Compositional Interpretation in Which the Meanings of Complex Expressions are not Computable from the Meanings of their Parts

Peter N. Lasersohn

October 26, 2007

**CENTER FOR THE STUDY
OF LANGUAGE
AND INFORMATION**

1 Introduction

The title of this paper should sound paradoxical. Isn't it a matter of *definition* that in a compositional interpretation, the meanings of complex expressions are computed from the meanings of their parts?

Certainly this would seem to be what many authors mean by the term *compositional*. In introductory textbooks, the idea of compositionality is often introduced by pointing out that if there weren't some systematic way of calculating the meanings of complex expressions from the meanings of their parts, we could not explain how speakers understand novel sentences, or indefinitely many sentences. The following passage from Larson and Segal (1995): 11–12 is representative; similar passages may be found in many other textbooks:

> The hypothesis that we know a set of compositional semantic rules and principles is a highly attractive one having a great deal of explanatory power. In particular, it accounts for three notable and closely related features of linguistic competence. First, it explains why *our understanding of sentences is systematic* — why there are definite, predictable patterns among the sentences we understand... Second, the hypothesis accounts for the obvious but important fact that *we can understand new sentences*, sentences we have never come across before... Third, the hypothesis accounts for the slightly less obvious but equally important fact that *we have the capacity to understand each of an indefinitely large number of sentences*...

Similar points are often made in more advanced or technical discussions of compositionality as well. Dowty (2007): 23–24 asks us to consider several points:

> • Linguists agree that the set of English sentences is at least recursive in size, that English speakers produce sentences virtually every day that they have never spoken before, and that they successfully parse sentences they have never heard before.
> • If we accept the idealization that they do understand the meanings of the sentences they hear, obtaining the same meanings from them that others do, then:
> • Since the meanings of all sentences obviously cannot be memorized individually, there must be some finitely characterizable procedure for determining these meanings, one shared by all English speakers.
> • As the sentences themselves can only be enumerated via derivations in the grammar of the language, then inevitably, the procedure for interpreting them must be determined, in some way or other, by their syntactic structures as generated by this grammar (plus of course the meanings of the individual words in them).

Dowty goes on to suggest that in light of these points, the primary

empirical issue regarding compositionality is not *whether* natural language semantics is compositional, but rather *how* it is compositional.

Given the frequency with which semanticists justify the principle of compositionality by appealing to the ability of language users to understand novel sentences (or indefinitely many sentences), it is striking to realize that in more mathematically-oriented work on compositionality, often very little is done to preserve the idea that the meaning of a complex expression may be computed from the meanings of its parts.

For example, in the system of Montague (1970), which I think represents the most standard approach to the mathematical representation of compositional interpretation, the central idea is that meaning is assigned homomorphically, not that it be assigned computably. The homomorphism requirement does nothing to guarantee computability, as we will see from a simple example.

In this paper, I will explore several issues regarding the relation between the principle that interpretation is assigned homomorphically and the idea that meanings can be systematically computed. Since language users presumably must have some way of computing the meanings of the expressions they use and understand, it might seem natural simply to impose a computability requirement on a system of homomorphic interpretation like Montague's. However, I will argue that there are good reasons not to do so. Rather, I claim, psychological computation of meanings may be done on an entirely separate basis from homomorphic interpretation; this will be illustrated using a second example, in which sentences in a small, artificial language may be easily understood, despite being assigned a non-computable homomorphic interpretation.

If psychological understanding of meanings is not done by computing a homomorphic interpretation function, then the claim that interpretation is assigned homomorphically can only be motivated on grounds other than explaining speakers' ability to understand novel sentences. I will argue that such motivation may be found by noting that the principle of homomorphic interpretation is equivalent to familiar principles licensing the substitution of synonyms or coreferring terms for one another. To the extent that we observe languages to respect such substitution patterns, then, a principle of homomorphic interpretation is well supported, regardless of its relation to the psychological computation of meaning.

2 Background Notions

As background to our first example, and to some of the points to be made later in the paper, we begin by reviewing the major features

of Montague's system, as well as some basic definitions. Central to Montague's approach is the idea that complex expressions are derived from more basic expressions through successive application of syntactic operations. If we close the set of basic expressions under all the syntactic operations, the result is the *syntactic algebra* for the language, which we may notate $\mathfrak{A} = \langle A, F_1, \ldots, F_i \rangle$.[1]

We assume that each expression is assigned a "meaning" from some set B. Although Montague made specific proposals about what "meanings" are, those proposals are largely independent of this part of the theory; for the purposes of defining compositionality, the elements of B may be anything whatsoever.

Although there are no requirements on what meanings *are*, we do require that they relate to each other in specific ways. In particular B must be closed under a series of operations $G_1, \ldots, G_i$, which have the same numbers of argument places as $F_1, \ldots, F_i$, respectively. Thus $\mathfrak{B} = \langle B, G_1, \ldots, G_i \rangle$ must be a "similar" algebra to $\mathfrak{A}$. We can think of $G_1, \ldots, G_i$ as operations for deriving meanings of complex expressions from the meanings of their parts. Each semantic operation $G_n (1 \leq n \leq i)$ corresponds to a syntactic operation F_n and is used to assign meanings to the expressions which F_n derives.

Now we require that the meanings of expressions in A be assigned by a homomorphism h from $\mathfrak{A}$ to $\mathfrak{B}$. That is, for each of our syntactic operations F_n and its corresponding semantic operation G_n, we require:

(1) $h(F_n(\alpha_1, \ldots, \alpha_m)) = G_n(h(\alpha_1), \ldots, h(\alpha_m))$

It is perhaps worth noting that Montague himself did not use the term *compositionality* for this requirement; but the idea that this homomorphism requirement is a formalization of the intuitive idea of compositionality seems quite widespread (see, e.g., Partee et al. (1990): 334; Janssen (1997): 450; Westerståhl (1998): 635–636). The intuition behind it is pretty straightforward: If you derive an expression syntactically from more basic expressions $\alpha_1, \ldots, \alpha_m$, its meaning should be derived from the meanings of $\alpha_1, \ldots, \alpha_m$.

This does not guarantee that the meanings of complex expressions can be computed from the meanings of their parts, for the very simple reason that there is nothing in this picture to require that the semantic operations $G_1, \ldots, G_i$ or the homomorphism h be computable functions.

[1] Here $F_1, \ldots, F_i$ are the syntactic operations, and A is the set obtained by closing the set of basic expressions under these operations. Montague's notation is a little more general, and avoids the inexplicitness of the three dots, but I think the current notation is easier to read and should be clear enough for our present purposes.

A *computable function* is one which can be represented as a recursively enumerable set of ordered pairs. Mathematically, there is no reason to suspect that the homomorphism requirement would force the semantic operations to be computable, and in fact it is easy to show that it does not.

3 A Simple Example

As an example of homomorphic interpretation in which the semantic operations are not computable, consider the following simple artificial language L_1:

(2) *Formation Rules:*

a. 0 is a term
b. 1 is a term
c. If α is a term and β is a term, then $[\alpha\beta]$ is a term
d. If α is a term, then $\#\alpha$ is a sentence

In this language, sentences consist of any number of 0's and 1's (bracketed to show their derivational history) preceded by #.

Given these rules, we may define our syntactic operations as F_1 and F_2, where:

(3) *Syntactic Operations:*

a. $F_1(\alpha, \beta) = [\alpha\beta]$
b. $F_2(\alpha) = \#\alpha$

So, our set A will be the closure of $\{0, 1\}$ under F_1 and F_2; our syntactic algebra is $\mathfrak{A} = \langle A, F_1, F_2 \rangle$.

To interpret this language, we use as "meanings" the set of natural numbers $\mathbb{N}$, the strings of a second language L_{NR}, which is not recursively enumerable, and a "dummy" object † to serve as the meaning of ill-formed expressions.[2] That is, $B = \mathbb{N} \cup L_{NR} \cup \{\dagger\}$.

Although L_{NR} is not recursively enumerable, we do make the standard assumption that it is a set of finite strings over a finite vocabulary, hence that it is denumerably infinite. Therefore, there is a function putting it in 1-1 correspondence with $\mathbb{N}$. Call this function f.

Even though f might do something as simple as arrange the sentences of L_{NR} in alphabetical order, or sort them by length, it is not a computable function. (If f were computable, it would be representable

[2]Because our operations are total and A is closed under them, A will contain strings such as #0#0 which are not well-formed according to the rules in (2). For ease of presentation, I follow Montague (1970) in allowing such expressions into A and using a set of syntactic rules to identify which elements of A are well-formed, rather than using a partial algebra in which the operations are undefined for arguments which would yield ill-formed values.

as a recursively enumerable set of ordered pairs. Then you could easily recursively enumerate L_{NR} just by stripping off the first element of each pair. But we have stipulated that L_{NR} is not recursively enumerable. Therefore f is not computable.)[3]

Now, we interpret L_1 by letting each term have a natural number as its meaning, and letting each sentence have a string of L_{NR} as its meaning. If a term is of the form $[\alpha\beta]$, its meaning will be the sum of the meaning of α and the meaning of β. Each sentence is of the form $\#\alpha$, and will have as its meaning the string of L_{NR} which f assigns to the number which serves as the meaning of α. Since the operation deriving the meanings of sentences from the meanings of terms appeals to this non-computable function, there will be no general procedure for computing the meanings of sentences from the meanings of their parts.

More formally, let the semantic operations be G_1 and G_2, where:

(4) *Semantic Operations:*

a. If $x, y \in \mathbb{N}$, then $G_1(x, y) = x + y$.
(If $x \notin \mathbb{N}$ or $y \notin \mathbb{N}$, then $G_1(x, y) = \dagger$.)

b. If $x \in \mathbb{N}$, then $G_2(x) = f(x)$.
(If $x \notin \mathbb{N}$, then $G_2(x) = \dagger$.)

Note that $B = \mathbb{N} \cup L_{NR} \cup \{\dagger\}$ is closed under G_1, G_2. We let our semantic algebra be $\mathfrak{B} = \langle B, G_1, G_2\rangle$.

Now let the meaning assignment be $h : A \to B$, where:

(5) *Meaning Assignment:*

a. $h(0) = 0$

b. $h(1) = 1$

c. If $h(\alpha), h(\beta) \in \mathbb{N}$, then $h([\alpha\beta]) = h(\alpha) + h(\beta)$
(If $h(\alpha) \notin \mathbb{N}$ or $h(\beta) \notin \mathbb{N}$, then $h([\alpha\beta]) = \dagger$.)

d. If $h(\alpha) \in \mathbb{N}$, then $h(\#\alpha) = f(h(\alpha))$.
(If $h(\alpha) \notin \mathbb{N}$, then $h(\alpha) = \dagger$.)

Now it is easy to see that $h(F_1(\alpha, \beta)) = G_1(h(\alpha), h(\beta))$, and that $h(F_2(\alpha)) = G_2(h(\alpha))$; so h is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$.

Note also that for every $n \in \mathbb{N}$, there is some term α such that $h(\alpha) = n$. Hence the function mapping $h(\alpha)$ onto $h(\#\alpha)$ for each term α is f, and not some proper subset of f.[4] But f is non-computable.

[3]Some people find it paradoxical that something as systematic as alphabetization or length-sorting could be non-computable. Of course, *if we had a list* of all the strings in L_{NR}, we could construct an algorithm for alphabetizing them or sorting them by length; the problem is that there is no way to construct such a list.

[4]Of course our semantic operation is G_2, which is an extension of f, not f itself. (G_2 differs from f in that it is defined for objects which are not the meanings of terms; it maps these onto $\dagger$.) But G_2 must also be non-computable; otherwise we

Since sentences in this language are of the form $\#\alpha$, there is no general procedure in this language for calculating the meanings of sentences from the meanings of their parts.

From a mathematical point of view this is not at all surprising. But it does show that if the idea we are trying to capture in formulating a principle of compositionality is that the meanings of complex expressions can be computed from the meanings of their parts, we don't get that result just by imposing a Montagovian requirement of homomorphic meaning assignment.

Now a natural reaction at this point might be to see this as a shortcoming — but a fairly trivial and easily fixed shortcoming — of the Montagovian conception of compositionality. After all, for any natural language, or even any useful language, we must be able to compute the meanings. So it might seem that the obvious thing to do would be to impose an additional constraint on our system, requiring that the meaning assignment h and/or the semantic operations $G_1, \ldots, G_i$ be computable functions. And in fact, those few discussions of this issue that I've been able to find in the literature seem to favor exactly this (Muskens (1995): 89; Kracht (2003): 186).

What I'd like to suggest is that we *don't* need to impose such a requirement in order to explain the fact that people can comprehend novel utterances, and that in fact there might be good reasons for *not* requiring semantic operations to be computable. In fact I think we can have a perfectly reasonable, comprehensible language with interpretations assigned in a semantic algebra with non-computable operations.

4 Non-Computable Reference Assignment

The idea of non-computable semantic operations might be easiest to accept if we consider it first in the context of compositionality of *reference*, rather than compositionality of *meaning* or *sense*. In a Fregean semantics, it is the sense of an expression, not the referent, which serves as the expression's "cognitive value"; it is the sense, not the referent, which is mentally "grasped" when one understands the expression. So if our main reason for believing that semantic operations must be computable is that this assumption is necessary for a psychological explanation of how people understand novel utterances, it is perhaps not quite so obvious that we must assume computability of reference as it is that we must assume computability of sense.

Here it is worth recognizing that Montague's system draws a three-

---

could compute f by computing G_2, then stripping off all pairs with † as their second element.

way distinction between "meaning," "sense" and "denotation" (or reference),[5] and it is only at the level of meaning that the homomorphism requirement is imposed. In other words, there is no compositionality of reference (or even sense) in Montague's system.

But compositionality of reference is quite crucial to some semantic theories, including — I would claim — that of Frege (1892/1970). Of course there has been some dispute whether Frege believed in compositionality,[6] and it may be that his views on this changed over his lifetime; but I think it is clear that some of his most famous arguments only make sense against a background assumption of compositionality, and specifically of compositionality of reference.

For example, Frege argued that sentences denote truth values, by noting that the truth value is what is guaranteed not to change if a constituent of the sentence is replaced by a different expression with the same reference. This argument is compelling only if we assume that the referent of the sentence is determined by the referents of its constituent parts. Without this assumption, we would have no reason to expect that the referent of the sentence *should* remain constant across the substitution.

Likewise, Frege argued that all expressions shift their reference in indirect quotation and similar contexts, where they refer instead to what would customarily be their sense. The reason is that in such contexts, substitution of a constituent by an expression with the same customary reference will not preserve truth value — that is, will not preserve the referent of the sentence as a whole. Again, this argument is only compelling if one assumes that the referent of the whole sentence is determined by the referents of the parts.

So I'd like to assume for the sake of discussion that reference is determined compositionally. Now if we take this to mean it is assigned homomorphically, we can ask whether it is reasonable to require the operations in the semantic algebra — that is, the reference algebra — to be computable. I think the answer is clearly *no*, because we can

[5]The "meaning" of an expression is analogous to its *character* in the sense of Kaplan (1989) — it determines its sense relative to a range of contexts. "Sense" and "denotation" are the familiar notions of intension and extension.

[6]The main evidence against the idea that Frege believed in compositionality appears to be his dictum "it is only in the context of a proposition that words have any meaning" (Frege (1884/1980) §62; see also the Introduction, §60, §106). As I read him, however, Frege is not arguing for a theoretical principle that words are meaningless in isolation, but for a methodological strategy of identifying the meaning of a word by considering what it contributes to sentences in which it appears (as opposed to simply contemplating the word in isolation). Frege scholars may disagree; see Pelletier (2001), Janssen (2001) for more discussion.

refer to specific strings in a non-recursively-enumerable set (and apply predicates to them, make true or false claims about them, etc.) using mechanisms like those in our previous example.

To see this, consider a language L_2 like L_1, but with a few changes: All terms of L_1 will be "number terms" of L_2. All sentences of L_1 will be "string terms" of L_2. We will continue to map the number terms onto natural numbers, and the string terms onto strings in a non-recursively-enumerable language L_{NR}; but we regard this mapping as an assignment of referents, not of meanings. L_2 will also have predicates, including a designated predicate **TL**. Formulas will consist of a predicate followed by a string term in parentheses. More formally:

(6) a. 0 is a number term.
b. 1 is a number term.
c. If α is a number term and β is a number term, then $[\alpha\beta]$ is a number term.
d. If α is a number term, then $\#\alpha$ is a string term.
e. $\mathbf{TL}, P_1, P_2, P_3, \ldots$ are predicates.
f. If α is a predicate and β is a string term, then $\alpha(\beta)$ is a formula.

Our syntactic operations include F_1 and F_2 as in our previous example,[7] as well as F_3, where $F_3(\alpha, \beta) = \alpha(\beta)$. A will now be the closure of $\{0, 1, \mathbf{TL}, P_1, P_2, P_3, \ldots\}$ under F_1, F_2, F_3; the syntactic algebra $\mathfrak{A}$ will be $\langle A, F_1, F_2, F_3\rangle$.

As in L_1, we will need to make use of a 1-1 correspondence f from $\mathbb{N}$ to L_{NR}. In our previous example, it did not matter which function this was, as long it was a 1-1 correspondence. For the interpretation of L_2, however, let us assume more specifically that f is a function which arranges the strings of L_{NR} in order of increasing length, mapping the smallest numbers onto the shortest strings, and larger numbers onto longer strings. Strings of the same length may be arranged alphabetically.

Let us further assume that L_{NR} is a language over a 26-letter alphabet.

Number terms will be assigned elements of $\mathbb{N}$ as their referents, and string terms will be assigned elements of L_{NR}. Predicates will be assigned subsets of L_{NR} (that is, elements of $pow(L_{NR})$), and formulas will be assigned elements of $\{\mathbf{true}, \mathbf{false}\}$. So referents in general are

[7] Technically, these are not quite the same functions as F_1, F_2 from L_1, since they must be defined for a broader set of expressions; but they work in intuitively the same way: $F_1(\alpha, \beta) = [\alpha\beta]$ and $F_2(\alpha) = \#\alpha$.

drawn from $\mathbb{N} \cup L_{NR} \cup pow(L_{NR}) \cup \{\mathbf{true}, \mathbf{false}\} \cup \{\dagger\}$; we now call this set B.

With this in mind, we may now define our semantic operations as follows. G_1 and G_2 are the same as before (but extended to deal with our expanded set B); G_3 simply checks to see if its second argument is a member of its first argument:

(7) a. If $x, y \in \mathbb{N}$, then $G_1(x, y) = x + y$.
(If $x \notin \mathbb{N}$ or $y \notin \mathbb{N}$, then $G_1(x, y) = \dagger$.)

b. If $x \in \mathbb{N}$, then $G_2(x) = f(x)$.
(If $x \notin \mathbb{N}$, then $G_2(x) = \dagger$.)

c. If $x \subseteq L_{NR}$ and $y \in L_{NR}$, then $G_3(x, y) = \mathbf{true}$ if $y \in x$ and $G_3(x, y) = \mathbf{false}$ if $y \notin x$.
(If $x \not\subseteq L_{NR}$ or $y \notin L_{NR}$, then $G_3(x, y) = \dagger$.)

B is closed under these operations, so we let our semantic algebra $\mathfrak{B}$ be $\langle B, G_1, G_2, G_3 \rangle$.

We assign referents via a function h, much like the one from L_1, but extended to deal with predicates and formulas:

(8) a. $h(0) = 0$

b. $h(1) = 1$

c. If α is a predicate, then $h(\alpha) \subseteq L_{NR}$.

d. $h(\mathbf{TL}) = \{x \in L_{NR} | x \text{ consists of at least two letters}\}$

e. If $h(\alpha), h(\beta) \in \mathbb{N}$, then $h([\alpha\beta]) = h(\alpha) + h(\beta)$.
(If $h(\alpha) \notin \mathbb{N}$ or $h(\beta) \notin \mathbb{N}$, then $h([\alpha\beta]) = \dagger$.)

f. If $h(\alpha) \in \mathbb{N}$, then $h(\#\alpha) = f(h(\alpha))$.
(If $h(\alpha) \notin \mathbb{N}$, then $h(\alpha) = \dagger$.)

g. If $h(\alpha) \subseteq L_{NR}$ and $h(\beta) \in L_{NR}$, then $h(\alpha(\beta)) = \mathbf{true}$ if $h(\beta) \in h(\alpha)$, and $h(\alpha(\beta)) = \mathbf{false}$ if $h(\beta) \notin h(\alpha)$.
(If $h(\alpha) \not\subseteq L_{NR}$ or $h(\beta) \notin L_{NR}$, then $h(\alpha(\beta)) = \dagger$.)

It should be evident that h is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$. And just as in L_1, where the meanings of sentences were derived by a non-computable function, here in L_2, the referents of string terms are derived by a non-computable function.

Now consider the following sentence of L_2:

(9) **TL**(#[[[[[[[[[[[[[[[[[[[[[[[[[[[11]1]1]1]1]1]1]1]1]1]1]1]1]1]1]1]1]1]1]1]1]1]1]1]1]1])

This sentence consists of the predicate **TL** followed in parentheses by # and 28 occurrences of 1 (with brackets).

This sentence makes a perfectly sensible and easily comprehensible claim, namely that the 28^{th} string of L_{NR} (when the strings of L_{NR} are arranged in order of increasing length) consists of at least two letters.

In fact it is clear from the rules above what *all* sentences of the form $\mathbf{TL}(\#\alpha)$ mean: any such sentence means that the n^{th} string of L_{NR} (in order of increasing length) consists of at least two letters, where n is the number of 1's in α. So despite our use of a non-computable operation in the reference algebra, we can understand the meanings of sentences. It appears that a limitation to computable functions is not a necessary prerequisite to comprehensibility, at least if we are considering compositionality of reference.

Moreover, it is apparent that sentence (9) is *true*, because L_{NR} was stipulated to be a language over a 26-letter alphabet. There are at most 27 strings of length less than 2: the empty string, and the 26 one-letter strings. The 28th string must contain at least two letters.

So here we have an example of a language which uses a non-computable semantic operation in its reference algebra, and of a sentence in that language whose truth value is derived in part via that non-computable operation — but where the sentence is not only comprehensible, but easily truth evaluable.

Does this mean that natural languages must have non-computable operations in their reference algebras? Of course it is only in very specialized contexts that we refer to specific strings in non-recursively-enumerable languages, but it does happen — for example, I've been doing it all through this paper — so we need to take some account of it. Whether or not we need non-computable semantic operations for this is more debatable, but is not, I think, completely implausible. Consider the English sentence in (10):

(10) Number twenty-eight consists of at least two letters.

A phrase like *Number twenty-eight* refers to different things in different pragmatic contexts. It seems to me that we must regard each context in which this phrase may be felicitously used as providing a domain of objects from which this reference is drawn, and a sequential ordering on those objects. Now relative to a given context, this phrase will refer to the 28^{th} item in the ordering. In a context which provides L_{NR} as the relevant domain and f as the relevant ordering, this phrase will refer to 28^{th} string of L_{NR} according to f, and sentence (10) will be true. But we also presumably want to claim that the name *twenty-eight* refers to the number 28 (in all contexts), and since this is a constituent of the phrase *number twenty-eight*, we will presumably have to apply this non-computable function f to 28 in deriving the referent of the whole phrase from the referents of its parts.

Whether this is done in the application of a semantic operation, or instead, we regard f as the referent (in context) of the word *number*,

or of some phonologically empty constituent, is less clear.[8] In the latter case, the non-computability is all in the denotations and not in the operations, which we could maintain to be computable. But an analysis that uses a non-computable semantic operation in its reference algebra is, I think, at least a reasonable competitor among the other candidate analyses; and most importantly, it cannot be ruled out on the grounds that sentences derived using non-computable semantic operations are necessarily incomprehensible.

5 C-Compositionality and H-Compositionality

In the preceding section, we considered compositionality of reference rather than meaning, because the referent of an expression is not generally a psychological object, and it therefore seemed less clear with reference than with meaning that there is a psychological reason for assuming that the operations of the semantic algebra must be computable. As we saw, it is possible to have a language in which the referents of complex expressions are derived via a non-computable operation from the referents of their parts; and that sentences whose truth values are assigned in part via such non-computable operations may nonetheless make sense, and be easily comprehensible — and even truth evaluable. But if we were able to evaluate the truth value of such sentences ((9), for example), we presumably did so through some sort of psychological computation. This shows that psychological issues are not irrelevant to compositionality of reference after all! But now we have a little paradox: We can easily see that sentence (9) is true — so we must have some way of calculating its truth value. But the semantic operations involved in deriving this truth value included one which was non-computable. How, then, was it possible to figure out the truth value?

Readers who view this primarily as a mathematical problem might point out that I have not proved that the referent of any particular string term in L_2 is non-computable — only that function f, taken as a whole, is non-computable. The first twenty-eight pairs of f form only a finite set and are therefore trivially computable. So in fact there *is* a computable function assigning a referent to the string term in (9), and one could easily give a computational procedure for deriving the truth value of (9).[9]

[8] Note that in L_1 and L_2, I was free to simply stipulate what the semantic operations were, since these were artificial languages; no one can complain that in the "right" analysis, '#' should be treated as denoting f rather than as syncategorematic. But in English, the right syntactic analysis must presumably be discovered rather than stipulated.

[9] In fact, one could assign truth values to all the sentences of L_2 using computable

But this response misses a crucial point, namely that we can easily see that (9) is true *without* figuring out what the referent of its string term is. In fact we can do this even though I have not even specified what the alphabet of L_{NR} is — I haven't given enough information to figure out what the referent is even if all our operations *were* computable.

I think it is obvious, then, that we were able to compute the truth value of (9) because we were *not* computing it by first computing the referents of its parts. All we needed in order to compute the truth value were certain crucial pieces of background information (L_{NR} is over a 26-letter alphabet; f arranges strings by length), and some very limited information about the referents of the parts (the referent of the string term comes 28th in the ordering; the members of the referent of **TL** are all strings consisting of at least two letters).

I would suggest that this is actually typical of the way we calculate the truth values of sentences, when we do so in real, practical situations. We rarely if ever calculate the referent of each constituent, deriving each from the referents of its immediate constituents; but instead figure out the truth value from relevant background knowledge combined with very limited, possibly non-identifying information about the referents of the parts.

This point will not be controversial, I think; and although it is made particularly vividly by examples involving non-computable operations, it is obvious enough just from ordinary subject-predicate sentences. The most standard analysis of a sentence like *John smokes*, for example, treats *smokes* as denoting the set of things that smoke. No one would claim that people have to identify this set in any meaningful sense in order to evaluate whether the sentence is true. But if we don't even know what the referents of the parts of a sentence are, how can we calculate the truth value of the sentence by applying semantic operations to them? This is the sort of question that students ask in introductory semantics classes all the time, and I think most of us who teach such classes probably respond by making the correct and elementary point

---

functions by replacing G_2 with an infinite series of operations $G_2^1, G_2^2, \ldots$, each of which encoded only the first n pairs of f for some n. But presumably a system with this sort of infinite series of operations is no more psychologically plausible than one with non-computable operations.

It is also worth noting that the set of sentences of L_2 whose truth values we can figure out are those containing string terms denoting *all but* the first 27 strings of L_{NR} — there is an infinite number of sentences whose truth values we can figure out, and only a small finite number whose truth values we can't. The truth evaluable sentences of L_2 thus do not correspond to a recursively enumerable sublanguage of L_{NR}.

that you don't *need* to know the referent of *smokes* to calculate the truth value; all you need to know is whether John is member of it. But if this is right, then we are *not* calculating the truth value by starting from the referents of the parts and successively applying semantic operations; we start instead from very limited, potentially non-identifying information about the referents of the parts, and calculate the truth value from that, together with any relevant background knowledge.[10]

A defender of the view that the truth values of verifiable or falsifiable sentences are computed from the referents of their parts might respond as follows: In semantic analysis, we are free to posit as the referents of expressions any objects which do the work we theoretically require of referents. If there is a way to compute the truth value of a sentence, we may assign as the referents of its parts whatever objects are used in the computation. Then the truth value is guaranteed to be computable from the referents of its parts.

For example, one could computably assign truth values to all those L_2 sentences of the form $\mathbf{TL}(\#\alpha)$, where α contains at least 28 occurrences of 1, simply by letting $h(\alpha) = h(\#\alpha) = h(\mathbf{TL}(\#\alpha)) = \mathbf{true}$. Or, one could assign truth values to sentences like John smokes by letting *smokes* denote, not the set of things that actually smoke, but a representation of some language user's mentally encoded information about who smokes.

It is reasonable to expect that in any case where we have the psychological ability to verify or falsify a sentence, it will be possible to recast its semantics in a computable fashion by reassigning referents in this way — choosing as "referents" whatever is needed to make the computation work. Whether the objects assigned in this way really deserve to be called "referents," however, seems debatable. It seems to me that this technique gives a theory of something rather different from what we usually mean by reference and truth.

First, one should note that this approach is a kind of verificationism, since it yields truth values only for those sentences whose truth values can be computed. But intuitively, sentences may be true or false even if we have no way of discovering the truth value. Moreover, if the motivation for this move is to explain people's psychological abilities to verify sentences, it leads immediately to a mentalistic conception of reference, since any computation we do is presumably done on psychological representations. But intuitively, reference is normally to the

[10]The introduction of "witness sets" in Barwise and Cooper (1981) provides a classic discussion of a different kind of case in which truth values may be calculated in ways other than applying semantic operations directly to the denotations of the immediate constituents of the sentence.

mind-independent entities we talk about and make claims about.

It is clear that we do have psychological procedures for computing the truth values of sentences, and presumably any step in these procedures must be representable as a computable function. But it also seems clear that this computation is not necessarily done directly from the referents of the parts of the sentence — at least if we understand "referents" in the usual way — but rather from potentially very limited information about those referents, and from other relevant background information.

If we want to motivate the idea that truth values are assigned "compositionally" based on the psychological fact that people are able to figure out the truth values of novel sentences, then the principle of compositionality amounts to a claim that this kind of procedure exists. Let us call this conception of compositionality *C-compositionality* (C for "computable"). To say that truth values are assigned C-compositionally is to say that there is a way of computing them. If people can assign truth values to novel sentences, this shows that they must be employing some sort of C-compositional system. But to say that reference is assigned C-compositionally does *not* mean that it is possible to compute the reference of a complex expression from the referents of its parts — only that there is some way of computing the referents of complex expressions; and this may well be on the basis of information which is insufficient to identify the referents of the parts.

On the other hand, the formal notion of compositionality which we have from Montague requires that "compositionally" assigned values be derivable by operations from the values of their parts, but makes no claims that these operations correspond to any sort of computational procedure. Let us call this conception of compositionality *H-compositionality* (H for "homomorphic"). To say that reference is assigned H-compositionally is to say that the referent of a complex expression is determined by the referents of its (immediate) parts.

6 H-Compositionality, Substitution, and Intensionality

If C-compositionality explains peoples psychological abilities,[11] is there still any reason for maintaining a principle of H-compositionality? For the moment, let us continue to limit ourselves to reference assignment in considering this question. We will return to the issue of compositionality

[11] Of course I have not given an actual theory of C-compositionality here, so I do not claim to have explained people's psychological abilities to understand novel sentences or evaluate their truth values. But *some* theory must explain such abilities, and such a theory is, by definition, a theory of C-compositionality.

of *meaning* below.

I suspect many semanticists might justify H-compositionality by claiming that people employ the C-compositional system they do only because they know it will give the same results as an H-compositional system whose principles form a more basic part of their semantic knowledge. This view is reinforced by the fact that I presented the semantics of L_2 in H-compositional format, and did not specify a C-compositional procedure for it at all, yet we were able to compute the truth value of (9). We must have figured out some sort of C-compositional procedure for doing so, presumably based on the H-compositional semantics which was presented in (8) — since this was the only information about L_2 we had.

But to a skeptic, this kind of argument will probably not be compelling. This example really shows only that a C-compositional procedure can sometimes be constructed to match an antecedently given H-compositional semantics; it does not show that language users must have such an antecedently given H-compositional semantics in order to construct a C-compositional procedure. To make a convincing case for H-compositionality, we need to find something which H-compositionality explains but C-compositionality does not — and I hope it is by now clear that the ability of language users to deal with novel sentences will not serve this purpose.

I think the explanatory value of H-compositionality becomes clear if we recognize that it is equivalent to a familiar substitutivity principle. Specifically, reference is assigned homomorphically if and only if one expression may always be substituted for another expression with the same reference without altering the reference of any larger expression of which it is a part.[12] (A proof is given in the appendix.)

We should maintain the principle of H-compositionality of reference, then, precisely if we hold that reference is preserved under substitution of coreferring parts — regardless of any considerations having to do with language users' ability to understand novel sentences. It is on this basis, I suggest, that any principle of H-compositionality must be justified.

[12]Essentially this same mathematical point was made with a short proof-sketch by Muskens (1995): 89, though in a non-algebraic context, and in a discussion of compositionality of meaning and substitution of synonyms rather than compositionality of reference and substitution of coreferring terms. As Muskens notes, if we impose the further restriction that all semantic operations must be computable, the equivalence fails. Westerståhl (1998) and Janssen (2001) sketch essentially the same result, in an algebraic framework. Hodges (2001) gives a more detailed proof, also in an algebraic framework, and explores the relation between compositionality and substitutivity in detail.

As a corollary to this equivalence, we can see that any counterexample to H-compositionality of reference must take the form of some context where coreferring expressions cannot be substituted for one another without altering the reference of a larger expression containing them. But this is just the standard definition of an intensional context, so it follows immediately that *any counterexample to H-compositionality of reference must involve an intensional context.*

In fact, *all* intensional contexts are prima facie counterexamples to H-compositionality of reference. But there are familiar ways of maintaining the principle in the face of such counterexamples: we may claim that what appear to be coreferring expressions do not really corefer after all, at least in the contexts in question; or we may claim that what appears to be the result of substituting one expression for another really isn't. The first strategy is Frege's — expressions in intensional contexts are analyzed as referring to something other than their customary reference. The second strategy is Russell's — sentences are analyzed as having an abstract logical form of which the apparently coreferring expressions are not really constituent parts, or do not really occupy the same structural position, so that the substitution is not well-defined.

Both approaches come with some cost. Davidson (1968) famously suggested that "if we could but recover our pre-Fregean semantic innocence," it would seem implausible to us to claim that expressions in intensional contexts refer to anything different than they do elsewhere. But if the Fregean strategy involves a loss of semantic innocence, the Russellian strategy involves just as much a loss of *syntactic* innocence — which, if recovered, would make it seem implausible to claim that the sentence *Scott is the author of Waverly* has a radically different syntactic structure from *Scott is Scott.* Either way, we do some violence to our innocent (or perhaps one should say "naive") intuitions.

Despite these concerns, I think that H-compositionality of reference is an intuitively sound principle, which we should try to maintain. The intuition behind it — or really, behind the substitutivity principle to which it is equivalent — is just that (for any x, y) if you can say something truthfully about x, and x is the very same thing as y, then you can say the same thing truthfully about y. If you say something true, then swap out some part of the sentence and wind up saying something false, you must not have been talking about the same thing in the substituted part as you were in the original. This intuition seems at least as secure as those underlying the "innocent" perspectives, and is sufficient in my view to render something like a Fregean theory of indirect reference plausible. Any analysis which tries to maintain both semantic and syntactic innocence at the cost of H-compositionality of

reference must either reconcile itself with this intuition, or explain why it should be abandoned.

That having been said, both the Fregean strategy and the Russellian strategy, if left completely unconstrained, seem likely to rob the principle of H-compositionality of reference of all content. One suspects that if we are free to assign whatever structure we like to the expressions we are analyzing, an H-compositional reference assignment will always be possible; and likewise, that if we are free to use any arbitrary object as the indirect referent of an expression in intensional contexts, an H-compositional reference assignment will always be possible even if the syntax is antecedently given.[13] If we are to regard H-compositionality of reference as a real issue, then, it should probably be in the context of a theory which places specific constraints both on syntactic analysis and on indirect reference.

7 Compositionality, Computability and Meaning

Having seen that the use of non-computable semantic operations does not necessarily render sentences incomprehensible, and having separated C-compositionality from H-compositionality, and found at least some motivation for H-compositionality of reference, it is now worth returning to the issue of compositionality of *meaning*. Could the use of non-computable operations be called for here as well? More importantly, does H-compositionality of meaning have any conceptual or empirical motivation, or is it only C-compositionality which is motivated?

These issues are complicated by the fact that the term *meaning* is imprecise, coming as it does from our pretheoretic, non-technical vocabulary; we need to be more explicit about what we mean by it before we can expect to make much headway.

Obviously, if we simply *define* meaning as something which language users calculate in understanding each other's utterances, meaning will be C-compositional by definition; and there will be no obvious reason to suspect a role for non-computable operations, or to expect that meaning assignment will be H-compositional. But if, in developing a technical notion of meaning, we focus instead on the role meaning plays in the

[13] Janssen (1986):74-75 proves that, given a recursively enumerable language and a function assigning interpretations to its sentences, a grammar and semantic algebra can always be given for which a homomorphic interpretation is possible; Janssen (1997):455-456 additionally shows that if the interpretation function is computable, the resulting semantic operations will be too. Janssen's technique involves an unintuitive assignment of interpretations to non-sentential expressions, and therefore is available only if we allow ourselves a good deal of freedom both in assigning syntactic structure and assigning interpretations to the "parts" of a sentence.

determination of truth and falsity, the issue appears in a rather different light.

Intuitively, there are two factors which go into determining the truth value of a sentence: what it means, and what the world is like. The study of indexicality has shown the usefulness of considering the context of use separately from other aspects of "what the world is like," so we get a conception of the meaning of a sentence as something which, in combination with a context and a world, determines a truth value. Generalizing, the meanings of other expressions will be things which, again in combination with a context and a world, determine a referent. Meanings, then should correspond to, or be representable as, or identified with, functions from contexts and worlds to referents. In fact, this is precisely the position taken in Montague (1970), and it bears an obviously close relation to the notion of *character*, from Kaplan (1989).

If this is what we mean by *meaning*, we should note immediately that if the set of contexts or worlds is sufficiently large, then the set of functions from context-world pairs to possible denotations will be non-denumerable, and operations on this set will be non-computable by reasons of cardinality alone. But assuming there are only denumerably many expressions in a language, we really only need denumerably many meanings for them, so we need not regard all functions from context-world pairs to possible denotations as meanings. We may always limit the set B in our semantic algebra to the range of the meaning assignment h; then B will be no larger than the set A from our syntactic algebra.

If meaning determines reference, and the reference algebra requires non-computable operations, does this mean the meaning algebra will too? It turns out the answer is *no* — in some cases, we can locate the non-computability in the context, rather than the operations.

To see this, recall the analysis mentioned above for the English noun phrase *number twenty-eight*: we regard the context as providing a domain of relevant individuals, and a sequential ordering relation on that domain. Relative to a context c, the noun phrase will denote the 28^{th} element of the domain, according to the ordering. Let us suppose for the sake of argument that the phrase *number twenty-eight* is derived from *twenty-eight* by a syntactic operation which prefixes the word *number*, and consider what the corresponding operation in the meaning algebra might be like.

In pursuing this line of analysis, we might represent each context c as an ordered pair consisting of a non-empty set D and a sequential ordering σ on D. The meaning of *twenty-eight* should be the constant function mapping each such pair (and a possible world) onto the num-

ber 28 — in other words a set of pairs of the form $\langle\langle\langle D, \sigma\rangle, w\rangle, 28\rangle$. Note that σ itself is a set of ordered pairs.[14] The meaning of *number twenty-eight* should be the function mapping each pair $\langle D, \sigma\rangle$ (and world w) onto $\sigma(28)$; that is, a set of pairs of the form $\langle\langle\langle D, \sigma\rangle, w\rangle, \sigma(28)\rangle$. Each pair in the latter function corresponds directly to a pair in the former function, and there is a straightforward procedure for deriving it from the pair it corresponds to: just scan the pairs in σ to find the one whose left-hand member is the number 28, then pair $\langle\langle D, \sigma\rangle, w\rangle$ with the right-hand member. As long as the number of contexts and worlds is at most denumerably infinite,[15] we may use this procedure to compute the meaning of *number twenty-eight* from the meaning of *twenty-eight*, even if in some cases σ is non-computable. The semantic operation is computable, even though in a context which provides L_{NR} and a length-sorting function f as the values of D and σ, the corresponding operation in the reference algebra is non-computable.

On the other hand, we can also imagine a language in which reference to a non-computable function is built directly into an operation of the meaning algebra. Suppose for example that pragmatic contexts are not pairs $\langle D, \sigma\rangle$ as above, but just arbitrary objects. Now for each expression α in L_2, assign as its meaning the constant function which maps every context c and world w onto the referent assigned to it by the rules in (8) above. Now the relativization to worlds and contexts is vacuous (every expression is a non-indexical rigid designator), and the operation deriving the meanings of string terms from number terms must appeal to the non-computable function f which sorts L_{NR} by length.

It seems unlikely that a natural language could work this way, but not because a language with such an operation must be incomprehensible; on the contrary, this interpretation of L_2 is perfectly intelligible. What is implausible is that a natural language might have a dedicated grammatical construction for talking about strings in some other specific non-recursively-enumerable language. This pragmatic function is simply too specialized and comes up too rarely to get grammaticized.

I would suggest, then, that concerns about the ability of language users to understand novel sentences should not lead us to impose a computability constraint on meaning algebras any more than on reference algebras — at least if by *meaning* we mean something like func-

[14] By a *sequential ordering* on D I mean a function from the set of natural numbers, or an initial segment of the natural numbers, into D.

[15] Of course it might not be, but this does not affect the mathematical point — that a non-computable operation in the reference algebra can correspond to a computable operation in the meaning algebra.

tions from contexts and worlds to denotations. And just as with H-compositionality of reference, we should motivate H-compositionality of meaning on grounds other than this ability.

Just as H-compositionality of reference turned out to be equivalent to the principle that one expression can always be substituted for another with the same reference without altering the reference of any larger expression of which it is a part, H-compositionality of meaning is equivalent to the principle that one expression can always be substituted for another with the same meaning without altering the meaning of any larger expression of which it is a part. (The same proof will show this; see the appendix.) And just as with reference, we should maintain the principle of H-compositionality of meaning precisely if we accept this substitutivity principle, regardless of any considerations having to do with language users' abilities to understand novel sentences.

An interesting corollary of this equivalence is that in a language with no perfect synonyms, an H-compositional meaning assignment is always possible. This follows immediately, because in a language with no perfect synonyms, the substitutivity principle is vacuously satisfied.[16] It is inconsistent to maintain as a point of doctrine that there are no perfect synonyms, while denying that meaning is H-compositional.

It also follows that any counterexample to H-compositionality of meaning must take the form of a context where synonymous expressions cannot be substituted for one another without altering the meaning of a larger expression containing them. As with H-compositionality of reference, intensional contexts would appear to be the primary source of apparent counterexamples: it might be true that John believes there is a woodchuck in his yard, but false that he believes a groundhog is in his yard, if he does not know that *woodchuck* and *groundhog* are synonyms. Our strategies for trying to maintain H-compositionality of meaning in the face of such examples are basically the same as the strategies for trying to maintain H-compositionality of reference: we may claim that in such contexts, words don't mean what they customarily mean, but instead express some special meaning (a metalinguistic meaning, for example[17]); or we may claim that the syntactic structure is different

[16]We must construe *synonym* broadly enough here to include complex expressions with the same meaning, not just lexical items. Westerståhl (1998) also points out that compositionality is automatic in a language with no synonyms.

[17]Consider how odd it would be to say of a dog, for example, that it believed that a woodchuck was in the yard but not that a groundhog was in the yard. This sort of assertion seems to imply a knowledge of English, implying a metalinguistic component to the belief. Note that the oddity disappears if no contrast between the synonyms is asserted; it is not at all strange to say of a dog simply that it believes a woodchuck is in the yard — suggesting the metalinguistic interpretation is not

from how it appears, so that the substitution is not legitimate. And just as before, H-compositionality of meaning would seem to impose a significant constraint only if both these strategies are also significantly constrained.

H-compositionality of meaning is motivated in precisely the same way as H-compositionality of reference. If two expressions have the same meaning (correspond to the same function from world-context pairs to referents), but substituting one for the other results in change of meaning for the sentence as a whole (that is, a different function from world-context pairs to truth values), then there must be some world-context pair at which *reference* is determined non-H-compositionally. H-compositionality of reference thus commits us to H-compositionality of meaning (if we understand meanings to be functions from indices to referents). If the intuition underlying substitutivity for reference seems legitimate, we must accept it for meaning as well.

8 Conclusion

We have seen that a principle of homomorphic interpretation will not explain the ability of language users to compute the meanings of novel sentences. However, I argued that we should not resolve this issue by placing a computability requirement on the semantic algebras used in homomorphic interpretation. Even if non-computable operations are used in the derivation of sentences, these sentences may be comprehensible and even truth evaluable.

The important point here is *not* that we need non-computable operations in an adequate semantic theory. After all, the examples discussed in this paper are mainly from artificial languages, and have interpretations which could be used only in highly specialized technical contexts. Rather, the point is that we need to separate the issue of how people compute meanings, referents, truth values, etc. — essentially a processing question — from the issue of whether the interpretations of complex expressions are functionally determined by the interpretations of their immediate parts — essentially a structural question. Considering these issues separately leads to two separate principles corresponding to the traditional idea of compositionality: a principle of computable interpretation, or "C-compositionality," and a principle of homomorphic interpretation, or "H-compositionality."

If we motivate the "Principle of Compositionality" by pointing out the ability of language users to understand an indefinite number of novel sentences, and then take the central issue to be one of how this

systematically present in all belief ascriptions.

ability relates to syntactic structure (as suggested by Dowty (2007), for example), the theory we construct will most naturally be construed as a theory of C-compositionality. This line of investigation will not, I have suggested, lead naturally to a theory of H-compositionality — and might, in fact, cause us to miss the very real motivation that exists for such a theory.

The principle of homomorphic interpretation turns out to be equivalent to a familiar substitutivity principle: one expression may be substituted for another with the same interpretation without altering the interpretation of any complex expression of which it is a part. We should maintain the principle of homomorphic interpretation precisely to the degree that we accept this substitutivity principle, regardless of any psychological considerations.

The equivalence of the H-compositionality to substitutivity entails that counterexamples to H-compositionality of reference must come from intensional contexts, and that H-compositionality of meaning will always hold in a language with no perfect synonyms.

9 Appendix: Proof that interpretation is homomorphic if and only if the substitutivity principle holds

If by "interpretation" we mean reference assignment, this proof will show that reference is assigned homomorphically if and only if one expression may be substituted for another with the same reference without altering the reference of any larger expression of which it is a part. If by "interpretation" we mean meaning assignment, it will show that meaning is assigned homomorphically if and only if one synonym may be substituted for another without altering the meaning of any expression of which it is a part.

To prove these claims, we must first make them a little more precise. Given a syntactic algebra $\mathfrak{A} = \langle A, F_1, \ldots, F_i \rangle$, set B, and interpretation function $h : A \rightarrow B$, let us say that interpretation is homomorphic iff there are operations $G_1, \ldots, G_i$ on B such that $\mathfrak{B} = \langle B, G_1, \ldots, G_i \rangle$ is an algebra similar to $\mathfrak{A}$ and h is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$.

When h is construed as assigning referents, we understand α and β to corefer iff $h(\alpha) = h(\beta)$. When h is construed as assigning meanings, we understand α and β to be synonyms iff $h(\alpha) = h(\beta)$.

Where $\mathfrak{A} = \langle A, F_1, \ldots, F_i \rangle$ and $\alpha, \gamma \in A$, let us say that α is a *first level part* of γ iff $\gamma = F_n(\alpha_1, \ldots, \alpha, \ldots, \alpha_m)$ for some syntactic operation F_n. We will say that α is a *second level part* of γ iff α is a first level part of a first level part of γ; and more generally that α is a

$(j+1)^{th}$ *level part* of γ iff α is a j^{th} level part of a first level part of γ. We say that α is a *part* of γ iff α is a j^{th} level part of γ for some j.

If α is a first level part of γ (that is, if $\gamma = F_n(\alpha_1, \ldots, \alpha, \ldots, \alpha_m)$ for some F_n), then γ' is a result of substituting β for α in γ iff $\gamma' = F_n(\alpha_1, \ldots, \beta, \ldots, \alpha_m)$. If α is a j^{th} level part of γ ($j \geq 2$), then γ' is a result of substituting β for α in γ iff $\gamma = F_n(\delta_1, \ldots, \delta, \ldots, \delta_m)$ (for some F_n), $\gamma' = F_n(\delta_1, \ldots, \delta', \ldots, \delta_m)$, and δ' is a result of substituting β for α in δ.

By the *substitutivity principle* let us mean the claim that if α is a part of γ, and $h(\alpha) = h(\beta)$, and γ' is a result of substituting β for α in γ, then $h(\gamma) = h(\gamma')$.

An m-place *operation* on B is a set G of ordered $(m+1)$-tuples of elements of B, such that for any $x_1, \ldots, x_m \in B$, there is exactly one $y \in B$ such that $\langle x_1, \ldots, x_m, y\rangle \in G$. If this condition is met we may also write $y = G(x_1, \ldots, x_m)$.

Now we show that given a syntactic algebra $\mathfrak{A} = \langle A, F_1, \ldots, F_i\rangle$, set B, and interpretation function $h : A \to B$, interpretation is homomorphic iff the substitutivity principle holds.

Left to right (If interpretation is assigned homomorphically, an expression β may always be substituted for an expression α with the same interpretation, without altering the interpretation of any expression of which α is a part):

Assume interpretation is assigned homomorphically. This means there is a semantic algebra $\mathfrak{B} = \langle B, G_1, \ldots, G_i\rangle$ similar to our syntactic algebra $\mathfrak{A} = \langle A, F_1, \ldots, F_i\rangle$, and h is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$.

First we show that substituting β for α will not affect the interpretation of any expression γ of which α forms a first level part, then we show that if substituting β for α does not affect the interpretation of any expression of which α is a j^{th} level part, it will not affect the interpretation of any expression of which it forms a $(j+1)^{th}$ level part.

Suppose that γ has α as first level part; that is, that $\gamma = F_n(\alpha_1, \ldots, \alpha, \ldots, \alpha_m)$ for some F_n. Let γ' be a result of substituting β for α in γ; that is, $\gamma' = F_n(\alpha_1, \ldots, \beta, \ldots, \alpha_m)$. Because h is a homomorphism, $h(\gamma) = G_n(h(\alpha_1), \ldots, h(\alpha), \ldots, h(\alpha_m))$, and $h(\gamma') = G_n(h(\alpha_1), \ldots, h(\beta), \ldots, h(\alpha_m))$. But since $h(\alpha) = h(\beta)$, it follows that $h(\gamma) = h(\gamma')$. That is, substituting β for α will not affect the interpretation of any expression of which α is a first level part.

Now suppose that substituting β for α does not affect the interpretation of any expression of which α is a j^{th} level part. Assume α is a $(j+1)^{th}$ level part of γ. Then $\gamma = F_n(\delta_1, \ldots, \delta, \ldots, \delta_m)$, for some δ of which α is a j^{th} level part (and some F_n). Because α is

a j^{th} level part of δ, $h(\delta) = h(\delta')$, where δ' is any result of substituting β for α in δ. Let γ' be the result of substituting δ' for δ in γ. This means that γ' is also a result of substituting β for α in γ. Because h is a homomorphism, $h(\gamma) = G_n(h(\delta_1), \ldots, h(\delta), \ldots, h(\delta_m))$, and $h(\gamma') = G_n(h(\delta_1), \ldots, h(\delta'), \ldots, h(\delta_m))$. But since $h(\delta) = h(\delta')$, it follows that $h(\gamma) = h(\gamma')$. That is, if substituting β for α does not affect the interpretation of any expression of which α is a j^{th} level part, it will not affect the interpretation of any expression of which α is a $(j+1)^{th}$ level part.

Right-to-Left (If one expression may always be substituted for another with the same interpretation without affecting the interpretation of complex expressions of which it is a part, then interpretation is assigned homomorphically):

Assume that one expression may always be substituted for another with the same interpretation without affecting the interpretation of any complex expression of which it is a part.

Now for each m and each m-place syntactic operation F_n, let G_n be the smallest set of $(m+1)$-tuples such that if $\gamma = F_n(\alpha_1, \ldots, \alpha_m)$, then $\langle h(\alpha_1), \ldots, h(\alpha_m), h(\gamma)\rangle \in G_n$.

We show that G_n is an operation: If G_n were not an operation, there would be some α, β, x, y such that $h(\alpha) = h(\beta)$ and $x \neq y$, but $\langle h(\alpha_1), \ldots, h(\alpha), \ldots, h(\alpha_m), x\rangle \in G_n$ and $\langle h(\alpha_1), \ldots, h(\beta), \ldots, h(\alpha_m), y\rangle \in G_n$. By the definition of G_n, $x = h(\gamma)$ where $\gamma = F_n(\alpha_1, \ldots, \alpha, \ldots, \alpha_m)$, and $y = h(\gamma')$, where $\gamma' = F_n(\alpha_1, \ldots, \beta, \ldots, \alpha_m)$. But since substitution of one expression for another with the same interpretation will not affect the interpretation of expressions of which it is a part, $h(\gamma) = h(\gamma')$; that is, $x = y$. We now have $x = y$ and $x \neq y$; this is a contradiction, so G_n must be an operation.

By the definition of G_n, if $\gamma = F_n(\alpha_1, \ldots, \alpha_m)$, then $\langle h(\alpha_1), \ldots, h(\alpha_m), h(\gamma)\rangle \in G_n$. Since G_n is an operation, we may write this as $h(\gamma) = G_n(h(\alpha_1), \ldots, h(\alpha_m))$. That is, $h(F_n(\alpha_1, \ldots, \alpha_m)) = G_n(h(\alpha_1), \ldots, h(\alpha_m))$. This is the relevant condition to make h a homomorphism; simply let $\mathfrak{B} = \langle B, G_1, \ldots, G_i\rangle$ and it immediately follows that h is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$.

Acknowledgments

Thanks to Chris Barker, Theo Janssen, Dag Westerståhl, and the audience at the symposium, particularly Gertjan van Noord, for helpful discussion. Errors are my own, of course.

References

Barwise, Jon and Robin Cooper. 1981. Generalized quantifiers and

natural language. *Linguistics and Philosophy* 4:159–219.

Davidson, Donald. 1968. On saying that. *Synthese* 19:130–146.

Dowty, David. 2007. Compositionality as an empirical problem. In C. Barker and P. Jacobson, eds., *Direct Compositionality*. Oxford.

Frege, Gottlob. 1884/1980. *The Foundations of Arithmetic*. Evanston, Illinois: Northwestern University Press, 2nd edn.

Frege, Gottlob. 1892/1970. On sense and reference. In P. Geach and M. Black, eds., *Translations from the Philosophical Writings of Gottlob Frege*. Oxford: Basil Blackwell, 2nd edn.

Hodges, Wilfrid. 2001. Formal features of compositionality. *Journal of Logic, Language and Information* 10:7–28.

Janssen, Theo. 1986. *Foundations and Applications of Montague Grammar, Part 1: Philosophy, Framework, Computer Science*. No. 19 in CWI Tracts. Amsterdam: Stichting Mathematisch Centrum.

Janssen, Theo. 1997. Compositionality. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*, pages 417–473. Cambridge, Massachusetts: MIT Press.

Janssen, Theo. 2001. Frege, contextuality and compositionality. *Journal of Logic, Language and Information* 10:115–136.

Kaplan, David. 1989. On demonstratives. In J. P. Joseph Almog and H. Wettstein, eds., *Themes from Kaplan*, pages 481–563. Oxford: Oxford University Press.

Kracht, Marcus. 2003. *Mathematics of Language*. Berlin: Mouton de Gruyter.

Larson, Richard and Gabriel Segal. 1995. *Knowledge of Meaning: An Introduction to Semantic Theory*. Cambridge, Massachusetts: MIT Press.

Montague, Richard. 1970. Universal grammar. *Theoria* 36:373–398.

Muskens, Reinhard. 1995. *Meaning and Partiality*. Stanford, California: CSLI Publications.

Partee, Barbara, Alice ter Meulen, and Robert Wall. 1990. *Mathematical Methods in Linguistics*. Dordrecht: Kluwer Academic Publishers.

Pelletier, Francis Jeffry. 2001. Did Frege believe in Frege's principle? *Journal of Logic, Language and Information* 10:87–114.

Westerståhl, Dag. 1998. On mathematical proofs of the vacuity of compositionality. *Linguistics and Philosophy* 21:635–643.