<http://www.diva-portal.org>

Postprint

This is the accepted version of a chapter published in *Oxford Handbook of Compositionality*.

Citation for the original published chapter:

Pagin, P. (2012)

Communication and the Complexity of Semantics.

In: Wolfram Hinzen, Edouard Machery, and Marcus Werning (ed.), *Oxford Handbook of Compositionality* (pp. 510-529). Oxford: Oxford University Press

N.B. When citing this work, cite the original published chapter.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:su:diva-85052>

Communication and the Complexity of Semantics*

Peter Pagin

June 30, 2008, 20:18

1. Introduction

A celebrated argument for the claim that natural languages are compositional is the *learnability argument*. Briefly: for it to be possible to learn an entire natural language, which has infinitely many sentences, the language must have a compositional semantics. This argument has two main problems: One of them concerns the difference between compositionality and *computability*: if the argument is good at all, it only shows that the language must have a *computable* semantics, which allows speakers to compute the meanings of new sentences. But a semantics may be computable without being compositional (and vice versa). Why would we want the semantics to be compositional over and above being computable? The learnability argument doesn't tell us.

The idea that is developed here is that we get further requirements on semantics by looking at linguistic communication, and in particular at the feature that we manage to convey new contents by means of new sentences *in real time*, i.e. that a hearer manages to compute the meaning on-line of an uttered sentence at speed that matches the speed of speech. It would seem that this can be explained only if the computation steps needed for interpretation are comparatively few and

*The final text is improved thanks comments from two anonymous referees. I have also benefited from discussing these issues with Martin Jönsson and Ken Chan at Rutgers, Alasdair Urquhart in Toronto, and over several years with Kathrin Glüer and Dag Westerståhl. An earlier version was presented at the Rutgers Compositionality series, spring 2007, and a still earlier version in the Compositionality Workshop in Paris 2004 (from where I am grateful for comments by Tim Fernando and Wilfrid Hodges). The research has been supported by grants from the Tercentenary Foundation of the Swedish National Bank, from The Swedish Research Council, and from the Spanish Ministry of Science and Education during a research stay in Barcelona (in the program *Profesores e investigadores extranjeros de acreditada experiencia, en régimen de año sabático en España*).

easy. We can even claim that between two semantic theories, if one allows for less complex computation, then it helps explaining on-line interpretation *better* than its rival. This is elaborated upon in section 3.

In order to justify a particular kind of semantics in this way, we would want to show that a semantics of kind in some respect *minimizes* computational complexity. The relevance of computational complexity for cognition is discussed in section 4. Sections 5-6 are devoted to selecting an appropriate complexity measure. Section 7 discusses the nature of minimal complexity under this measure. Section 8, finally, sketches an argument that the class of minimal semantics is derived from standard compositional semantics by means of one restriction and one generalization. The net result is that complexity considerations do give us a new reason for believing that natural language semantics is compositional (in the restricted way), or else has generalized (restricted) compositionality.

Before proceeding with the later sections, we will need to relate the concepts of compositionality, computability and recursiveness. This is the task of the next section.

2. Preliminaries

I shall call a function μ that maps syntactic items on meanings (irrespective of what entities serve as meanings) a *semantic function*. I shall call a function ρ_i that for some n maps meanings $m_1, \dots, m_n$ on a meaning m a (meaning) *composition function*. A *generalized composition function* ρ is then a function such that, given a language L , for any syntactic operator α in L , $\rho(\alpha)$ is a composition function. Then, a semantic function μ for a language L is *compositional* just in case there is a generalized composition function ρ such that for each operator σ in L and any relevant syntactic items $t_1, \dots, t_n$ (with μ defined for $\sigma(t_1, \dots, t_n)$) it holds that

$$(PC) \quad \mu(\sigma(t_1, \dots, t_n)) = \rho(\sigma)(\mu(t_1), \dots, \mu(t_n)).$$

Intuitively, (PC) says that the meaning of the complex is a function (ρ) of the meanings of the parts and the mode of composition (σ).¹

The syntactic items may be expressions, i.e. surface strings. But in general strings are syntactically ambiguous in that they can be generated in more than one way from atomic expressions and operations. The semantic function must take disambiguated items as arguments (since the meaning may depend on the derivation, not just on the resulting string). Hence, when expressions are ambiguous, expressions cannot (always) be the arguments. Instead, it is common to take the arguments to be *terms*, whose surface syntax reflect the derivation of the string. To give a simple example: where σ is an operation that maps a noun phrase and a verb phrase on a sentence by means of concatenation, we have the string ‘John_ runs’ (where ‘_’ marks the word space that is part of the string), with the corresponding term ‘ $\sigma(\text{John}, \text{runs})$ ’.

Here I shall be concerned with the syntactic terms and regard the syntactic domain T as a domain of terms. The domain E of *expressions* is derived from T by an evaluation function V . V corresponds, in classical generative grammar, to a mapping from deep structures to surface structures.²

Now, it is clear that a compositional semantics need not be recursive. For the semantic function μ is recursive just in case the generalized composition function ρ is recursive, but it is not required in the definition (PC) (or in any common definition of compositionality) that ρ be recursive. It must be a function of the right type, i.e. with the right arguments and values, that is all. Hence, compositionality does not entail recursiveness.

Neither does recursiveness entail compositionality. In arithmetic, a recursive function is either a projection function that selects an argument from a sequence of arguments, the constant zero function, the successor operation s , or a function defined from these by means of function composition, primitive recursion, or minimization. The counterpart to the successor operator in the syntactic domain is the collection Σ of syntactic operators σ_i . These operators are in general partial.

¹The advantage of this format is that if σ is just the first argument of ρ , the arity of ρ would vary if there are syntactic operators of different arities.

²I am by and large using the framework proposed by Wilfrid Hodges and used e.g. in Hodges 2001, Westerståhl 2004, and Pagin 2003.

The set of well-formed terms T is defined inductively from a finite set A of primitive expressions by means of the syntactic operations σ_i of Σ . Hence, the semantic function μ differs in one respect from arithmetic functions in that in general the domain is defined by more than one construction type.

The semantic function differs in one other important respect from an arithmetic function, since it maps entities between domains, from a syntactic to an ontic or conceptual domain of meanings (I shall refer to this as *the conceptual domain*). Therefore, to have a recursive semantic function, we need not only recursion over syntax, but also recursion over the conceptual domain. In order for this to make sense, we must regard the meaning domain M as being inductively defined from a finite set B of *basic* meanings, by means of a collection Γ of basic meaning composition functions γ_i . In this case the elements γ_i of Γ correspond to the successor operation of arithmetic. New functions can be defined from B and Γ by means of function composition, primitive recursion and minimization. Let $\bar{\Gamma}$ be the closure of Γ under these operations.³

The situation is in fact more complicated, since what corresponds to the composition functions in the compositional case, the elements of $\bar{\Gamma}$, will take arguments both in the syntactic and the conceptual domain (but their values will be in M). I shall refer to them as *mixed* composition functions. The domain of these functions is the *union* $U = T \cup M$ of the syntactic and the conceptual domains. The semantic function μ is then defined by simultaneous recursion over T and M .

For a function to be recursive over U , it is then required that it be a constant function with a basic meaning as value, a projection function, a member of Σ or Γ , or defined from these by means of function composition, primitive recursion or minimization. In arithmetic, the minimization $\text{Mn}[f](x_1, \dots, x_n)$ is a function that gives y as value if y is the smallest x such that $f(x_1, \dots, x_n, x) = 0$, with f defined for all $x_1, \dots, x_n, x$ with $x < y$, and undefined if no such number exists. Since the syntactic and conceptual domains in general contain several minimal elements and several generating operations, there is in general no direct analogy. Rather, we would have to fix one by means of stipulations such as lexicographic

³ I shall let the variables ' γ_i ' etc. vary over the elements of $\bar{\Gamma}$.

orderings. Since this would be arbitrary and since minimization has not played any role in any semantic system I know of, minimization will be ignored. Instead we will consider the counterparts to primitive recursion.

The function composition ingredient generates new functions in accordance with *composition equations* of the following general format:

$$(FC) \quad f(\vec{x}) = g(h_1(\vec{x}), \dots, h_n(\vec{x}))$$

where ' $\vec{x}$ ' is short for ' $x_1, \dots, x_m$ '. Here the function f is defined by composition from the functions $g, h_1, \dots, h_n$.

The primitive recursion ingredient instantiate *recursion equations* of the following format for the application of the semantic function μ :

- (Rec) i) For each simple term $t \in A$ there is a function $\gamma_t \in \bar{\Gamma}$ such that $\mu(t) = \gamma_t(t)$
- ii) For any n and operation $\sigma_i \in \Sigma$ of arity n there is a function $\gamma_i \in \bar{\Gamma}$ such that for all terms $t_1, \dots, t_n$, if μ is defined for $\sigma_i(t_1, \dots, t_n)$, then

$$\mu(\sigma_i(t_1, \dots, t_n)) = \gamma_i(t_1, \dots, t_n, \mu(t_1), \dots, \mu(t_n))$$

Here, it is immediate from the recursive clause, ii), that μ is directly defined by recursion over T . If γ_i is also defined by means of primitive recursion, this will be recursion over M , or over both T and M .

We can note that the requirement of clause i) is usually met by a simple list of the values of μ for each simple term t . Secondly, we can observe that the requirement in clause ii) that $\gamma_i \in \bar{\Gamma}$, i.e. that γ_i be recursive over U , imposes a restriction that is not imposed in (PC). Thirdly, the fact that the functions γ_i take the syntactic terms $t_1, \dots, t_n$ themselves as arguments, has the effect that the compositional substitution laws need not hold. For

$$\gamma_i(t_1, t_2, \mu(t_1), \mu(t_2))$$

may well differ from

$$\gamma_i(t_1, t_3, \mu(t_1), \mu(t_3))$$

even if $\mu(t_2) = \mu(t_3)$. Hence, recursiveness does not entail compositionality.⁴

The concept of (effective) *computability* is closely connected to that of recursiveness. Intuitively, for a function f to be effectively computable is for there to be an effective/mechanical procedure p such that for any argument x for which f is defined, p as applied to x terminates after a finite number of steps with giving the value of f for x . The paradigm of effective computability is a Turing machine. Arithmetical recursive functions are Turing computable, and Turing computable arithmetic functions are recursive (see e.g. Boolos, Jeffrey, and Burgess 2002, chapters 7-8). The claim that all arithmetic functions that are computable in the general intuitive sense are recursive is known as *Church's Thesis*.

When we move from arithmetic to the field of natural language semantics, we can satisfy demands of intuitive computability by ensuring that the functions involved are Turing computable. This can be shown directly, in part by showing how the formal substitution operations induced by the recursion equations can be executed.

With these basic concepts in place, I turn to the reason for justifying compositionality by appeal to communication.

3. From learnability to communication

Almost all of the standard arguments for the claim that natural languages are compositional suffer from severe flaws.⁵ I'll take the *learnability* argument as the prime example. This argument was given by Donald Davidson in 1965. The argument, applied to some language L , can be presented as follows:

- (LA) a) There is an infinite set M of meanings.
 b) Each disambiguated expression in L has at most one meaning.

⁴As pointed out in Werning 2005 and developed in Pagin and Westerståhl 2008, we can have a natural recursive semantics for quotation that is not compositional. An example can be found in Potts 2007. Cf. the present section 8.

⁵ For a brief survey of arguments for compositionality and a critical discussion, see Pagin and Westerståhl 2008.

- c) For every possible context c , every element in M is the meaning in c of some expression in L .
- d) Humans can learn at most finitely many basic expression-meaning pairs.
- e) Humans can learn L .
- f) Hence, L has a compositional semantics.⁶

The main idea of this argument is that since it is impossible to learn the meanings of all expressions if all expression-meaning pairs are basic, some have to be non-basic, i.e. derived, and if the meaning of an expression is derivable, the language must be compositional.

There are two main problems with this argument. The first problem is the premise (LAc), for it is a very strong assumption that there are infinitely many meanings expressed, in any context, by sentences of a natural language, say English. Since at any time t , speakers of the language have used at most finitely many sentences, it follows that there are (infinitely many) sentences that have not been used but are nonetheless meaningful.⁷ On what basis do we claim that sentences that no speaker has used is already endowed with a meaning, waiting to be correctly computed by the speakers of English? The most natural justification of this claim would be that English has a compositional semantics, for with such a semantics, the meanings of unused sentences may already be determined. But with such a justification of (LAc), the (LA) argument involves a *petitio principii*. We need either a different justification for (LAc), or a replacement of (LA).

The second problem with (LA) is that the condition of allowing for meanings of some expressions to be *derivable* does not require a compositional semantics. It does require a *computable* semantics. As we saw above, however, a function may be recursive (and hence computable) without being compositional. The learn-

⁶Often some of the elements in this arguments are slurred over: It is not enough to say that there are infinitely many meaningful sentences to be learned, for if every sentence has the same meaning, the task of learning the meaning of every sentence is not that hard. Therefore, we need the condition that there are infinitely many expressible meanings. But further, it is not enough to say that there are infinitely many meanings that can be expressed, for with the single sentence 'I am here', infinitely many propositions can be expressed, by using it in different contexts.

⁷ The alternative premise, instead of (LAa), that M is finite but larger than the set of sentences used at t , would work as well, and have the same problem.

ability requirement therefore, barring the first problem, provides an argument for computability, and therefore (assuming Church's Thesis in the domain of semantics) recursiveness, but not an argument for compositionality. The same problem afflicts several related arguments.⁸

If we turn from learnability to communication, the first problem disappears. The reason is that in each case of linguistic communication, the speaker associates a propositional content with the sentence she uses and the hearer associates a propositional content with that same sentence (on that occasion), and there is or there is not agreement between them in the sense that they associate the same content or at least contents that are sufficiently similar for communication to succeed. Now there is a question of what explains why speaker and hearer associate the same or similar contents with the same sentences. We need not assume that there is any correct or incorrect association of content with a new sentence; all we need care about is the coincidence.

Compositionality enters the picture because it helps explaining the rate of success in linguistic communication when the sentence used or the content communicated is *new*. The first to emphasize this role for compositionality was Gottlob Frege:

It is astonishing what language can do. With a few syllables it can express an incalculable number of thoughts, so that even a thought grasped by a terrestrial being for the very first time can be put into a form of words which will be understood by somebody to whom the thought is entirely new. This would be impossible, were we not able to distinguish parts in the thought corresponding to the parts of a sentence, so that the structure of the sentence serves as an image of the structure of the thought (Frege 1923, opening paragraph).

⁸ Jeff Pelletier (1994) similarly argues that it is enough if a semantics is what he calls *grounded*. Being grounded in Pelletier's sense is somewhat looser than being recursive: what corresponds to recursion equations in Pelletier's examples allows a clause like $\mu(\sigma(t)) = r(\mu(\beta(t)))$, for some function r , where $\beta(t)$ may be a *successor* of t . In general, derivations with such equations are not guaranteed to terminate. According to Pelletier, it is enough if they in fact do, and only if they do is the semantics grounded. As already shown, however, ordinary recursiveness is enough for exemplifying a semantics that is computable but non-compositional.

This passage is remarkably rich in content. Frege correctly points to the infinite expressive power of a language (rather than its infinite syntax) as the important feature. Frege also draws attention to *communication* as the phenomenon that is crucial for the systematic nature of semantics. In this context, communication has three features that contrast with learnability. First, communication relates two *speakers* who each has an independent interpretation of a sentence. As we saw, this is an advantage compared with the learnability argument.

Secondly, communication relates a speaker and a hearer, with different roles. The speaker is to find a suitable sentence to express a Thought, while the hearer is to find a suitable Thought for interpreting the uttered sentence, and this asymmetry demands more of semantics than interpretation does by itself.⁹ Frege's main claim is that the ability to communicate new thoughts would not be possible without an isomorphism between the sentence and the thought.¹⁰

There are several problems with this claim, but the most relevant problem concerns computability. In order that the hearer can *compute* the right content from the (parsed) expression, all we need is that the semantic function is recursive. Why would we also want an isomorphism, or that it be compositional?

This question brings us to the third respect in which communication contrasts with learnability: communication takes place in *real time* and under tight time constraints, while learnability has no temporal significance at all and sentence understanding as such (as opposed to the processing of tokens) is not a real time process (at best, the resulting end state of understanding can be assigned a time point).

During oral communication, speaker and hearer have to figure out, respectively, what expression to use and what it means, on-line, at a high speed. The *complexity* of the cognitive task therefore becomes crucial: the complexity of the articulation and interpretation tasks cannot be too high, on pain of making our

⁹This idea is developed in detail in Pagin 2003.

¹⁰The isomorphism requirement is stronger than the compositionality requirement in the respect that the semantic function must be one-one, while compositionality only requires many-one. It also requires meanings to be structured (since otherwise there is automatically an isomorphism), while compositionality does not. Taken literally, it is in another respect weaker: that for every sentence *s* there is an isomorphism between *s* and its meaning does not entail that there is *one* semantic function μ that for every sentence *s* is an isomorphism between *s* and $\mu(s)$.

near-immediate communication impossible. Computability in principle does not put any upper bound on the (finite) complexity of the computation task. In order to secure a reasonable complexity, we must look for some other property.

The general idea, then is to justify further properties of the semantics by appeal their role in minimizing complexity. It turns out that compositionality has a role to play in that enterprise. First, we shall need to discuss the cognitive relevance of computational complexity measures.

4. Computational complexity and cognitive tasks

The speaker has the task of *articulation*, i.e. of finding and uttering a suitable linguistic expression for conveying her thought content to the hearer, and the hearer has that of *interpretation*, i.e. of finding a suitable thought content to associate with the expression uttered. Here, I shall focus on the task of the hearer.

We want to know ultimately what properties a semantic function should have so as to be least taxing for the hearer to compute. To this end we want to provide some measure of the complexity of the task. That measure will have to be mathematical in character. But this already introduces a risk of misrepresenting the processes. For whether a certain interpretation task *s* is more or less complex than a task *s'* depends not only on obvious differences in some size or other, but also on what happens to be easier for *us*, human language users, given our cognitive architecture. To take an example from the literature, consider the sentences

- (1) a. The reporter who attacked the senator admitted the error.
- b. The reporter who the senator attacked admitted the error.

(example taken from Gibson 1998, 2). It is well-known in psycholinguistics that certain subordinate constructions are more difficult to process than others. Typically, we find *subject*-extracted relative clauses, as in (1a), easier to process than *object*-extracted relative clauses, as in (1b). There is no obvious mathematical reason why this should be so. The tree structures of the two complementizer phrases in (1) appear to be of about the same complexity, with an equal number of nodes, and resulting from one another by the interchange of the two NP nodes, as shown

Figure 1: Complementizer phrases of (1).

in Figure 1. If anything, the a) tree is intuitively more complex, since it has five levels while b) has four. A proposed explanation (cf. Gibson 1998) is that the distance between the relative pronoun ‘who’ and its corresponding trace t_i is greater in b) than in a). Processing from left to right involves keeping elements in mind that will be tied to other elements later on, and so the longer it takes before a relative pronoun can be anchored to a position in the subordinate clause, the heavier short term memory is taxed.

Whether the explanation is right or wrong, the phenomenon exemplifies the general fact that results that depend on mere intuitive assessments of mathematical complexity are subject to correction by empirical studies. Nevertheless, the difficulty in this case can be modeled and measured by a mathematical property—the left-to-right distance (in the tree/term) between the relative pronoun and the coordinated empty position in the relative clause. That property might seem to be completely unrelated to compositionality, since the left-right order of expressions, as opposed to the constituency order, is in principle irrelevant to the structure of the underlying syntactic term. However, minimizing left-to-right distance between a surface element and the last further surface element needed to process it, is an example of a general complexity principle that John Hawkins (e.g. in Hawkins 2003, 144) has called *Maximize On-Line Processing*. According to that principle, grammatical systems of natural languages tend to be organized so as to minimize the effort of processing elements on-line, i.e. when and where they

are encountered in an utterance or a text. As we shall see, minimizing general complexity is closely connected with maximizing on-line processing. The example suggests that we have reason to be optimistic about the cognitive relevance of general computational measures of complexity.

5. Complexity and efficiency

In computational complexity theory, three types of measure have been studied extensively, all defined in terms of Turing machines:

The *time complexity* of a problem P (relative to a way of describing P) with respect to an algorithm A , is the maximum number of computation steps that are needed for a Turing machine that implements A to solve a problem instance of the same *size* as P (cf. Garey and Johnson 1979, 6, 26).

The *space complexity* of a problem P (relative to a way of describing P) with respect to an algorithm A , is the maximum number of distinct tape squares visited by a Turing machine that implements A to solve a problem instance of the same *size* as P (cf. Garey and Johnson 1979, 170).

The *Kolmogorov complexity* of a problem P (relative to a way of describing P) is the size (relative to a linear encoding of Turing machines) of the smallest Turing machine needed to solve P (cf. Li and Vitányi 1997, 93-98).

In these contexts, a “problem” is usually a problem *type*, and what is solved in each particular case is an *instance* of that problem type. Such a problem type is e.g. *The Traveling Salesman*: a salesman is to visit a number of cities exactly once and then return home, and the task is to find a visiting order that minimizes the total distance traveled. The number of cities is the *size* of the instance.

Depending on the choice of complexity measure and on the choice of method for solving problems of a chosen type, a particular problem instance gets assigned a natural number as the complexity of that instance, numbering e.g. the squares that a chosen Turing machine has visited for computing the solution. One is interested not only and not primarily in the complexity of individual instances, but in the *maximal complexity* of instances of the same *size*: given e.g. a number of cities to visit (disregarding further information), what is the maximal number of

steps needed to determine the best order? Given a problem type P and a method ψ , one is therefore interested in the *complexity function* $C_{P,\psi}$ from natural numbers to natural numbers which, for a given argument k gives as value the maximal complexity of P -instances of size k .

Suppose we can hold P constant. We can then simply associate each *method* ψ with a complexity function C_ψ . When each method is associated with a complexity function, we can compare methods with respect to *efficiency*: if for all k it holds that $C_\psi(k) < C_\epsilon(k)$, then we can say that method ψ is more *efficient* than method ϵ . In general, the efficiency comparison is less straightforward, since one method may be more efficient than another only in the long run. The most natural way of capturing this idea is to require that from some size onwards, the *sum* of the C values are lower for the one than for the other:

(EC) ψ is more efficient than ϵ iff there is an n such that

$$\forall k > n \left[\sum_{i=1}^k C_\epsilon(i) - C_\psi(i) > 0 \right]$$

We are interested in finding out the properties of those interpretation methods, i.e. semantic functions, that are *most efficient* in this sense. As we shall see below, however, this comparison is significant only for large differences.

6. Measures of complexity

We turn now to the question selecting an appropriate complexity measure. Which of the three aforementioned types of complexity are relevant to an intuitive measure of the difficulty of the cognitive task? Time complexity appears to be most directly relevant, since considerations of time pressure motivated looking at complexity in the first place. The general idea of time complexity is that we count the number of steps that have to be taken for completing a process. It is plausible to assume that by and large, an increase of the number of steps required in the formal computation corresponds to an increase of the real time needed for human processing. I shall assume so. Hence, we will in the first place want to minimize time complexity of individual tasks, and therefore to maximize efficiency of semantic

functions with respect to time complexity.

The next questions to answer are: What is the character of the relevant problem type? How do we measure the size of an instance? What constitutes an individual step of the computation process?

It would very natural at first to take maximizing efficiency to consist in finding the most efficient semantic function for a given language L and a given domain M of meanings, where L is understood purely syntactically. That is, measure the size of the problem instance as the size of the term, and look for the longest computation needed for a term of that size. But although this question makes sense, there is a good reason to look at the issue differently. For with some choices of L and M , semantic interpretation becomes *intractable*, whatever the semantic function. To see this, consider the following example of *Lisa*.

We have a language L consisting of α and the successor operator $'$ as basic elements. So L consists of the terms $\alpha, \alpha', \alpha''$ etc. Then we have a conceptual domain consisting of the object l (Lisa) and the two conceptual functions f (father) and m (mother). So M consists of $l, m(l), f(l), m(m(l)), m(f(l)), f(m(l)), f(f(l))$ etc. Let the *size* of an object of either domain consist in its number of basic elements (basic object plus number of function/operator occurrences), and let the *size* n of a domain be the number of elements of at most size n it contains. The *growth rate* of the domain is a rate of increase of the function $g(n)$ that maps a number n on the number of elements of domain with size n or lower. In these terms, we have a conceptual domain with a growth rate of $g(n+1) = 2g(n) + 1$ and a syntactic domain with a growth rate of $g(n+1) = g(n) + 1$.

In order that the semantic function μ for each element m in M up to size n maps a term t on m , $2^{n+1} - 1$ distinct terms will be needed, since there are that many elements of M of size n or lower. Therefore, there will be at least one element m_i and one term t_j of L such that $\mu(t_j) = m_i$ and the size of t_j is $2^{n+1} - 1$ or greater. Assuming that exactly one computation step is needed for processing each element of a term t , at least $2^{n+1} - 1$ steps are needed to compute t_j .¹¹ Hence,

¹¹ This assumption is made for ease of exposition. It does not matter much. It will in any case hold that there is a finite number k such that *in the long run*, at most k elements can be computed in each step. Then there is still exponential growth.

the maximal number of steps needed to process a term referring to an object of a certain size grows *exponentially* with the size of the object, regardless of the choice of semantic function. In terms of computational complexity theory, using L for referring to M in the *Lisa* example is a strictly *intractable* task. When tasks are considered tractable, the increase in time complexity (number of steps needed) is at most a polynomial function.

More precisely, we get the intractability result if we regard the *problem type* as the type of understanding the expression of a concept, for then the size of a problem instance is the size of that concept. If by contrast we regard the problem type as that of processing a *term*, then the size of the problem instance is the size of the term. The time complexity function, under the assumption above, is then simply x : as many steps are needed as there are elements in the term. From that point of view, the semantic function appears very efficient. For the same conceptual domain M we could have a more appropriate language L' and a semantic function μ' mapping L' on M that would require, say, terms that are twice as large as the concepts they are mapped on. With the same assumption of one step per element of the term, μ would be as efficient as μ' if the problem type is that of processing terms, but exponentially less efficient if the problem type is that of arriving at contents. The latter is clearly what is cognitively more relevant: $L + \mu$ form an intractably cumbersome combination for talking about or expressing M , while the alternative $L' + \mu'$ would be manageable. The upshot is that the problem type should be that of *interpreting expressions of content*, and that hence the measure of the size of the instance of that problem simply is the size of the content. That is the invariant factor in the comparison between methods.¹² We have to take account not only of the efficiency of the mapping from code to concepts, but also of the efficiency of the encoding itself, i.e. the size of the code.¹³

¹²A further reason not to use the term size as the size of the problem instance is that we can make terms arbitrarily much larger by throwing in junk constituents that are not needed for the semantics and therefore do not add to computation complexity. With a lot of junk in the terms, a semantic function can appear to be more efficient, which is again counterintuitive.

¹³ This aspect of the issue shows the similarity with questions of efficient encoding handled in Mathematical Information Theory, as originated in Shannon 1949. There are also important differences, however. An encoding E is efficient in the information theoretic sense if the average rate of information sent over an information channel and encoded by E is high. In that context, a signal conveys more information if the fact that it reports is less probable. States of affairs that are highly

But how do we measure the size of contents? Does it make sense to say that one concept or one proposition is larger than another? There is no immediate way of making a relevant sense of that idea, but it does not matter so much. Computations can anyway not be defined over contents, only over symbols. What we can and must do, then, is to measure the size of *representations* of content. We shall need a formal language where we give *canonical* representations of conceptual contents. With such a formal, unambiguous language of canonical representations, we can again count the number of symbols in its expressions for determining the relevant size of contents represented.

The final question concerns the nature of the computation steps that are to be counted. As mentioned above, standard time complexity takes the number of operations of Turing machines as the measure. If that were the choice, we would have to settle for some particular *kind* of Turing machine, whether a standard single-tape machine with a tape that is infinite in both directions, or something else. There is no uniquely right choice, and no absolute measure. Turing machine operations will involve steps needed in order to find the relevant information (on other tape squares) and moving symbols in order to make room for others etc, and how many such operations are needed will depend on the choice of machine. Therefore, these operations are to some extent arbitrary, and to that extent less essential to the complexity measure.

There is a natural alternative, which is to employ the *equation system* used for defining a function also as a method for computing the function. Take as a simple example, Donald Davidson's *Annette* (Davidson 1967, 17-18):

(2) i) Ref('Annette') = Annette.

probable will in the long run occur more often, and should be reported by means of shorter codes. So the efficiency of an encoding depends on the matching between the distribution of lengths of codes and the distribution of probabilities, over the same possible states of affairs.

In the present case, the questions of truth or falsity of sentences used or the probabilities of facts reported on, do not arise. We are only concerned with the expressive power and the efficiency of the interpretation. In the information theoretic case, questions of efficient encoding arise even if there is only a small finite number of signal types (sentences) used over and over. In the present context, having only a finite number of sentences would reduce the interpretation problem to triviality, since then the meaning of all sentences could be given by a finite list. This would reduce the total number of processing steps needed for any sentence to exactly 1.

$$\text{ii) } \text{Ref}(\text{'the father of'} \frown t) = \text{the father of Ref}(t).$$

This simple definition provides a method for deriving the interpretation of ‘the father of the father of the father of Annette’ in four steps of substitution. Let ‘ F ’ be the object language father operator and ‘ $\mathbf{F}$ ’ its analogue in the meta-language, and let ‘ a ’ be the object language name of Annette. Then we have in four steps with the semantic function μ_a :

$$\begin{aligned} (3) \quad & \mu_a(F(F(F(a)))) \\ &= \mathbf{F}(\mu_a(F(F(a)))) \\ &= \mathbf{F}(\mathbf{F}(\mu_a(F(a)))) \\ &= \mathbf{F}(\mathbf{F}(\mathbf{F}(\mu_a(a)))) \\ &= \mathbf{F}(\mathbf{F}(\mathbf{F}(\text{Annette}))) \end{aligned}$$

where (what corresponds to) the second clause of (2) is applied three times and the first clause once.

Each derivation step in (3) is a substitution step. Each substitution is performed in accordance with equations in (2). These equations are applied only for substitution from left to right: an instance of the left-hand side is replaced by the corresponding instance of the right-hand side. This makes the system into a so-called *term rewriting system*. Term rewriting systems are sets of *rewrite rules*. Rewrite rules apply to *terms* and license formal *substitutions* of/in those terms. Rewrite rules can contain variables, in which case an *instance* of the left-hand-side is allowed to be transformed to the corresponding instance of the right-hand-side. Clause ii) of (2) can be regarded as such a rule, with the variable t occurring once on each side. Relative to some rewriting system R , when no rule of R applies to a term u , u is said to be in *normal form*. The little derivation in (3) transforms the initial term ‘ $\mu_a(F(F(F(a))))$ ’ to its normal form ‘ $\mathbf{F}(\mathbf{F}(\mathbf{F}(\text{Annette})))$ ’ in four steps.

Transforming terms to normal form by means of a sequence of rewrite rule applications is a completely general form of computation. It has been shown that any both-way infinite one-tape Turing machine can be simulated by a term rewriting system such that each rule of the rewriting system corresponds to a machine

transition and each machine transition is represented by at least one rewrite rule (cf. Baader and Nipkow 1998, 94-97). In virtue of this relation it is not only very convenient but also well motivated to use the count of rewrite rule applications as a measure of time complexity.

Then, for each non-normal rewriting term s , we consider the shortest derivation by which s is normalized. Only normal terms correspond to full interpretation, i.e. to our representations of the world; other terms only have a role in deriving the normal terms. Let *input terms* of the rewriting system be terms of the form ' $\mu(t)$ ', with t a syntactic term. For a *normal* term s we consider the *shortest* derivation by which some input term $\mu(t)$ is normalized to s .¹⁴ Let that be the *term complexity* $Ct_R(s)$ of s relative to R . The time complexity $C_R(k)$ for the size k relative to the system R is then the *maximal* $Ct_R(s)$ such that s has size k .

With this much of background, I turn to characterizing the main ideas about minimizing time complexity.

7. Minimizing complexity

In this section and the next I shall very briefly sketch the ideas and the reasoning that lead up to the results about compositionality and complexity. Because of space limitations, the presentation must be largely informal, and full precision is not possible here.¹⁵

In general, a rewriting system R is a set of rewrite rules of the form

$$F(\vec{x}) \rightarrow G(\vec{y})$$

(where the arrows over the variables indicate that it is a sequence of variables).¹⁶

An example would be

$$h(x_1)bx_2 \rightarrow g(x_1, c)bd$$

where ' b ', ' c ' and ' d ' are constants. A derivation is a sequence of applications

¹⁴There need not be any longest derivation, since it is possible that there is no upper bound to the size of terms that reduce to the same normal form.

¹⁵It is formally developed in Pagin 2008*b*.

¹⁶For an excellent introduction to term rewriting, see Baader and Nipkow 1998.

where a term (often a subterm of a larger term) that is an instance of the lhs (left-hand-side) is replaced (in the containing term, in case it is a subterm) by the corresponding instance of the rhs. An instance of term s is any term s' resulting from s by uniform substitution by terms for rewrite variables. Thus, ' $h(s_7)bf(s_9)$ ' is an instance of the lhs above.

A rewriting system R is said to *terminate* iff every derivation eventually leads to a term in normal form (to which no rule applies). R is said to be *confluent* iff it holds for any terms s_1, s_2, s_3 such that s_2 and s_3 both can be derived from s_1 , that there is a term s_4 such that s_4 can be derived from both s_2 and s_3 . R is *convergent* iff R both terminates and is confluent.

Not all term rewriting systems terminate and not all are confluent, and neither property is in general decidable. However, the systems we are concerned with, that satisfy the format of primitive recursion equations, involve substitutions of a very restricted kind. It is straightforward to show that these systems are convergent.

Furthermore, it can be shown that every derivation terminates with a canonical term. Given that the set of canonical terms is the fragment of the formal language that represents the conceptual domain, every derivation ends with a term representing the conceptual domain. This means that every such rewriting system, when the rules are stated as equations, has the format for defining a semantic function.

We can call these systems ' μ systems'. A μ system R for a language L is then such that for every (meaningful) grammatical term t of L , there is a rule $r \in R$ such that the rewrite term ' $\mu(t)$ ' instantiates the lhs of r . And a grammatical term occurs in an instance of the lhs of a rule only as a subterm of a larger rewrite term, since it can only occur as an argument to a function.

μ systems have four properties that are crucial for complexity. A fifth property is peculiar to *direct* μ systems:

- (RS) a. Every μ system has a finite number of rules.
- b. No term in the formal language of the μ system (as opposed to terms occurring in the rule formulations) contains any rewrite variables.
- c. For any rule r of the μ system, the set of rewrite variables on the rhs of

- r is a subset of the set of rewrite variables occurring on the lhs of r .
- d. In a μ system R , no rewrite term containing a terminal symbol instantiates the lhs of any rule $r' \in R$.
 - e. Every rule of a *direct* μ system is an output rule. In an output rule a terminal symbol occurs on the rhs, and any symbol on the rhs that does not occur on the lhs is a terminal symbol.

Properties (RSc) and (RSd) have the combined effect that terminal symbols cannot be produced by means of instantiating variables. Any terminal symbol occurring in a term is produced by means of a rule where it is explicitly used on the rhs. At most finitely many terminal symbols can occur on the rhs of any rule. Since the rule system is finite, there is a largest number w of terminal symbol occurrences that can be produced in any single rule application. This is the MaxApp number of the system. It is immediate that the smallest number of application steps needed produce a term s of size k in a μ system R is $\lceil k/\omega \rceil$, where ω is the MaxApp number of R and $\lceil z \rceil$ is the smallest whole number at least as great as z . Since there are infinitely many terms of normal form, the ratio $\lceil k/\omega \rceil$ will be an upper limit of efficiency in the long run. Hence, no μ rule system R can be faster than having a linear time complexity function C_R .

The real efficiency may be much lower. If rules that produce new non-terminal symbols are present, the upper limit of efficiency may be an exponential function of the size of the canonical terms. If the rule system is *direct*, every new symbol on the rhs of a rule is a terminal symbol. In virtue of property (RSd) of μ systems, that term cannot itself be an argument, i.e. instantiate the lhs of a rule. Only its proper subterms can. In μ systems, substitutions are only performed on subterms that do not contain terminal symbols. Because of this, μ systems that are direct are guaranteed to transform terms to normal form in an incremental fashion, in each step replacing non-terminal by terminal symbols, until only terminal symbols remain.

Let the MinApp of a μ system be the minimal number of terminal symbol occurrences that are produced by any single rule application. Hence, for a direct system R , $\text{MinApp}(R) \geq 1$. This means that for a direct rule system R , we can estimate

the complexity function C_R as

$$\lceil k/\omega \rceil \leq C_R(k) \leq \lceil k/\alpha \rceil$$

where ω is $\text{MaxApp}(R)$ and α is $\text{MinApp}(R)$. Since $\alpha \geq 1$, it follows that $C_R(k) \leq k$ if R is a direct rule system. I shall say that systems with such a complexity function are *maximally time efficient*.

Clearly, since there is no finite upper bound the value of ω , there is no highest efficiency value. It still makes sense to speak of maximal time efficiency, for the reason that rewriting computation can be sped-up by more than any finite factor. Where we have a system R_μ that computes a function μ we can devise a system R'_μ that computes the same function μ at roughly twice the speed. We do this by creating more complex rules, i.e. rules that apply to larger terms. Such rules are more specialized, and hence more such rules are needed for having an equivalent system.¹⁷

We have already seen that a direct rule system is maximally time efficient. Because of the possibility of speed-up, the simple converse, that a system with maximal time efficiency is direct, doesn't hold. We can, however, define a notion *maximal efficiency simpliciter* in a way that excludes ad hoc speed-up features. This allows proving the following claim:

(ED) If R is maximally efficient, then R is direct.

We have connected efficiency with directness. The second major step will be to connect the directness of a rule system with the nature of the semantic functions that be defined by the corresponding equation systems.

8. Compositionality and minimal complexity

In order to see how minimal complexity connects with the question of compositionality, we must inquire about the *form* of maximally efficient systems. It can be

¹⁷ This corresponds to speed-up transformations of Turing machines. For a given Turing machine M we can e.g. devise a machine M' that is twice as fast by letting the new one process two tape squares at a time (cf. Hartmanis and Stearns 1965).

shown that

(DH) If R is a maximally efficient rule system, and r is a rule of R with a lhs of the form $\mu(\sigma(x_1, \dots, x_n))$, then the rhs has the form $\rho(\mu(x_1), \dots, \mu(x_n))$.

Proof. Suppose for *reductio* that R is a maximally efficient μ system that contains a rule r of the form

$$(4) \quad \mu(\sigma(t_1, \dots, t_n)) \rightarrow p(t_1, \dots, t_n, \mu(t_1), \dots, \mu(t_n))$$

where p is simple or complex. Suppose we have an instance q of the rhs of r . Since p is a simple or complex *new* symbol and R is maximally efficient, and therefore by (ED) direct, by (RSe) p can only consist of terminal symbols. Hence, by (RSd), neither q , nor any subterm q' of q that contains any constituent of p , instantiates the lhs of any rule r' of R . Only the $\mu(t_i)$ or the t_i do. But μ systems don't have rules where the grammatical terms themselves instantiate the lhs. And since the t_i are not themselves terminal symbols, q contains non-terminal symbols that cannot be eliminated, which is impossible.¹⁸ $\square$

Instead, the rules for complex terms in a maximally efficient μ system must be of the form

$$(5) \quad \mu(\sigma(t_1, \dots, t_n)) \rightarrow p(\mu(t_1), \dots, \mu(t_n))$$

This means that the corresponding equation format for defining a semantic function is

$$(6) \quad \mu(\sigma(t_1, \dots, t_n)) = p(\mu(t_1), \dots, \mu(t_n))$$

which conforms to the homomorphism format of compositionality.

The format is, however, stricter than what compositionality requires, since p is required to be a simple or complex *terminal* (operator) symbol. Hence, either p is a

¹⁸In order to keep the proof simple, it must here be required that atomic terms are distinct from the corresponding atomic expressions. Otherwise quotation will produce examples where some atomic grammatical terms in fact are terminal symbols, but this is an inessential complication.

simple operator or else a *polynomial term* $g(h_1(\vec{x}), \dots, h_n(\vec{x}))$ over simple operators and (rewrite) variables. If we look at it as a specification of a semantic function, this means that either p denotes a simple function on the conceptual domain which therefore belongs to the set Γ of basic meaning composition functions, or else it denotes a function definable from basic meaning composition functions by means of function composition, i.e. the schema (FC) Let's say that semantic functions that comply with this restriction are *polynomially compositional*.¹⁹

If a semantic function is polynomially compositional, it is specifiable by a direct rule system, and hence by a maximally time efficient rule system. To the extent we can speak of a semantic function itself as having a degree of complexity, we may say that such a semantic function has minimal time complexity. It does not hold for compositional semantics in general, only for the polynomial kind.

The converse does not follow immediately, however.²⁰ The most important reason is that we can have *other functions* from terms to values that behave well with respect to computational complexity. If we add a *quotation* operator to a language L , we will make another function relevant: the function V from terms to the *expressions* of L . Somewhat oversimplified, with κ as quotation operator, we have

$$(7) \quad \mu(\kappa(t)) = 'V(t)'$$

(this is a simplified formulation; assume here that substitutions within quotes are allowed). For familiar reasons, clause (7) induces non-compositionality: $V(t) \neq V(u)$ even if $\mu(t) = \mu(u)$. However, this need have no adverse effect on time complexity. If the V function itself is a homomorphism, so that for every operator σ there is some simple or polynomial expression operator ϵ such that for any terms

¹⁹ Interestingly, this is the intuitively simplest and most natural version of compositionality. It imposes a strong similarity between the syntactic and conceptual algebras. Theo Janssen (1986, 1997) has insisted on the restriction to polynomially defined algebras for the intermediate logical language, in cases when the semantics is defined indirectly, via translation into an intermediary logical language, although for quite different reasons. The restriction has been criticized by Herman Hendriks (2001) as not essential to compositionality and as imposing conditions that are sometimes impossible to meet. The present considerations agree with Hendriks in principle, but also provide separate reasons why approximating polynomial compositionality is nevertheless desirable.

²⁰ One reason for this that I shall not elaborate on is that it follows only if the meta-language itself has a well-behaved semantics.

$t_1, \dots, t_n$ for which σ is defined it holds that

$$(8) \quad V(\sigma(t_1, \dots, t_n)) = \epsilon(V(t_1), \dots, V(t_n))$$

then semantic processing can continue with the same time efficiency as before. To accommodate a quotation operator, we would need to add expressions to the conceptual domain, so that they can serve as values to the semantic function, but this is not in itself a problem.

In fact, the situation generalizes. Let's say that a set Δ of functions from the syntactic domain T to the (enriched) conceptual domain M is *jointly-compositional* just in case, for any $\delta \in \Delta$ and any operator σ that there is a meaning composition function ρ and functions $\delta_1, \dots, \delta_n \in \Delta$, not necessarily distinct, such that for all terms $t_1, \dots, t_n$ (with δ is defined for $\sigma(t_1, \dots, t_n)$) it holds that

$$(9) \quad \delta(\sigma(t_1, \dots, t_n)) = \rho(\delta_1(t_1), \dots, \delta_n(t_n))$$

If in each clause of this kind in the semantic system ρ is polynomial, time complexity will still be minimal, since the relevant $\delta_i \in \Delta$ are read off from the clause, and need not be further computed. The system is not compositional in the normal sense. I'll refer to this as *generalized compositionality*. It retains the two features that are crucial for time complexity: meaning composition is polynomial and only δ values for immediate subterms matter. The limiting case is standard polynomial compositionality, where there is only one δ_i , the main semantic function μ itself. Strictly speaking, then, what is motivated from complexity considerations is not standard compositionality, but this generalized form.²¹

The general trend is clear: a drive to minimizing complexity is a drive to compositionality, more particularly to polynomial compositionality, and less particularly to generalized polynomial compositionality. Under certain restrictive conditions, compositionality is entailed, but these conditions tend not to be met in natural language. There are reasons to suspect that syntactic complications and

²¹Alternatively, this can be presented as a system with one semantic function $\mu(t, c)$ that takes contextual arguments, but that leads to a computational increase. Cf. Pagin and Westerståhl 2008.

Generalized compositionality is suitable also for modal contexts (cf. Glüer and Pagin 2006, Glüer and Pagin 2008), propositional attitude contexts and mixed quotation contexts (Pagin 2008a).

widespread context dependence make (generalized) polynomial compositionality impossible.²² Hence, in the end, no strict argument for compositionality is forthcoming.

Cognitively, generalized polynomial compositionality allows semantic processing, both in interpretation and in production, to proceed in what is intuitively the simplest possible way: by mere association of atomic terms with atomic concepts, and syntactic modes of composition with conceptual modes of composition. This in turn, allows the speaker/hearer to efficiently process the input on-line, i.e. to incrementally construct the output while the input is being observed (in the case of the hearer) or itself constructed (in the case of the speaker). For the speaker, incremental production allows articulation of one part of a Thought to take place in advance of detailed plans about later parts. For the hearer, incremental interpretation allows the semantics to bear on the parsing process, thereby reducing underdetermination in the transition from string to structure (cf. e.g. Mahesh, Eiselt, and Holbrook 1999).²³ What we have a reason to believe, therefore, is that actual semantics of natural languages nonetheless *approximate* the computationally optimal format, since it helps explaining our communicative capacities.

Department of Philosophy
Stockholm University

References

- Baader, F. and Nipkow, T. (1998): *Term Rewriting and All That*, Cambridge: Cambridge University Press.
- Boolos, G. S., Jeffrey, R. C., and Burgess, J. P. (2002): *Computability and Logic*, 4th ed., Cambridge: Cambridge University Press.

²²This holds e.g. in the architecture of the Principles and Parameters system: here a syntactic phrase structure term is mapped on a syntactic LF term before semantic interpretation can take place. For objections to such an architecture, see e.g. Jacobson 2002 and the conception of so-called *direct compositionality*. This is related to but distinct from what is here called *polynomial compositionality* and differently motivated.

²³In both cases, preliminary decision are subject to later corrections, as studied especially as regards parsing, with garden-path sentences.

- Davidson, D. (1965): 'Theories of Meaning and Learnable Languages', in Y. Bar-Hillel (ed.), *Logic, Methodology and Philosophy of Science II*, Amsterdam: North-Holland. Reprinted in Davidson 1984, 1-15. Page references to the reprint.
- (1967): 'Truth and Meaning', *Synthese* 17: 304–23. Reprinted in Davidson 1984, 17-36. Page references to the reprint.
- (1984): *Inquiries into Truth and Interpretation*, Oxford: Clarendon Press.
- Frege, G. (1923): 'Gedankengefüge', *Beiträge zur Philosophie des Deutschen Idealismus*. 3: 36–51. Reprinted in Frege, *Logische Untersuchungen*, Vandenhoeck & Ruprecht, Göttingen, 1976. Translated by R. Stoothoff as 'Compound thoughts', published in *Mind* 72 (1963), 1–17.
- Garey, M. R. and Johnson, D. S. (1979): *Computers and Intractability. A Guide to the Theory of NP-Completeness*, New York: W. H. Freeman, Company.
- Gibson, E. (1998): 'Linguistic Complexity: Locality of Syntactic Dependencies', *Cognition* 68: 1–76.
- Glüer, K. and Pagin, P. (2006): 'Proper names and relational modality', *Linguistics & Philosophy*: 507–35.
- (2008): 'Relational Modality', *Journal of Logic, Language and Information*.
- Hartmanis, J and Stearns, R. E (1965): 'On the Computational Complexity of Algorithms', *Transactions of the American Mathematical Society* 117: 285–306.
- Hawkins, J. A. (2003): 'Efficiency and Complexity in Grammars: Three General Principles', in J. Moore and M. Polinsky (eds.), *The Nature of Explanation in Linguistic Theory*, Stanford, Cal.: CSLI Publications, 121–52.
- Hendriks, H. (2001): 'Compositionality and Model-Theoretic Interpretation', *Journal of Logic, Language and Information* 10: 29–48.
- Hodges, W. (2001): 'Formal Features of Compositionality', *Journal of Logic, Language and Information* 10: 7–28.
- Jacobson, P. (2002): 'The (Dis)organisation of the Grammar: 25 years', *Linguistics & Philosophy* 25: 601–26.

- Janssen, T. (1986): *Foundations and Applications of Montague Grammar. Part 1, Philosophy, Framework, Computer Science*, CWI Tract 19, Amsterdam: Centre for Mathematics, Computer Science.
- (1997): ‘Compositionality’, in J. van Benthem and A. ter Meulen (eds.), *Handbook of Logic and Language*, Amsterdam: Elsevier, 417–73.
- Li, M. and Vitányi (1997): *An Introduction to Kolmogorov Complexity and its Applications*, 2nd ed., New York: Springer.
- Mahesh, K., Eiselt, K. P., and Holbrook, J. K. (1999): ‘Sentence Processing in Understanding: Interaction and Integration of Knowledge Sources’, in A. Ram and K. Moorman (eds.), *Understanding Language Understanding*, Cambridge, Mass.: MIT Press.
- Pagin, P. (2003): ‘Communication and strong compositionality’, *Journal of Philosophical Logic* 32: 287–322.
- Pagin, P. (2008a), ‘Belief Sentences and Quotation with Generalized Compositionality’. Presentation at the Logos Seminar, Barcelona.
- Pagin, P. (2008b), ‘Compositionality, Computability, and Complexity’. Draft. Earlier version presented at the Compositionality Series of lectures, Rutgers University, 2007.
- Pagin, P. and Westerståhl, D. (2008): ‘Compositionality’, in C. Maienborn, K. von Stechow, and P. Portner (eds.), *Semantics. An International Handbook of Natural Language Meaning*, Elsevier.
- Pelletier, F. J. (1994): ‘The Principle of Semantic Compositionality’, *Topoi* 13: 11–24. Expanded version reprinted in S. Davis and B. Gillon (eds.) *Semantics: A Reader* (2004), pp. 133–157.
- Potts, C. (2007): ‘The Dimensions of Quotation’, in C. Barker and P. Jacobson (eds.), *Direct Compositionality*, Oxford: Oxford University Press, 405–31.
- Shannon, C. E. (1949): ‘The Mathematical Theory of Communication’, in C. E. Shannon and W. Weaver (eds.), *The Mathematical Theory of Communication*, The University of Illinois Press.

- Werning, M. (2005): 'Right and Wrong Reasons for Compositionality', in M. Werning, E. Machery, and G. Schurz (eds.), *The Compositionality of Concepts and Meanings: Foundational Issues*, Frankfurt/Lancaster: Ontos, 285–309.
- Westerståhl, D. (2004): 'On the compositional extension problem', *Journal of Philosophical Logic* 35: 549–82.