Compositionality, Complexity, and Evolution

Peter Pagin

Department of Philosophy, Stockholm University

peter.pagin@philosophy.su.se

Introduction

It is widely believed that human cognition has evolved together with human language. It is also widely believed that the compositionality of natural language is one of its central features. The question then naturally arises why, and how, language evolved into a compositional means of communication.

Compositionality allows the body of semantic knowledge to be considerably smaller than the piecemeal knowledge of the language, i.e. independent knowledge of the meaning of each of the sentences of the language. Having a language with a greater expressive power seems like a good thing for a population, and maybe that could be enough for answering the question. It isn't, however.

One thing that can be asked further is by what evolutionary mechanisms a compositional language evolved. We don't believe that there was a sudden leap from an un-systematic to a compositional language, and we may ask what principle may lead, step by step, from the former to the latter[9]. In the section "The complexity of learning" I'll review an attempt to answer this question by appeal to the cognitive difficulty and computational complexity of *language learning*.

In the section "The complexity of interpretation" I shall present a different aspect of computational complexity as applied to the cognition of language, this time concerning the complexity of interpretation. I shall spell out the relation to compositionality, and end with a suggestion of how this connects with questions of evolution.

In the next section, I shall give a brief introduction to the idea of compositionality.

Of course, this is not convincing, since a small change in a rule system, or in a hard wiring, can create recursion, or feedback loops.

Compositionality

There are some remarkable facts about human linguistic communication: language users manage to successfully communicate new thought contents (contents not thought before) by means of new sentences (not used or considered before). The first part, that *new contents* are communicated, is a phenomenon that could happen simply because of context dependence: if 'kraa' means *food here now*, then every new utterance of 'kraa' means something different, even though the sophistication of speaker or hearer need

[9] In a public talk in Paris, 29 May 2010, entitled 'Poverty of Stimulus: Some Unfinished Business', Noam Chomsky commented on the evolution of language literature, claiming that it does not concern the evolution of language. This is because it suffers from the prejudice that evolution must take place in small steps. However, Chomsky remarked, the step from a finite to an infinite linguistic capacity is anything but small. So it cannot have been a step of evolution, as long as that is understood according to current theories.

not exceed that of seagulls. However, humans, unlike seagulls, appear to be able to communicate new contents, from an open-ended set, even if context is held fixed[10]. When context is held fixed (at least in relevant respects), if a different content is communicated, this is because a different sentence has been used[11]. So the human hearer H appears to have the ability to associate a new sentence uttered by the speaker S with a (probably new) *content* that is the *same* as, or at least sufficiently similar to, the content which S wanted to communicate, and the speaker S appears to have the ability to *select* a suitable sentence s that makes the task of H feasible. The speaker therefore has an *articulation* (production) capacity that interlocks with the *interpretation* (comprehension) capacity of the hearer. Since mind-reading is out of the question, and chance would deliver a microscopic rate of communicative success, given the sizes of the domains of contents and sentences, there must be something about language itself that makes this possible[12].

The interpretation part of this coordination task has received much more attention than articulation part. The hearer's task is made possible if there is a systematic correlation between the build-up of a sentence and its meaning. This is usually stated as the principle of *compositionality*:

(PC) The meaning of a complex expression is a function of the meanings of its parts and its mode of composition.

We shall say that the *semantics* of language is compositional or not, intending the assignment of meaning by some particular semantic function or semantic theory. In a derivative sense, we can say that *meaning property* of a particular language is compositional, if the *correct* semantics for it is compositional in the first sense.

The picture that the appeal to (PC) gives of the hearer's capacity is this: The hearer knows the meaning of the simple expressions of the language (lexical items and morphological units), which are finitely many. The hearer also knows the semantic significance of the (morpho-) syntactic modes of combination, which are finitely many. Putting knowledge of these two kinds together, the hearer can work out, step by step, the meaning of complex expressions. He starts with working out the meaning of those complex expressions that have simple parts, and then using this knowledge he can go on with combinations that have these complexes as parts, and so on. In interpreting (1a), given the syntactic structure (phrase structure) of (1b),

(1) a. John saw the dog

b. [S John_{NP} [$_{VP}$ saw_{VT} [$_{NP}$ the_{Det} dog_{N}]]]

the hearer knows the meaning of 'the', 'dog', 'John', and 'saw', and the semantic significance of a) the Determiner + Noun (Det+N) combination, b) the Verb (transitive) + Noun Phrasecombination(VT+NP),and c) theNounPhrase + Verb Phrasecombination(NP+VP). He starts with working out the meaning of 'the dog', using a), goes on working out the meaning of 'saw the dog', using b), and finishes by working out the meaning of (1a) from there, using c).

Although (PC) is a standard formulation, what is meant by 'part', as the principle is normally understood, is immediate part (immediate constituent). The principle is often incorrectly stated as saying that the meaning of the complex expression is a function of the meanings of the simple parts and the way they are combined. But this principle is degenerate. The mode of combination of the simple parts in (1) is in itself a combination of Det+N, VT+NP, and NP+VP, and the speaker would have to know the semantic

[10] Of course, this is inferred; because of the uniqueness of contexts, it cannot be directly tested.

[11] It may also be that the same sentence has been used, but under a different syntactic analysis. More about this below.

[12] Indeed, this phenomenon has (on and off) motivated reflection on language during a hundred years. Cf. Frege 1980; Frege 1923, opening passage.

significance of that complex mode. In fact, there are denumerably many ways of combining simple parts, and so if the hearer needs to know the semantic significance of each, the explanation of how he can understand new sentences cannot any more appeal to a working out strategy based on *finite* knowledge. Moreover, on the alternative formulation, the hearer who knows that the father of Annette is also the father of Isabel is not licensed to infer that the sentences

(2) a. The father of Annette sleeps

b. The father of Isabel sleeps

have the same truth value, for he is not licensed by the appeal to only simple parts of making use of that fact that 'the father of Annette' and 'the father of Isabel' are coreferring noun phrases.

If instead of requiring that the meaning of the complex is determined by the immediate parts and *their* combination, we require that it be determined by the immediate parts and their immediate parts, and the mode of combination of these two syntactic layers, we get a principle that is strictly weaker than (PC), but not degenerate. In fact, it has been argued that some particular constructions in English obey only this weaker principle[13].

A principle closely related to (PC) is

(PS) If in a complex expression A a constituent e is replaced by a constituent *e*' that has *the same* meaning as *e*, then the meaning of the new expression *A*'= *A*[*e*'/*e*] has the same meaning as *A*.

(where '*A*[*e*'/*e*]' is designates the expression the results from substituting e' for e in one or more occurrences). The intersubstituted constituents need not be immediate. In fact, if the semantics is total, so that each grammatical expression is meaningful, then (PC) and (PS) are *equivalent*.

If, by contrast, the semantics is *partial*, the two principles can come apart. On the one hand, if not all parts of a complex expression are meaningful, then the meaning of the complex is trivially not a function of the meanings of the parts ((PC) does not hold), but it can still hold that when two parts with the same meaning (i.e. meaningful parts) are intersubstituted, the meaning of the complex is preserved ((PS) holds). On the other hand, it may also be that *A* is meaningful, and that e and e' mean the same, yet *A*[*e*'/*e*] is *not* meaningful ((PS) does not hold), while it is still true that the meaning of every complex expression that *has* meaning is function of the meaning of its parts and their mode of combination ((PC) holds).

These observations are due to Wilfrid Hodges (2001). He calls the principle that the parts of every meaningful expression are meaningful *the domain principle*. He also calls the principle that if two expressions mean the same, then substituting the one for the other does not lead to loss of meaning in the complex *the husserl principle* (and the semantics *husserlian*). He also proved that given that the domain principle and the husserl principles hold, (PC) and (PS) are again equivalent, even if the semantics is partial.

Using the equivalence, we can see that (PC) in several respects quite weak. To get a *counterexample* to (PS), and thereby to (PC), we need two expressions *e* and *e*' with the *same* meaning, and two complex expressions, *A* and *A*[*e*'/*e*] with *different* meanings. If there is no counterexample, (PS) and hence also (PC) hold. Now, if no two different expressions in the language in question have the *same* meaning (meaning is *hyperdistinct*), then no counterexample is possible, and hence the semantics is vacuously compositional. Similarly, if no two expressions in the language *differ* in meaning, again the semantics is trivially compositional.

The first of these two observations, that hyperdistinctness entails compositionality, is somewhat damaging for the explanatory power of compositionality. Meaning for complex expressions of a language may obey

[13] According to Peters and Westerståhl (2006, chapter 6), this holds for the semantics of possessive constructions of the form NP's.

no regularity whatsoever, so that it is impossible to *work out* the meaning of a new complex expression, and yet because of hyperdistinctness, its meaning is compositional. So compositionality alone does not explain how speakers can work out the meaning of new complex expressions.

To explain the ability of hearers, the semantics therefore needs some additional properties. In particular, the semantic function that assigns meanings to expressions must be *computable*. This raises the following problem: If the semantic is computable, why should it also be compositional? This is a problem, for, as we shall see in "The complexity of interpretation", a language may have a computable semantics that is not compositional.

So far we have been talking as if semantic values or properties are assigned to *expressions*, where expressions are understood as *surface* forms, i.e. types of spoken utterance, types of written inscriptions, or types of other concrete marks or events. Surface forms are, however, often syntactically ambiguous. Two different words may have the same surface form (homonymy), and complex expressions may on top be ambiguous with respect to scope relations (witness Quine's 'pretty little girls' camp'). Since different disambiguations often yield different meanings, we must define the semantic function for *disambiguated expressions*, such as analysis trees, or *grammatical terms*. For setting out compositionality formally, we therefore need on the one hand a syntactic framework, which specifies disambiguated expressions of the language, and the semantic function and properties defined over these. We will also need a domain of semantic values and domains of other entities that can enter as arguments to the semantic function. In the modern tradition, there are two approaches to this. In the tradition from Montague[14], the syntax is built up from categories of expressions and corresponding sets of expressions that are of these categories, and the syntactic rules contain specifications of which categories their arguments and values have. The syntax then has the form of an *algebra*, where the grammatical rules are implemented as operations from *n*-tuples of expressions (of the right categories) to expressions. Correspondingly, the semantic values are organized into types of entities, and there is a *meaning* algebra, where the entities are possible semantic values, and operations in this algebra map *n*-tuples of entities (of the right types) on entities. There is a mapping between the syntactic categories and the meaning types. The semantic function, finally, is a *homomorphism* from the syntactic algebra to the meaning algebra. The function must respect the mapping between categories and types[15].

In the more recent tradition from Hodges (2001), there is no appeal to categories and types. Instead, syntactic operations are taken to be *partial*; they are defined only for certain arguments (combinations of arguments). There is a syntactic algebra, but there is no assumption that there is a meaning algebra, only a domain of meanings where no prior structure is required[16]. We shall here follow Hodges, with some modifications. The syntax for a language L is a triple (G_L, E_L, V_L), where G_L is a *grammatical term algebra*, EL is the set of *expressions* of L, and V_L is a mapping from grammatical terms to expressions (for convenience I shall henceforth drop the subscript). G itself is a triple (T, Σ, A), where T is the set of *grammatical terms*, Σ the (finite) set of *operations* that map *n*-tuples of grammatical terms on grammatical terms, and A is a (finite) set of *atomic* grammatical terms. T is the closure of A under Σ, i.e. the set of terms that are generated from A by means of (possibly repeated) applications of the operations in Σ.

[14] Cf. Montague 1970; Montague 1973, Janssen 1986; Janssen 1997, Hendriks 2001.

[15] 7The rest of this section is somewhat technical and may be skipped by readers who only want the main ideas.

[16] 8If we require that the semantics be *recursive*, then an inductive structure in the meaning domain is needed. Cf. section on "The complexity of interpretation".

To exemplify, let A be the set {'John','saw','the','dog'} and Σ the set $\{\alpha, \beta, \gamma\}$, corresponding to the rules for forming Det+NP, VT+NP, and NP+VP, respectively, of example (1). Then the grammatical term that corresponds to the phrase structure analysis (1b) is

(3) γ('John', β('saw', α('the','dog')))[17].

The V function compositionally defined on the grammatical terms. For each operation σ on terms, there is a corresponding operation σ on expressions such that

(V) $V(\sigma(t_1,..., t_m)) = \sigma(V(t_1),...,V(t_n)$

In the very simple case of our example, we just have

(4) $V(\sigma(t_1, t_2)) = V(t_1)␣V(t_2)$

where σ is α or β or γ, and '␣' marks a word boundary (space). We therefore get, after three applications of (V),

(5) V (γ('John', β('saw', α('the','dog')))) = 'John'␣'saw'␣'the'␣'dog'

which is a structural-descriptive name of 'John saw the dog'. No abstract limitations are imposed on V (such as that the value of the first argument should appear in the expression (surface string) to the left of the value of the second argument), although a number of restrictions will be empirically motivated, in order that the hearer be able to parse the string. The algebraic framework as such is very general; it can be adapted to phrase structure rules as well as to transformations or movements.

A *semantic* function μ is also defined on the grammatical terms, mapping terms on a domain M of semantic values. Given the algebraic framework we can now state a more formal version of (PC):

(PC') For every n-ary syntactic operation $\sigma \in \Sigma$ there is a function r_σ: $M^n \longrightarrow M$ such that for all grammatical terms $t_1,..., t_n$ such that $\alpha(t_1,..., t_n)$ is defined and μ meaningful, it holds that $\mu(\alpha(t_1,..., t_n)) = r_\sigma(\mu(t_1),..., \mu(t_n))$.

We call the function r_σ a *meaning operation*. If (PC') holds for a semantic function μ, we say that μ is compositional. Here μ is of course left unspecified. Compositionality is a general formal property that any particular semantic function, *given* a syntax, either has or lacks[18]. A statement that a semantics is compositional, in this standard sense, therefore is a $\forall\exists^1\forall$ statement, where the existential quantifier is second-order.

Historically, the first precise statement of (PS) was given by Gottlob Frege (1892), for *Bedeutung*. Clarity about (PC) developed slowly after that, and did not reach maturity until in the mid-late 1970s[19].

The complexity of learning

The question why evolution would tend to produce language with compositional semantics can be seen from a learning point of view. Learning, in turn, can be viewed both from a genetic and from a cultural perspective. The issues involved are surveyed in Smith and Kirby 2012. In the cultural perspective, the issue is that of *iterated learning*: A speaker B learns from a speaker A, a speaker C from speaker B, and

[17] This is an illustration. It might be preferable e.g. to have as top operation one that combines the term that corresponds to 'the dog'i with the term that corresponds to 'John saw i ', analogous to a Montagovian quantification rule.

[18] It is *another* matter that language theorists may want to adjust their *syntactic* theory for the sake of making their semantics compositional.

[19] For more on compositionality, see Pagin and Westerståhl 2010b; Pagin and Westerståhl 2010c; Pagin and Westerståhl 2010a, Janssen 1997, Szabó 2008.

so on. Without considering genetic change, we can inquire into the conditions under which such iterated learning tends to produce language change, in each step, that drifts towards a compositional semantics.

Here I shall focus on a particular proposal in Brighton 2005. Brighton, among others, emphasizes the role of the so-called *learning bottleneck*. By this is meant the fact that only a meagre subset of the whole language is actually transmitted to the learner, and the learner needs to develop a hypothesis about the underlying regularity in order to generalize to new cases. This of course is a counterpart to Chomsky's *poverty of the stimulus* argument. See e.g. Chomsky 1971. The point of that argument was that the sample of utterances the child gets is very small, and if there are no built-in restrictions on how to project the sample on to a grammar there are just too many possibilities compatible with the sample for a projection to take place at all. Hence, there must be built-in restriction, an innate universal grammar.

Brighton, and those working in the same tradition, reject this argument. In contrast to Chomsky, Brighton thinks that we can lay down general principles for learning that will generate the desired result, without the need to appeal any particular innate linguistic structures. As regards linguistic meaning, and in particular the emergence of compositional semantics, Brighton suggests how.

Brighton operates with simple abstract languages, strings over an alphabet (as is standard in formal language theory). As meanings he uses a feature-value space: the meaning of a meaningful string is a vector, consisting of one value for each feature (e.g. (2,2,2) in a space of three features each of which can take two values). The interpreted language is a set of pairs of strings and meanings. Such a language can be specified by a comprehensive list of string-meaning pairs, and the question is whether it can be specified in some more efficient way.

As the theory of automata and formal languages were developed in the 1950s, grammars that generate languages by means of production rules correlate with automata that *accept* languages. Typically, this concerned grammar. In the case of semantics, Brighton employs the idea of a special kind of finite-state machine, a so-called *transducer*. Basically, a transducer accepts certain string-meaning pairs and rejects others. For each such pair in the language the machine accepts there is a path through *states* of the machine, starting with the initial state and ending in the final, accepting, state. For each new letter of a meaningful string, as read from left to right, there is a possible branch along which the machine can move into a new state, one branch for every meaning compatible with the initial segment of the string that has been considered so far. After the final letter, the machine moves into the accepting state, and if the string happens not to have a meaning in the language, the machine just halts in a non-accepting state.

In the most elementary, non-efficient case, there is a 1−1 correlation between strings of the language and paths of the machine. But for some languages it is possible to *compress* the machine. For instance, if two strings start with the same letter and also are associated with the same initial feature-value, the initial segments of the two paths can be *merged* into a single *shared* segment of both paths. And in general, when a letter value part is shared between two string-meaning pairs, there is a corresponding merge to be made in the path. The machine gets compressed. The *description* of the machine becomes shorter.

In case the semantics of such a language is compositional, each letter of the alphabet corresponds to a particular value of some feature. In such a case, compression of the machine can go very far, resulting in a minimal transducer with one state for each string position / semantic feature, and between these states one branch (edge) for each value of the corresponding feature. Such a machine is minimal. In the opposite case, where no compression, or hardly any compression, is possible, the language is called "random" or "holistic"; such a language has to be learned piecemeal, one word at a time[20].

[20] This is not the customary use of the term 'holistic' in the philosophy of language. There, it typically indicates interdependence in meanings between different expressions, and therefore in this sense semantic holism is the opposite of semantic randomness. Cf Pagin 2006.

We can therefore see that there is a certain connection between compositionality and minimal *complexity*. Complexity can be measured with respect to different parameters and in different ways. Brighton appeals to *Kolmogorov* complexity (or *minimal description length complexity*) (cf. Li and Vitányi 1997): given some standard encoding (such as a particular universal Turing machine), complexity is measured in the number of bits required to describe certain data or functions. For instance, if, given a way of encoding Turing machines by means of some particular universal machine, a smaller Turing machine can be used for computing a function *f* than can be used for computing a function *g*, function *f* is deemed to have *lower* Kolmogorov complexity than *g*.

Brighton applies to a language learning situation where the learner gets certain linguistic data (a sample of string-meaning pairs) and forms a hypothesis about the data (in the form of a transducer). The complexity is then measured as a certain sum: the sum of the length of the description of the machine and the length of the description of the data by utilizing the machine. If the machine is compact, the complexity is low. Therefore, roughly, low complexity corresponds to compositionality.

This is then implemented in an evolutionary framework, i.e. here in iterated learning framework. The learning bottleneck has the effect that in each transition from teacher to learner, only a proper subset of the entire language can be actually exhibited, and the learner needs to form a hypothesis about the language that fits the data. Brighton studied what happens when it is *built in* to the evolutionary process that learners try to minimize complexity, i.e. try to find the most compact machine compatible with the data, and *also* generate *new* string-meaning pairs only if they are compatible with the hypothesis they have formed. It was shown in computer simulations that a process starting from an arbitrary language, after a number of learning generations, eventually leads to a stable state where the language approximately compositional.

This is an interesting result. Still, from the point of view of my own interests, it suffers from two interrelated drawbacks. On the one hand, compositionality is built into the evolutionary process in the sense that it governs the *hypothesis formation* of new learners, as well as their creative production of new string-meaning pairs (with random creative production, no compression results). That limits the value of the explanation. Especially, building in a compositional hypothesis formation function does not offer a very distinct alternative to Chomsky's innateness hypothesis.

On the other hand, compositionality is contrasted only with randomness ("holism"). No account is taken of semantics that can be given a compact description by means of some *recursive* semantic function that is still not compositional. This may be a theoretical oversight, but it is also connected with the simplified semantic model in terms of acceptability by transducers. Compression depends on the association of letters with values, and if two letters are associated with the same value, they will be interchangeable, in the strings of a fully compressed language, without changing the acceptability of the string-meaning pair. By contrast, if we were to use a Turing machine for accepting string-meaning pairs, a compact machine could accept a language with infinite string-meaning pairs but a non-compositional semantics. Then the question arises again: why would compositional semantics be desirable, over and above a computable semantics? I shall propose an answer to this question in the next section.

The complexity of interpretation

The function version of compositional semantics is given by recursion over syntax, but that does not imply that the meaning operations are defined by recursion over *meaning*, in which case we have *recursive semantics*. Standard semantic theories are typically both recursive and compositional, but the two notions are mutually independent. For a semantic function μ to be given by recursion it must hold that:

Rec(μ) There is a function b and for every $\alpha \in \Sigma$ an operation $r\alpha$ such that for every meaningful expression s,

$$\mu(s) = \begin{cases} b(s) & if\ s\ is\ atomic \\ r\alpha(\mu(u1), \ldots, \mu(un), u1, \ldots, un) & if\ s = \alpha(u1, \ldots, un) \end{cases}$$

For μ to be recursive, the basic function b and the meaning composition operation $r\alpha$ must themselves be recursive, but this is not required in the function version of compositionality. In the other direction, the presence of the terms $u1$,..., un themselves as arguments to $r\alpha$, has the effect that the compositional substitution laws need not hold. Substituting a subterm for another subterm with the same meaning may change the meaning of the mother term[21].

By a generalization of *Church's Thesis*, a semantics is computable if and only if it is recursive. Therefore, if we want semantics to be computable, we want it to be recursive. Again, is there any reason why it should be compositional as well?

One possible reason has to do with the efficiency, and thereby with the *complexity*, of interpretation. We can look at the task of interpretation, i.e. semantic processing, as a problem in the sense of complexity theory. Classical computational complexity theory (of which the theory of Kolmogorov complexity is one species) is concerned with giving mathematical measures of the difficulty of mathematical problems. The problem need not be a problem within any standard branch of mathematics, such as number theory or geometry, but must be a problem that can be adequately represented in a formal language, as an input to computation.

For measuring the complexity of a problem one needs a computation method. One then asks how much of resources is needed by this method for arriving at a solution to the problem. A standard method that is used as a reference device in this sense is that of one-tape Turing machines. One of the standard resources is time, in the sense of the number of computation steps needed by the Turing machine for arriving at the solution. This is so-called *time complexity*.

With a problem type and computation method is associated a *time complexity function C*. This is a function that takes as argument a measure of the size of a problem instance, as a numerical value, and gives as value the size of the *largest* computation that is needed to compute any problem of the same size. We can illustrate this with one of the most classical examples, the problem of the *Travelling Salesman*: a salesman is to visit a number k of cities exactly once and then return home, and the problem is to find a visiting order that minimizes the total distance travelled. In this case the solution consists in selecting the optimal order and verifying that it is optimal. The number of cities k is the *size* of the problem instance, and this is the argument to the complexity function C. Its value $C(k)$ is the number of computation steps needed *at most* for determining the solution for any problem instance of size k.

[21] This can happen e.g. with simple semantics for quotation, as noted e.g. in Werning 2005. Such a semantics is given by Christopher Potts (2007), incorrectly claiming that it is compositional. Cf. Pagin and Westerståhl 2010d

In complexity theory one is interested not so much in the value of *C* for a particular argument, but rather in how fast the value *C* (*k*) *grows* when the argument increases. If *C* (*k*) is bounded by a linear function of *k*, the time complexity is said to be linear; if it is bounded by k^n, for some natural number *n*, the time complexity is said to be *polynomial*, or equivalently that the problem is solvable in polynomial time.

Problems that are solvable in polynomial time are generally regarded as *tractable*, or *feasible*, while if the value of the complexity function grows faster, they are said to be *intractable* (this is known as the *Cobham-Edmonds thesis*). It is not known whether the travelling salesman problem is intractable in this sense. The reason is that no method is known for determining the solution (with certainty and for any finite *k*) that is more efficient than calculating the total travelling distance for each visiting order and selecting the shortest. Since the number of visiting orders for *k* cities is $k!$, the factorial of *k*, and since $k!$ grows faster than k^n, for any *n*, the general problem is intractable if there is no method sufficiently faster then checking all possible orders of travelling[22].

How does this apply to semantic interpretation? We need a method of computing meanings from disambiguated expressions as inputs. If we then think of semantics in functional terms, we want to compute a semantic function μ that takes as arguments disambiguated expressions — *grammatical terms* — and gives as values *meanings m* of some sort in this format:

$\mu(t) = m$.

Since meanings are non-syntactic abstract entities, they must be syntactically represented, i.e. by means of a sufficiently formal meta-language *ML*. That means that in an equation instance of this format, '*t* ' is replaced by an expression denoting a grammatical term, and '*m*' by an expression of *ML*.

Then we need an algorithmic method of some kind for computing meanings. A type of method that is particularly well suited is that of *term rewriting systems*. In general, a term rewriting system (a TRS) R is a pair (*F*, *R*) of a signature *F* and a set *R* of rewrite rules over that signature. The signature consists of a set of basic terms, and a set of operators. To this is added a set of *rewrite variables* which are used in stating the rules. A rewrite rule has the form

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

(where the arrows over the variables indicate that it is a sequence of variables)[23]. An example would be

$h(x_1)bx2 \rightarrow g(x_1, c)bd$

where '*b*', '*c*' and '*d*' are constants. Every rule application is a substitution operation, where an instance of the left-hand-side (lhs) of the rule is replaced by the corresponding instance of the right-hand-side (rhs) of the same rule. The substitution may be per' formed on a subterm of a larger term. An instance of a 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 derivation is a sequence of rule applications, where every step except the initial one is an application to a term that results from a previous step. In case a term is reached such that no rule of the TRS applies to it (and hence not to any of its subterms either), the derivation has *terminated*, and the term is said to be in *normal form*. The original term is then *reduced* to normal form. A rewrite system R terminates iff every derivation eventually leads to a term in normal form. R is said to be *confluent* iff it holds for any distinct 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.

[22] For interesting partial results concerning this problem, cf. the Wikipedia article http://en.wikipedia.org/wiki/Travelling_salesman_problem. The problem is known to be *NP hard*, which entails that if NP =P, as is generally believed, it is intractable.

[23] For an excellent introduction to term rewriting, see Baader and Nipkow 1998.

Rewriting systems are general computation devices, in the sense that the reduction of a rewrite term to normal form is a computation. It is a standard result that any Turing machine can be simulated by a term rewrite system (cf. Baader and Nipkow 1998, , 9497). We also get a very natural measure of time complexity by just counting the number of rule applications, i.e. reduction steps, until normal form is reached. One reason why term rewriting is a natural choice for semantic interpretation is that the clauses by means of which a semantic system is defined correspond closely to rewrite rules, and can be transformed into rules by a minimal change.

To illustrate, consider Davidson's *Annette* example (Davidson 1967, 17-18) of a compositional semantics:

(6) i) Ref('Annette') = Annette

ii) Ref('the father of' ⌢ *t*) = the father of Ref(*t*)

This simple definition has the form of a system of equations, and 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 '*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 μa:

(7) $\mu a\ (F\ (F\ (F\ (a)))) = \mathbf{F}(\mu a\ (F\ (F\ (a)))) = \mathbf{F}(\mathbf{F}(\mu a\ (F\ (a))))$

$= \mathbf{F}(\mathbf{F}(\mathbf{F}(\mu a\ (a))))$

$= \mathbf{F}(\mathbf{F}(\mathbf{F}(\text{Annette})))$

where (what corresponds to) the second clause of (6) is applied three times and the first clause once.

Each derivation step in (7) is a substitution step. Each substitution is performed in accordance with (what corresponds to) equations in (6). 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. We have then in fact used the system as a rewrite system. To make that explicit, replace the identity signs with left-right arrows:

(8) i) $\mu a\ (a) \rightarrow$ Annette

ii) $\mu a\ (F\ (x)) \rightarrow \mathbf{F}(\mu a\ (x))$

In rewrite system (8), any term of the system is reduced to normal form in a number of steps that is identical to number of symbol occurrences (i.e. occurrences of '*F*' and '*a*') of the term. If we take the size of the problem to be the size of the input term then the associated time complexity function $C(8)$ is the identity function. That is, $C(8)(k) = k$.

We c an easily speed up the system by adding a third rule:

(9) i) μa (a) → Annette

ii) μa (F (x)) → **F**(μa(x))

iii) μa (F (F (x))) → **F**(**F**(μa (x)))

Because of the third rule, two occurrences of '*F*' can be processed in one step. So with this addition we get another complexity function: $C(9)(k) = k/2+1$ for odd k (i.e. even number of *F* 's), and $k + 1/2$ for even k. Clearly, by applying this method, for each system we can find another that is more efficient with respect to time. Still there is an upper bound the speed-up. Since for any system there is finite number n such that no rule application processes more than n symbol occurrences, for that system each full reduction to normal form will take at least k/n steps. Hence, no system has reductions faster than linear time.

The speed-up between systems (8) and (9) is acquired at the cost of enlarging the rule system, adding a redundant rule. Hence, we can see that there is a trade-off between the size of the system, with respect to the number of rules, and the speed of the system. It is natural to ask for the speed of a system that has a minimal number of rules, i.e. a system R such that for any equivalent system R', one that reduces the same input terms to the same normal form terms, R' has at least the same size as R. It is natural to set identity as the maximum of efficiency for such a system. That is, if for a minimal rule system R the corresponding time complexity function C_R is such that $C_R(k) \leq k$, then we say that R has maximal efficiency. $C(8)$ is maximally efficient in this sense.

For languages without variable binding, as is shown in Pagin 2012a, a semantics with minimal complexity is compositional. The conversation does not hold: just as compositional semantics can fail to be computable at all, it can be computable but not efficient. But the requirement of maximal efficiency *induces* compositionality. Moreover, if we add first-order quantifiers, complexity will still be very low, although not minimal in the sense defined (cf. Pagin 2012b). Even more significantly, if do allow recursive semantics, then the time complexity function will be *exponential*, and hence the semantics will be *intractable* (cf. Pagin 2011). And if, by contrast, we disallow recursion over meanings, the resulting semantics is compositional. Hence, the requirement even of *tractable complexity* forces compositionality.

What is the bearing of these results on language acquisition and evolution? The simple immediate ideas are these: if a language is easier to interpret, then it is also easier to learn, and language evolution would tend to favour a semantics that makes interpretation easy. But it is possible to say something more interesting.

If complexity grows rapidly, then sentences of moderate syntactic complexity can be quite hard to process. A sentence like

(10) Whenever you go into a cave, bring a torch!

is of moderate syntactic complexity, and could obviously convey useful information. With a recursive semantics, the interpretation would require very many steps, rendering the sentence unusable in practice, even though interpretable in principle. Hence, we can see why there would be a selection pressure for low interpretation complexity. Compositional semantics allows greater expressive power, simply because a systematic but non-compositional semantics of a language with the same expressive power would be intractable.

References

Baader, Franz and Tobias Nipkow (1998). *Term Rewriting and All That.* Cambridge: Cambridge University Press.

Brighton, Henry J. (2005). 'Linguistic Evolution, and Induction by Minimum Description Length'. In: *The Compositionality of Concepts and Meanings: Applications to Linguistics, Psychology and Neuroscience.* Ed. by Markus Werning, Edouard Machery, and Gerhard Schurz. Vol. 2. Frankfurt/Lancaster: Ontos, pp. 13–39.

Chomsky, Noam (1971). 'Topics in the Theory of Generative Grammar'. In: *Philosophy of Language.* Ed. by John Searle. Oxford: Oxford University Press.

Davidson, Donald (1967). 'Truth and Meaning'. In: *Synthese* 17, pp. 304–23. Reprinted in Davidson 1984, 17-36.

— (1984). *Inquiries into Truth and Interpretation.* Oxford: Clarendon Press. Feigl, Herbert and Wilfrid Sellars, eds. (1949). *Readings in Philosophical Analysis.* New York: Appleton-Century Croft.

Frege, Gottlob (1892). 'Über Sinn und Bedeutung'. In: *Zeitschrift für Philosophie und Philosophische Kritik* 100, 22–50. Translated by Herbert Feigl as 'On sense and nominatum', in Feigl and Sellars 1949, 85–102.

— (1923). 'Gedankengefüge'. In: *Beiträge zur Philosophie des Deutschen Idealismus.* 3, pp. 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.

— (1980). 'Letter to Jordain 1914'. In: *Philosophical and Mathematical Correspondence.* Ed. by Gottfried et. al Gabriel. Chicago, Ill.: Chicago University Press, pp. 78–80.

Hendriks, Herman (2001). 'Compositionality and Model-Theoretic Interpretation'. In: *Journal of Logic, Language and Information* 10, pp. 29–48.

Hodges, Wilfrid (2001). 'Formal Features of Compositionality'. In: J*ournal of Logic, Language and Information* 10, pp. 7–28.

Janssen, T.M.V. (1986). *Foundations and Applications of Montague Grammar. Part 1, Philosophy, Framework, Computer Science.* CWI Tract 19. Amsterdam: Centre for Mathematics and Computer Science.

— (1997). 'Compositionality'. In: *Handbook of Logic and Language.* Ed. by Johan van Benthem and Alice ter Meulen. Amsterdam: Elsevier, pp. 417–73.

Li, Ming and Vitányi (1997). *An Introduction to Kolmogorov Complexity and its Applications.* 2nd ed. New York: Springer.

Montague, Richard (1970). 'Universal Grammar'. In: *Theoria* 36, pp. 373–98.

— (1973). 'The Proper Treatment of Quantification in Ordinary English'. In: *Approaches to Natural Language.* Ed. by Jaakko Hintikka, J. Moravcsik, and P. Suppes. Dordrecht: Reidel.

Pagin, Peter (2006). 'Meaning holism'. In: *Handbook of Philosophy of Language.* Ed. by Ernie Lepore and Barry Smith. Oxford: Oxford University Press.

— (2011). 'Compositionality, Computability, and Complexity'. Draft.

— (2012a). 'Communication and the Complexity of Semantics'. In: *The Oxford Handbook of Compositionality.* Ed. by Wolfram Hinzen, Edouard Machery, and Markus Werning. Oxford University Press, pp. 510–29.

— (2012b). 'Truth-theories, Competence, and Semantic Computation'. In: *Davidson's Philosophy: A Reappraisal.* Ed. by Gerhard Preyer. Oxford: Oxford University Press.

Pagin, Peter and Dag Westerståhl (2010a). 'Compositionality'. In: *Semantics. An International Handbook of Natural Language Meaning.* Ed. by Claudia Maienborn, Klaus von Heusinger, and Paul Portner. Elsevier.

— (2010b). 'Compositionality I: Definitions and Variants'. In: *Philosophy Compass* 5, pp. 250–64.

— (2010c). 'Compositionality II: Arguments and Problems'. In: *Philosophy Compass* 5, pp. 265–82.

— (2010d). 'Pure Quotation and General Compositionality'. In: *Linguistics and Philosophy* 33, pp. 381–415.

Peters, Stanley and Dag Westerståhl (2006). *Quantifiers in Language and Logic*. Oxford: Oxford University Press.

Potts, Christopher (2007). 'The Dimensions of Quotation'. In: *Direct Compositionality.* Ed. by Chris Barker and Pauline Jacobson. Oxford: Oxford University Press, pp. 405– 31.

Smith, Kenny and Simon Kirby (2012). 'Compositionality and Linguistic Evolution'. In: *The Oxford Handbook of Compositionality.* Ed. by Wolfram Hinzen, Edouard Machery, and Markus Werning. Oxford: Oxford University Press, pp. 493–509.

Szabó, Zoltán (2008). 'Compositionality'. In: *Stanford Encyclopedia of Philosophy.* Ed. by Edward N. Zalta. Stanford University. URL: http://plato.stanford.edu/archives/win2008/entries/compositionality/.

Werning, Markus (2005). 'Right and Wrong Reasons for Compositionality'. In: *The Compositionality of Concepts and Meanings: Foundational Issues.* Ed. by Markus Werning, Edouard Machery, and Gerhard Schurz. Frankfurt/Lancaster: Ontos, pp. 285–309.

Werning, Markus, Edouard Machery, and Gerhard Schurz, eds. (2005). *The Compositionality of Concepts and Meanings: Foundational Issues.* Frankfurt/Lancaster: Ontos, pp. 23–58.