneous to derive this meaning from the urinal itself, as if it were an "indexical" that indicates different things in different contexts; quite the contrary. That object means or connotes, within the displaying event, whatever a urinal means or connotes in everyday life. Only by retaining its ordinary, undignified status, can it

cause the surprise, the outrage, and the joke that constitute the aesthetics of the display.

We can now take stock and assess the extent to which pointer semantics departs from a type-based one. We can use as a sort of measure the extent of token—or pointer—dissent caused by the functioning of 'true' (and not by indexicals, or ambiguities of sorts). In this respect, the formal system of the present proposal provides a good view, as it excludes indexicals, or any kind of ambiguity. Now, the results LPD and HLD mentioned in the "Preview" imply a very simple picture, perhaps surprisingly simple—given that it holds for full first-order logic and arbitrary pointer networks. Whenever pointers dissent, one of them has the value *GAP* and the other, which has a standard value, is one level higher in the linguistic hierarchy. Informally, we can say that departure from type-based semantics occurs just when a pointer fails to express that which another at a higher level succeeds in expressing.[42] Thus, departure takes place when a necessary ascent in the linguistic hierarchy cannot be accommodated within Tarskian semantics or any of its offshoots—and here I include Kripke's system and all its variants—thereby forcing us to change the language. To encompass it within a single language, which is what the present framework does, requires a price and the results show that we are paying the minimum.[43]

While the semantic paradoxes have focused a great amount of research, resulting in an extended and often sophisticated literature, they have not been at the center of the research program of natural-language semantics. This is understandable, given the tangle of problems encountered in the rather elementary stages of the investigation. To give the semantics of 'true' while we are far from clear on that of 'event', or that of mass terms, might seem a luxury.[44] But the tendency to regard the problems raised by the paradoxes as something esoteric, to be handled later (if at all) when more basic

[42] In this respect, all such cases are like the simple two-line example. But the sentence in question can be as complex as any first-order sentence and the failure can be caused by an extremely intricate loop. Moreover, it can occur at any level, the ordinal of which can be of any cardinality not exceeding that of the total network.

[43] I should add, however, that it is conceivable that in more sophisticated variants of the algorithm LPD will fail. I see no reason why LPD, pleasing as it is, should be imposed as a prerequisite on evaluation procedures in general. With indexicals present, different tokens of the same sentence may get different non-*GAP* truth values. That a similar phenomenon may arise because of deeper reasons, deriving from the use of 'true', is not to be ruled out.

[44] *Using* the concept of truth in order to give semantics in the form of truth conditions, as was proposed by Davidson, is of course quite different from giving the semantics of 'true' itself.

aspects have been clarified, is, I think, an error. We are not concerned here with marginal brain teasers, but with an essential feature of language. The failure of compositional semantics (where "composition" is intended for sentence parts) is highly significant.

Note also that 'true' is *the* paradigm of a network-creating predicate. It can send us, via 'what she said', 'whatever he asserted', 'what is written on page 1', and their like, from John to Joan, from Joan to Jack, to Jill, to McX, and so on; the tour is unrestricted and sometimes endless. 'Know' (in the sense of "know that") and 'believe' (in the sense of "believe that") have similar potential: 'I know what she knows', 'what he believes I doubt'. But 'true' is the more fundamental.[45] It is no wonder that problems arising out of this chain-forming capacity have surfaced in the theoretical research of artificial intelligence. Once it is realized that knowledge, or belief, are to be modeled as predicates rather than as sentential operants,[46] the semantics of 'true' becomes relevant in the contexts of knowledge and belief representation. This can be seen in quite a few works published mostly in the volumes of TARK. Thus, L. Morgernstern[47] and M. Kremmer[48] draw on Kripke's model, while N. Asher and H. Kamp[49] follow the constructions of Gupta and Herzberger. Among other works motivated by problems of self-reference in epistemic contexts are those of Asher,[50] R. Koons,[51] D. Perlis,[52] J. des Rivieres and H. Levesque,[53] and R. Thomason.[54]

[45] Note how often 'know', and to lesser extent 'believe', ride on the back of 'true'. We might say 'I know that what she says is true', but we cannot say 'I know that what she says', while 'I know what she says' means something different. Knowledge modalities like $K_J(x)$, which are used in formal systems, are rendered into natural language by invoking 'true' or an equivalent expression: 'J knows that x is true' or 'J knows that x is the case'.

[46] Sentential operants are too restrictive: 'John knows something that Bill does not know' is inexpressible if 'know' is treated as a sentential operant.

[47] "A First Order Theory of Planning, Knowledge, and Action," in TARK, J. Halpern, ed. (Los Angeles: Morgan Kauffman, 1986), pp. 99–115.

[48] *Logic and Truth,* Ph.D. Dissertation (University of Pittsburgh, 1986).

[49] "The Knowers Paradox and Representational Theories of Attitudes," in TARK (1986), pp. 131–48.

[50] "Reasoning about Belief and Knowledge with Self-reference and Time," in TARK (1988), pp. 61–81. Also, "Intentional Paradoxes and an Inductive Theory of Propositional Quantification," in TARK, R. Parikh, ed. (Los Angeles: Morgan Kauffmann, 1990), pp. 11–28.

[51] "Doxastic Paradoxes without Self-reference," in TARK, M. Vardi, ed. (Los Angeles: Morgan Kauffmann, 1979), pp. 29–42.

[52] "Languages with Self-reference I: Foundations," *Artificial Intelligence,* xxv (1985): 301–22.

[53] "The Consistency of Syntactical Treatment of Knowledge," in TARK (1986), pp. 115–130.

[54] "Paradoxes and Semantic Representation," in TARK (1986), pp. 225–39.

IV. APROPOS PROPOSITIONS

Some would have it that truth and falsity attach to propositions (or statements)—entities from which linguistic particulars have been abstracted away. There is, however, no agreed view on the nature of these objects and the difficulties surrounding the concept may incline us to consider, at least provisionally, less problematic creatures. In any case and however propositions are conceived, the crucial question is that of relating them to linguistic constructs—to sentences, for example. On the present view, the question takes the form: What proposition, or statement,[55] is expressed by a pointer?

I have suggested in the "Preview" that, prima facie, a pointer can be taken to express whatever is expressed by the sentence it points to—except for failures, marked by *GAP*. On this suggestion, non-failed pointers to the same sentence express the same thing. Since (by LPD) such pointers get, invariably, the same truth value, we will not run into trouble, as far as truth values are concerned. The plausibility of the suggestion rests, however, not on LPD, but rather on its proof, which shows the following. Whenever p and q are non-failed pointers to the same sentence, their evaluation requires the same sequence of necessary steps, ending at a stage where each can be assigned the same truth value by applying the same rule.[56] This indicates that an account that correlates propositions with sentences could be extended to pointers. The heuristics would be as follows. Two non-*GAP* pointers express the same proposition, inasmuch as this can be said of the sentences they point to.

[55] There is considerable overlap in the use of the two terms and I am not altogether certain as to the "right" one. There is a usage, which I do not follow, whereby propositions are taken as the meanings of declarative sentence types, including indexical ones. By contrast, statements are attached to particular utterances. In the tradition deriving from Frege and Russell, however, the proposition expressed, say, by 'it is now raining' changes according to time. Read thus, propositions can be associated with sentence types only after converting the sentences into "eternal" versions. What is then the difference between this notion of proposition and the notion of statement as advocated by J. L. Austin and Peter Strawson? It appears that propositions are prelinguistic entities, existing in their own right, whereas a statement is something one makes by producing a linguistic token. Put bluntly, if there were no people, there would have been no statements but there would have been propositions. Stripping them of their more metaphysical aspects, one can say that propositions are possible statements. And this is how I propose to view them here. I have opted for 'proposition', because it is usually this term which is employed when a formal treatment is intended. Thus, Barwise and Etchemendy (*op. cit.*) propose a formal model of so-called Austinian propositions for what Austin, who rejected 'proposition', called statements. Indeed, if propositions are possible statements, then the general theory of statements is the theory of propositions.

[56] Were we to use another version of the evaluation procedure, a similar analysis of the procedure would be required in order to justify stipulations about "expressing the same thing."

The propositions to be gotten along the suggested lines are "ontological" rather than "epistemic." A person stating what to him is a perfectly clear thought might fail to express a proposition, because unwittingly he has involved himself in some loop. This is similar to the view obtained from some current accounts of propositions. On these accounts, certain terms (the so-called directly referring ones) introduce their very references as components of the proposition.[57] Mistaken beliefs about reference may lead to a person's expressing a proposition he had not intended. And when a directly referring term fails to refer, no proposition is expressed.

But nothing prevents us, at least not in principle, from considering other kinds of propositions aimed at reflecting finer shades of meaning. Just as, for various purposes, it is natural to hide the reference-picking machinery of certain terms, it might be desirable, for other purposes, to expose it in the proposition. What propositions are depends on what dimensions of meaning they are supposed to reflect. The question what are the right propositions is as meaningful as the question what sort of items a map should display, or what is the right granularity of a film.

A move to more epistemic propositions may entail distinctions between various gap pointers. The gap constituted by the line-1 sentence derives from a contingent fact: the place where that sentence is written. Someone might have put it there because of a mistaken belief about line numbers.[58] By contrast, the loop produced by 'What I am saying at this very moment is not true' cannot be amended by moving it, and other utterances, around. Assuming that the speaker knows the language, it is difficult to see what statement of fact he might have intended. The present formalism does not distinguish between the two; both are construed as a pointer, p, which points to $\neg Tr(p)$. The distinction can be introduced at the cost of additional structure: for example, a set of possible pointing functions in addition to the actual one.[59]

Finally, it can be argued that *some* meaning is expressed even in the case of noncontingent gaps. For a pointer provides a prescrip-

[57] Cf. Kaplan, "Demonstratives: An Essay on the Semantics, Logic, Metaphysics, and Epistemology of Demonstratives and Other Indexicals," in *Themes From Kaplan*, J. Almog, J. Perry, and H. Wettstein, eds. (New York: Oxford, 1989).

[58] In an example of Prior, used by Burge, a student who thinks he is in room 9 writes on the blackboard: 'Whatever is written on the blackboard in room 10 is false', expressing thereby his true belief. It turns out that he himself is in room 10. (N. Prior, "On a Family of Paradoxes," *Notre Dame Journal of Formal Logic*, II, 1 (1961): 16–32.)

[59] Distinctions of similar nature, between kinds of pointing, are used in certain more sophisticated variants of the evaluation procedure, aimed at reducing black holes.

tion for computing truth values. It amounts to a kind of algorithm. And as an algorithm it makes sense, even when it leads inevitably to a nonterminating computation.

While not endorsing, at least initially, any particular position concerning propositions, the present proposal incorporates a certain methodology: consider propositions after having clarified sufficiently the rules and mechanisms of truth-value assignments. Propositions—if and when we introduce them—can be, then, tailored to fit the patterns we deem important. To do otherwise is to commit ourselves to preconceived notions, which can be interesting and fruitful in their domain of origin, but ill-matched to the issues at hand.

An opposite approach underlies the work of Barwise and Etchemendy (cf. fn. 15), some aspects of which I shall now discuss in brief. Barwise and Etchemendy propose formal models of propositions, constructed according to situation semantics. They classify their propositions into Russellian and so-called Austinian ones, with clear preference for the latter. One of their goals is to include in the model *cyclic* propositions: those involving self-reference. And their ingenious idea is to employ for that purpose AFA (set theory with Aczell's *antifoundation axiom*)—a theory of non-well-founded sets which accommodates, naturally, a variety of "cyclic creatures," for example, a proposition P such that P = *not-P*.

While the BE (Barwise-Etchemendy) framework assigns truth values to propositions, it does not provide guidelines, let alone a procedure, for assigning truth values to sentences—either types or tokens. For when it comes to determining the proposition that is expressed by a given utterance, a kind of free parameter enters.[60] In cases of self-reference there is considerable latitude; much is up to the interpreter, who can toggle truth values by suitable adjustments in what he can choose. Therefore, such utterances are construed as ambiguous. Presumably, Barwise and Etchemendy regard the problem of disambiguation (which must be addressed if we are to determine truth values) as belonging to some branch of pragmatics.

It can be shown that the assignment of truth values according to the principles of pointer semantics is outside the scope of the BE framework. For the needed distinctions cannot be made in AFA.

[60] Namely, the situation—in the technical sense of situation semantics—associated with the utterance. A situation can be very roughly described as a collection of facts. But the complete circumstances of the utterance—including, if you wish, the whole world history—fall short of constituting a complete situation. For the truth or falsity of some circular proposition is itself a "fact" that the interpreter may, or may not, include in the situation.

For example, AFA cannot provide for distinguishing between the sentence tokens on lines 1 and 2 in the two-line puzzle.[61]

Can we correlate sentence tokens with propositions of the BE kind, whose truth values are the values derived by pointer semantics? To a certain extent this is possible. It may work in general, but it requires the removal of certain restrictions underlying the BE modeling of propositions.[62] Perhaps these are artifacts of the modeling.

The major nontechnical point of conflict between the BE framework and pointer semantics is the former's failure to make place for the notion of a *failed utterance.* It provides the liar sentence and its kin with circular propositions, which I do not find objectionable—as long as we mark certain propositions by *GAP.* But, in the BE framework, propositions are either true or false. When P = *not-P*, the proposition and its negation are accorded F; which fact is explained in BE by the special modeling of negation—clearly not the classical concept. Under a different negation concept, proposed in BE and called *denial,* both a proposition and its denial can be true. Whatever the insights revealed by these nonstandard negation concepts, their service in disguising gaps is negative. A picture that fails to distinguish certain types of failure can be quite misleading. It is also at odds with Austin, who devoted much effort to the taxonomy of failures and who distinguished failures caused by illegitimate self-reference from mere self-contradiction.[63]

V. THREE PREVIOUS ACCOUNTS

I shall discuss three accounts related to my proposal, those of Parsons (which can be seen as a development of a theme in Russell),

[61] The two nodes in figure 2 of the "Preview" represent in AFA the same set, namely, the unique set containing 1 as its sole member. This is forced by the extensionality axiom.

[62] The definition of disjunctive propositions is such that all the disjuncts are about the same situation—which is also the situation the disjunction is about. A similar restriction applies to conjunctions. This has some problematic consequences. Consider a disjunction $S_1 \vee S_2$, where S_1 is a liar sentence and S_2 is a sentence saying that S_1 is not true. If both disjuncts must be about the same situation, the second disjunct, and along with it the whole disjunction, come out nontrue. In pointer semantics this disjunction (i.e., a token, or some pointer to it) gets the value T, because the second disjunct does.

[63] "Those of the so-called 'logical paradoxes' . . . which concern 'true' and 'false' are *not* to be reduced to cases of self-contradiction, any more than '*S* but I do not believe it' is. A statement to the effect that it is itself true is every bit as absurd as the one to the effect that it is itself false. There are other types of sentence that offend against the fundamental conditions of all communication in ways distinct from the way in which 'This is red and this is not red' offends . . . ." Austin, "Truth," in *Philosophical Papers* (New York: Oxford, 1961). Fn. in the 1970 paperback ed., p. 129.

Burge, and Skyrms.[64] These accounts do not assign truth values to sentence tokens, but they share with the present work the view that places the strong liar and its kin at the heart of the semantic paradoxes. Needless to say, I have no intention to review the vast literature devoted to the liar. The choice of these works is dictated by their relevance to my proposal and reflects no attempt at ranking.

Ambiguity and interpretation schemes. Parsons proposes to view the liar and its kin as cases in which different interpretation schemes, based on different truth concepts, are applied to the same sentence (type). No formal explication is given to "interpretation scheme," but a direction is suggested by rewriting '*A* is true' as 'there exists a proposition *P* such that *A* expresses *P* and *P* is true'. The interpretation scheme is then determined by fixing the domain of propositions over which the quantified '*P*' ranges. The idea goes back to Russell, who suggested that the paradox arises from an attempt to quantify over "too many propositions" and is to be resolved by restricting the quantifier ranges. Such a solution underlies Russell's theory of types. Since there are many natural domains that can serve as the range for the propositional variable, the sentence turns out to be ambiguous. But whatever the domain, it cannot include the proposition expressed by the liar sentence itself.

Intuitively, the interpretation scheme reflects the comprehensiveness of the point of view from which truth-value judgments are made. In the case of our two-line puzzle, the sentence on line 1 fails to express a proposition, under the first "straightforward" interpretation; but under a second scheme, which takes into account the previous failure, the sentence is seen to be true. Parsons does not give an account of how the interpretation is determined, except for noting that it is determined by the circumstances of the sentence's utterance. He proposes to regard this ambiguity in the same light as the ambiguity surrounding the interpretation of quantifiers or the use of 'say', 'mean', and their like.

In a 1982 postscript to his paper, Parsons suggests that Burge's indexical conception of 'true' can serve to explicate the ambiguity.

[64] Parsons, "The Liar Paradox," *Journal of Philosophical Logic,* III (1974): 381–412.

Russell, "Mathematical Logic as Based on the Theory of Types," in *Logic and Knowledge: Essays 1901–1950,* R. Marsh, ed. (New York: Macmillan, 1956), pp. 59–102.

Burge, *op. cit.*

Skyrms, "Return of the Liar; Three-Valued Logic and the Nature of Truth," *American Philosophical Quarterly,* VII (1970): 153–61; and his "Intensional Aspects of Semantical Self-reference," in *Recent Essays on Truth and the Liar Paradox.*

But, in the light of the analysis of section III (and of the next subsection devoted to Burge's work), this would not do justice to the intended picture. For the ambiguity of the liar is not to be traced to indexical-like shifts in what the 'true' that occurs in the sentence denotes. If there is ambiguity, it can derive only from different points of view that we—the interpreters—adopt from the outside.

According to pointer semantics, there is no ambiguity. The token on line 1 is definitely not true (and not false either); the token on line 2 definitely is. But if one were to smudge over the distinction between tokens, one could say that the sentence *type* is ambiguous: different instantiations of it get different truth values. One can therefore regard different pointers to the same sentence as expressing its meaning under different interpretation schemes.

There are other systems that can provide formal backups for the ambiguity picture. The constructions of Gupta and Herzberger, whereby the liar changes its truth value according to the evaluation stage, give rise to such a picture. And the more recent work of Barwise and Etchemendy, which I discussed above, provides another modeling of ambiguity. It is indeed directly related to Russell's ideas, since one of its aims is to model Russellian propositions.

While these models do not solve the two-line puzzle, they solve or illuminate what we might call the one-line puzzle, which is as follows. A philosopher contemplates the sentence on line 1, which asserts its own nontruth. First he infers that the sentence is not true. Realizing on second reflection that the sentence itself "says" just this, he infers that it is true after all. He may continue to the third round, whereby the sentence becomes nontrue, after which it becomes true again, and so on.

Burge's proposal. I have already discussed in section II the indexical construing of 'true'. Here I shall confine myself to more specific details of Burge's work. His proposed setup comes in two parts. The first consists in a formal language containing a family of truth predicates, $true_i$, $i = 0, 1, \ldots$, arranged in a Tarski-like hierarchy, with subscripts denoting the levels. Each predicate is to serve as a truth predicate for the portion of the language "below it," that is, for all sentence types containing truth predicates with a strictly smaller index. Yet each predicate takes as arguments all sentence names of the language. Because of self-reference there are, for each i, sentences that inevitably fail to get a truth_i value: neither the sentence nor its negation is true_i. The idea is to regulate the assignment of truth_i values so as to approximate, as far as possible, the properties required of a truth predicate, while maintaining the predicate's position in the hierarchy. Burge stipulates various plausible axioms; for

example, axioms to the effect that if A is true$_i$, then A, as well as $true_i(A)$ are true$_k$, for all $k > i$.[65] Apparently, the proposed axioms are not sufficient.[66] But this is a technicality that can be amended. There is, in fact, a neat generalization of Kripke's construction to languages containing a hierarchy of truth predicates, which achieves the desired result in a simple straightforward way. It is mentioned briefly in my 1983 paper.[67] We can therefore assume that truth$_i$ values have been assigned in a satisfactory way to all sentences and all i. Call this language the stratified language. (Note that a liar$_i$, i.e., a sentence saying of itself that it is not true$_i$, is not true$_i$ and neither is its negation. But a liar$_i$ is true, when evaluated from the outside, via the usual truth definition. Since each $true_k$ behaves like an ordinary truth predicate for the portion of the language below it, a liar$_i$ is true$_k$ for all $k > i$.)

The second part of Burge's proposal is to rewrite the ordinary sentences within a given discourse as sentences of the stratified language; this is done by "indexing," i.e., by attaching subscripts to the occurrences of 'true'. No precise rules are given; Burge sees it as a pragmatic question and provides certain informal guidelines.[68] The recourse to informal pragmatics has brought the proposal under severe criticism by Gupta.

The main trouble with Burge's proposal, however, is not that the pragmatic principles for indexing are imprecise, but that the indices

[65] Actually, Burge considers various sets of axioms, but this need not concern us here.

[66] The following is, I think, not derivable: If $\neg true_i(A)$ is true, then A is not true$_i$.

[67] I found it when, independently of Burge's work, I was considering formal languages that incorporate Tarskian hierarchies. Here it is. Start by evaluating all sentences by the *standard rules*, treating each of the predicates $true_i$ as a truth predicate. The standard rules consist of the usual (Tarskian) truth conditions (applied, as in Kripke's construction, via Kleene's strong three-valued logic) as well as the following rule. If A is true$_i$, then $true_i(A)$ is true$_j$ for all j (including the case $j < i$!). Having done this, extend the resulting valuation as follows. For each A which is not true$_0$, declare the sentence $\neg true_0(A)$ to be true$_i$, for all $i > 0$. Now repeat the same process of applying the standard rules, but only to the truth predicates on level > 0. Thus, by the end of the first stage the extension of $true_0$ is determined. At the end of the second stage, for each A which is not true$_1$, declare $\neg true_1(A)$ to be true$_i$ for all $i > 1$. Again, repeat the process for all predicates at level > 1. And so on, until we run out of truth predicates.

[68] A principle called "verity" instructs us to assign indices in a way that maximizes the applicability of the truth-defining schema, that is, in a way that minimizes gaps. A less important principle called "justice" says that the assignment should not prefer one statement to another, unless for some good reason. Thus, if each of two persons asserts of the other that he is not saying the truth, both should be given—for reasons of symmetry—the same subscript, resulting in both being gaps. Either principle is subject to a *ceteris paribus* clause.

do not do the desired work. An indexing can only lead to $truth_i$ values; it does not determine at which level a given sentence is to be evaluated: Which $true_i$ are *we* employing when we say of a sentence *A*—written already in indexed form—that it is true, or not true, or false? Burge invokes here what he calls a pragmatic implicature: consider the evaluating sentence—the sentence asserting the truth, or nontruth, of *A*; index 'true' in *that* sentence. That index is the level at which *A* is evaluated. This, however, is only the beginning of the story; for occasionally the implicature is canceled in favor of a different one. For example, the sentence on line 1 (in the two-line puzzle[69]), which asserts its own nontruth, is the evaluating sentence for itself. Suppose the 'true' there is indexed by *i*. By the implicature, when asserting that the sentence is not true, we are—in fact—asserting that it is not $true_i$; which, indeed, it is not. But the same implicature rules that when we evaluate our assertion, we do so again on level *i*. Therefore, our assertion, expressed by the sentence on line 2, is nontrue as well. This unwelcome result is avoided by having the implicature cancelled and by raising the level of the second evaluation to some *k*, greater than *i*. As Burge sees it, the move from line 1 to line 2 signifies a switch from one implicature to another. The change of truth value is only apparent: both are $true_k$, and both are not $true_i$.

What Burge fails to observe is that the resort to implicatures voids the indexing of its significance. Not only the liar, but all problematic cases of self-reference, require implicature switches. Thus, in the Plato-Aristotle example quoted from Buridan (cf. section III), the 'true' in both sayings gets the same index, resulting in both being nontrue. If a third party were to assert that Plato's, or Aristotle's, saying is not true, *his* saying would be true. But this can be sustained only if the implicature employed in judging the first two assertions is changed when judging the third. Or take Nixon-Dean example, where each says of the other that all his utterances concerning Watergate are not true. If all other Watergate assertions made by each are not true, a loop is created and *we* can say that these two assertions are not true. Again, the implicature must be changed when judging *our* assertion; because when Nixon's and Dean's sayings are spelled out as conjunctions, our assertion is seen to repeat a conjunct from each.

In fact, whenever the jump rule is used in the evaluation procedure, an implicature must be canceled in favor of a different one

[69] The example used by Burge is somewhat different, but the difference does not matter here.

that shifts the level of 'true' in the evaluating sentence. Since the shifts are cumulative, a count must be kept to determine the "right implicature" for choosing the level; and the level can be any finite or infinite ordinal. Burge ignores all this. Having introduced implicature in the analysis of the liar, he goes on in the second half of the paper to discuss the stratified language and the indexing, without mentioning that matter again. The process of implicature shifts is altogether obscure; no guideline, nor any indication, is given. But there lies the real problem. For knowing the mechanism that governs implicature choice is almost like knowing the evaluation procedure. It is this mechanism which does the real work. Indexicality rests idle.

The intensional conception. The analogy between an intensional case and a strong liar, on which Skyrms draws, is as follows.[70] That the sum of 7 and 2 is odd is necessary, that the number of planets is odd is not—even though the sum of 7 and 2 *is* the number of planets; just so, the sentence on line 2 is true, the sentence on line 1 is not—even though the sentence on line 2 *is* the sentence on line 1. Skyrms proposes to solve the strong liar by construing 'true' as intensional. Truth values are to be assigned to sentence types *under descriptions.* Under the description 'the sentence on line 2', the sentence is true; under the description 'the sentence on line 1', it is not.

Formally, Skyrms proposes to assign truth values to sentence-description pairs (s,d), where d is a description of the sentence type s, and to do so in a way analogous to Kripke's fixed-point construction. He shows that, given a partial valuation of sentence-description pairs (which may distinguish between pairs that differ only in the description coordinate), a variant of Kripke's construction can be used to extend it. Skyrms's construction falls short of the desired goal since, being based on (the analogs of) Kripke's rules, it does not have the gap and jump rules, which are crucial. It works as long as we are moving through standard values; e.g., if we have already assigned (s, d) the value T, we can go on and assign (s', d') the value F, where s' is the negation of s and d' is some appropriate description obtained from d. But it does not provide a way for assigning T to the token on line 2 (i.e., to the pair (s, d), where s is the sentence and d is the description 'the sentence on line 2'). In general, it falls short whenever the jump rule is invoked.

[70] I use our two-line puzzle; the difference between it and the example used in Skyrms does not matter here.

Now, if desired, one can regard the pair (s,d) as a pointer to s. And, vice versa, a pointer to s can be changed to a pair (s, d) where d is some description derived from the pointer. *Formally,* my algorithm can be viewed as providing a truth-value assignment to sentences under descriptions, achieving thereby Skyrms's goal.[71] I do not see, however, how the failure of substitutivity arising in the strong-liar puzzle can be accounted by intensionality, without stretching "intensionality" so as to render it uninteresting. For, by the same token, one could subsume under it all other context-dependent phenomena, including, for example, indexicality. When each of Jack and Jill says, "I am tall," but only Jack is tall, we could explain the truth-value difference by the intensionality of 'true', namely, 'I am tall' is true, under the description 'the sentence uttered by Jack', but not true under the description 'the sentence uttered by Jill'. I doubt that we want to adopt this picture.

Intensionality is affected by the way in which referring terms pick their references. The sum of 7 and 2 must be the number 9, but it is only a cosmic accident, or so we think, that 9 is also the number of planets. But nothing in the liar situation hinges on some unclarity in the referential picture. The intensional effect disappears if the referential connection is made explicit. Given that the number of planets is the sum of 7 and 2, it *is* necessary that the number of planets is odd. Or, given that George IV knew that the author of *Waverly* was Scott, it is inconceivable that he wondered whether Scott wrote *Waverly*. But can we say that, given that the sentence type on line 2 is the sentence type on line 1, the sentence on line 1 is true? Certainly not.

Viewed in retrospect, both conceptions of truth—as an indexical, and as an intensional notion—are attempts to account for phenomena arising out of liar-like paradoxes by using tools that fall essentially within the framework of traditional logic. Both come short of providing a solution, but for different reasons.

HAIM GAIFMAN

Columbia University

[71] Indeed, judging from discussions I had with him, Skyrms regards the algorithm as a way of constructing the desired intensional fixed point.