Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

D.J.N. van Eijck

Formal semantics

Computer Science/Department of Software Technology Report CS-R9055 October

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Formal Semantics

Jan van Eijck
Centre for Mathematics and Computer Science
Kruislaan 413, 1098 SJ Amsterdam
The Netherlands

This paper gives an introduction to formal theories of meaning for natural language.

Note: The paper will appear in the *Encyclopedia of Language and Linguistics*, Pergamon Press and Aberdeen University Press.

1985 Mathematics Subject Classification: 68S10.
CR Categories: H.2.3, I.2.4, I.2.7.
Keywords and Phrases: semantics of natural language, query languages, knowledge representation languages.

1 Introduction

Formal semantics of natural language is the study of the meaning of natural language expressions using the tools of formal or symbolic logic. The ultimate aim of the discipline is commonly taken to include the following: to give an explication of the concept of meaning and to use this explication to provide an account of the rôle of language in the activity of drawing inferences (the individual processing of information), and of the rôle of language in the activity of sharing or communicating information (the collective processing of information).

This article provides an introduction to formal theories of meaning and to logical analysis of language in the footsteps of Gottlob Frege, the founding father of the enterprise. The process of composition of meaning and the relations between the concepts of meaning, inference and truth are presented, and contextual aspects of meaning are discussed. The article ends with a sketch of the emerging perspective of a dynamic theory of meaning.

2 The Composition of Meaning

Introductory logic textbooks usually include a selection of exercises for translating natural language sentences into first order logic. Invariably, these exercises assume that the student already has a firm grasp of what the sentences mean. The aim of such exercises is to expand the student's awareness of the expressive power of predicate logic by inviting him or her to express an (intuitively) well-understood message in the new medium. Because of this presupposed understanding of the original message, such translations cannot count as explications of the concept of meaning for natural language.

We should ask ourselves under what conditions a translation procedure from natural language into some kind of logical representation language can count as an explication of the concept of meaning. Obviously, the procedure should not presuppose knowledge of the meaning of complete natural language sentences, but rather should specify how sentence meanings are derived from the meanings of smaller building blocks. Thus, the meanings of complex expressions are

derivable in a systematic fashion from the meanings of the smallest building blocks occurring in those expressions. The meaning of these smallest building blocks is taken as given. It has been argued that the real mystery of semantics lies in the way human beings grasp the meaning of single words; see for example Percy (1954), or Plato's dialogue *Cratylus*.

Formal semantics has little or nothing to say about the interpretation of semantic atoms. It has rather a lot to say, however, about the process of composing complex meanings out of smaller building blocks. The intuition that this is always possible can be stated somewhat more precisely; it is called the Principle of Compositionality:

> The meaning of an expression is a function of the meanings of its immediate syntactic components plus their syntactic mode of composition.

The principle of compositionality is implicit in Gottlob Frege's writings on philosophy of language; it is made fully explicit in Richard Montague's approach to natural language semantics. Rather than indulge in philosophical reflections on the meaning of compositionality, we will illustrate the principle by showing how Alfred Tarski's definition of the semantics of *first order predicate logic* (see the article FIRST ORDER LOGIC) complies with it.

From the end of the nineteenth century until the nineteen sixties the main tool of semantics has been the language of first order predicate logic, so called because it is a tool for describing properties of objects of the first order in Bertrand Russell's hierarchy of *things*, *properties of things*, *properties of properties of things*, etcetera. Essentially, predicate logic was first presented in 1879 by Frege.

Modern semantic theories of natural language are generally not based on first order logic but on typed intensional logic, because of its still larger scope and because of the fact that this tool is more suited for a compositional treatment of the semantics of natural language. Typed logics and intensionality will be discussed further on in this article.

3 Meaning in Predicate Logic

According to Frege, the key concept of semantics is reference in the real world. For sentences, this boils down to truth simpliciter. Proper names are assumed to refer to individuals in the real world bearing those names, common nouns to sets of things in the real world having the appropriate properties, and so on. Because in this view any name names only one object, a sharp distinction between name and object is not crucial. At the start of the development of predicate logic, no sharp distinction was made between syntax and semantics.

Later on, there gradually emerged a clearer distinction between the *syntax* and the *semantics* of formal representation languages. The semantics of sentences of first order logic is then given in terms of classes of models (see the article MODEL) in which the sentences are true. The validity of inferences in first order logic, from a set of premisses to a conclusion, is in turn described in terms of truth: the valid inferences are precisely those with the property that any model in which all the premisses are true makes the conclusion true as well.

This habit of generalizing over models is a typical feature of formal semantics. The generalisation reflects the fact that validity of inferences concerns the form of the inferences, not their content. It should be borne in mind, though, that this concept of form is arrived at by generalizing over content. As far as semantics is

concerned with interpretation of language in appropriate models the discipline is concerned with (semantic) content as opposed to (syntactic) form.

The first clear discussion of the discipline of *semantics* conceived as the study of the relations between the expressions of a logical language and the *objects* that are denoted by these expressions is due to Tarski (1933). (See Tarski (1956) for an English translation.) This section gives the gist of the matter.

The non-logical vocabulary of a predicate logical language L consists of a set $\mathrm{C} = \{c_0, c_1, c_2, \ldots\}$ of names or individual constants, for each $n > 0$ a set $\mathrm{P}^n = \{P_0^n, P_1^n, P_2^n, \ldots\}$ of n-place predicate constants, and for each $n > 0$ a set $\mathrm{f}^n = \{f_0^n, f_1^n, f_2^n, \ldots\}$ of n-place function constants. Note that not all of these ingredients have to be present: in most cases, most of the P^n and f^n will be empty. A typical value for the highest n with either P^n or f^n non-empty is 3, which is to say that predicate or function constants with higher arity than 3 are quite rare. The *arity* of a predicate or function constant is its number of argument places.

The logical vocabulary of a predicate logical language L consists of parentheses, the connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$, the quantifiers $\forall$ and $\exists$, the identity relation symbol $=$ and an infinitely enumerable set V of *individual variables.*

If the non-logical vocabulary is given, the language is defined in two stages. The set of *terms* of L is the smallest set for which the following holds:

- If $t \in V$ or $t \in \mathrm{C}$, then t is a term of L.
- If $f \in \mathrm{f}^n$ and $t_1, \ldots, t_n$ are terms of L, then $f(t_1 \cdots t_n)$ is a term of L.

This definition says that terms are either individual variables or constants, or results of writing n terms in parentheses after an n-place function constant. Terms are the ingredients of formulae. The set of *formulae* of L is the smallest set such that the following holds:

- If t_1, t_2 are terms of L, then $t_1 = t_2$ is a formula of L.
- If $P \in \mathrm{P}^n$ and $t_1, \ldots, t_n$ are terms of L, then $Pt_1 \cdots t_n$ is a formula of L.
- If φ is a formula of L, then $\neg\varphi$ is a formula of L.
- If φ, ψ are formulae of L, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ and $(\varphi \leftrightarrow \psi)$ are formulae of L.
- If φ is a formula of L and $v \in V$, then $\forall v\varphi$ and $\exists v\varphi$ are formulae of L.

This completes the definition of the syntax. The semantic account starts with the definition of models. A *model* M for L is a pair $\langle D, I\rangle$ where D is a non-empty set and I is a function that does the following:

- I maps every $c \in C$ to a member of D.
- For every $n > 0$, I maps each member of P^n to an n-place relation R on D.
- For every $n > 0$, I maps each member of f^n to an n-place operation g on D.

D is called the *domain* of the model M, I is called its *interpretation function.*

Sentences involving quantification generally do not have sentences as parts but open formulae, i.e. formulae in which at least one variable has a free occurrence. As it is impossible to define truth for open formulae without making a decision about the interpretation of the free variables occurring in them, one

employs infinite *assignments* of values to the variables of L, that is to say functions with domain V and range $\subseteq D$. However, it is easy to see that only the finite parts of the assignments that provide values for the free variables in a given formula are relevant.

The assignment function s enables us to define a *value* function for the terms of L. Let the model $M = \langle D, I \rangle$ be fixed and let s be an assignment for L in D. The function v_s mapping the terms of L to elements of D is given by the following clauses:

- If $t \in C$, then $v_s(t) = I(t)$.
- If $t \in V$, then $v_s(t) = s(t)$.
- If t has the form $f(t_1 \cdots t_n)$, for some $f \in \mathrm{f}^n$, then $v_s(t) = I(f)(v_s(t_1), \ldots, v_s(t_n))$.

Note that the clause for function terms is recursive, and moreover, it precisely follows the recursion in the syntactic definition of such terms.

We continue to follow the syntactic definition in the semantics. The second stage of the semantic definition process consists of explaining what it means for an arbitrary formula φ of L to be true in the model relative to an assignment s. We recursively define a function $[\![\cdot]\!]_s$ mapping the formulae of L to the set of truth values $\{0, 1\}$ (0 for falsity, 1 for truth). The recursive definition follows the syntactic definition of the formulae of the language. First the basic case is handled where φ is an atomic formula.

1. If φ has the form $t_1 = t_2$, then $[\![\varphi]\!]_s = 1$ if and only if $v_s(t_1) = v_s(t_2)$.
2. If φ has the form $Pt_1 \cdots t_n$, then $[\![\varphi]\!]_s = 1$ if and only if $\langle v_s(t_1), \ldots, v_s(t_n) \rangle \in I(P)$.

The logical connectives are treated as follows:

3. If φ has the form $\neg\psi$, then $[\![\varphi]\!]_s = 1$ if and only if $[\![\psi]\!]_s = 0$.
4. If φ has the form $(\psi \wedge \chi)$, then $[\![\varphi]\!]_s = 1$ if and only if $[\![\psi]\!]_s = [\![\chi]\!]_s = 1$.

   If φ has the form $(\psi \vee \chi)$, then $[\![\varphi]\!]_s = 1$ if and only if $[\![\psi]\!]_s = 1$ or $[\![\chi]\!]_s = 1$.

   If φ has the form $(\psi \rightarrow \chi)$, then $[\![\varphi]\!]_s = 0$ if and only if $[\![\psi]\!]_s = 1$ and $[\![\chi]\!]_s = 0$.

   If φ has the form $(\psi \leftrightarrow \chi)$, then $[\![\varphi]\!]_s = 1$ if and only if $[\![\psi]\!]_s = [\![\chi]\!]_s$.

Finally, we consider the case of the quantifiers $\forall$ and $\exists$. To start with a simple example, suppose we want to describe the circumstances under which $\forall x Px$ is true. In the description we want to refer to information about the truth or falsity of Px, for we want the account to be compositional. Saying that Px must be true in the model, given s, is not enough, because $[\![Px]\!]_s$ depends on the value that s assigns to x: $[\![Px]\!]_s = 1$ if and only if $s(x) \in I(P)$. What we want to say is something different: Px is true no matter which individual is assigned to x. This means that we are interested in assignments that are like s except for the fact that they may assign a different value to some variable v. Here is a precise definition:

$$s(v|d)(w) = \begin{cases} s(w) & \text{if } w \neq v \\ d & \text{if } w = v. \end{cases}$$

Armed with this new piece of notation we dispose of the quantifier case:

5. If φ has the form $\forall v\psi$, then
   $[\![\varphi]\!]_s = 1$ if and only if $[\![\psi]\!]_{s(v|d)} = 1$ for every $d \in M$.

   If φ has the form $\exists v\psi$, then
   $[\![\varphi]\!]_s = 1$ if and only if $[\![\psi]\!]_{s(v|d)} = 1$ for at least one $d \in M$.

This completes the definition of the function $[\![\cdot]\!]_s$. If $[\![\varphi]\!]_s = 1$ we say that assignment s satisfies φ in the model, or that φ is true in the model under assignment s.

As was remarked above, truth or falsity of a formula in a model under an assignment s only depend on the finite part of s that assigns values to the free variables of the formula. Sentences do not contain free variables, so the truth of falsity of a sentence in a model does not depend on the assignment at all. We say that a sentence φ of L is true in a model if φ is true in the model under every assignment s. Equivalently, we could have said that φ is true in the model if φ is true under some assignment. The notion of an assignment was a tool that can now be discarded.

The main feature of the Tarski semantics for predicate logic is its recursive nature: the meaning of a complex formula is recursively defined in terms of the meanings of its components. This is what compositionality is all about.

4 Abstraction and Quantification

In the above presentation of the semantics of first order predicate logic, quantifiers were introduced syncategorematically, which is to say that they are not regarded as building blocks of the language in their own right. It follows that the quantifiers do not have meanings of their own. The compositional semantics of first order predicate logic would look more elegant if the quantifiers were to be considered building blocks. This can be done by means of the concept of *abstraction.*

Abstraction as a conceptual tool is already used by Frege, but his notation is rather awkward. Rather than stick with Frege's presentation we present a version using lambda operators or λ-operators (see the article LAMBDA OPERATOR). Lambda operators were introduced by Alonzo Church (1940). We will show how this device can be used to construct meanings for separate building blocks of languages. In doing so we sketch a version of typed logic (see the article TYPED LOGICS). Typed logics are currently the most widely used tools for representing the semantics of natural language expressions.

The fact that in example (1) *John* can be replaced by *Fred* to form a new sentence shows that we can abstract from *John.*

(1) *John repects Bill.*

This process of abstraction starts with a sentence, removes a proper name, and yields a function from proper names to sentences, or semantically, a function from individuals to truth values, i.e. a characteristic function (see the article CHARACTERISTIC FUNCTION). Lambda operators allow us to refer to this function explicitly.

(2) $\lambda x.x$ *respects Bill.*

The function denoted by (2) corresponds to the property of respecting Bill. For convenience we call functions of this type *properties*. Next, we can abstract from the second proper name, and we get a functional expression denoting the relation of respecting.

(3) $\lambda y \lambda x.x \textit{ respects } y.$

Expression (3) denotes the relation of respecting by presenting it as a function that combines with an individual to form a property (a function from individuals to truth values). Actual variable names are unimportant but binding patterns matter: (3) and (4) are (logically equivalent) alphabetic variants.

(4) $\lambda x \lambda y.y \textit{ respects } x.$

The relation of being respected is denoted by a slightly different lambda expression (5).

(5) $\lambda x \lambda y.x \textit{ respects } y.$

The distinction between the active and passive voice is reflected in the different binding patterns of (4) and (5).

In the context of lambda operators, quantifiers can be viewed as higher order functions. The quantifier $\forall$ combines with the expression $\lambda x.\varphi$, where φ is a sentence, to form a sentence $\forall \lambda x.\varphi$ (which we will continue to write as $\forall x \varphi$). Observe that the quantifier itself does not have to act as a binder any more. The binding mechanism is taken over by the lambda operator. The sentence $\forall x \varphi$ is true if and only if $\lambda x.\varphi$ denotes a function which gives true for every argument. Thus, $\forall$ denotes a function from characteristic functions to truth values. It maps every characteristic function that always gives true to true, and every other characteristic function to false. Similarly, $\exists$ denotes the function from characteristic functions to truth values which maps every function that for some value assumes the value true, to true, and the function that assigns false to every argument, to false.

Lambda abstraction and quantifiers put one in the fortunate position of being able to express what it means to admire an attractive girl (6).

(6) $\lambda x.\exists y(\textit{girl } y \wedge \textit{ attractive } y \wedge x \textit{ admires } y).$

To be courted by every unmarried man, on the other hand, is something quite different, as expression (7) makes clear.

(7) $\lambda x.\forall y((\textit{man } y \wedge - \textit{ married } y) \rightarrow y \textit{ courts } x).$

It is also possible to abstract over objects of more complex types. Again starting from (1), one can abstract over the transitive verb or over the predicate. Abstracting over the predicate yields (8), an expression which combines with a property denoting expression (i.e., a predicate) to form a sentence.

(8) $\lambda P.P(\textit{john}).$

Interestingly, (8) is an expression of the same type as that of quantified noun phrases. The quantified noun phrase *every man* combines with a property denoting expression to form a sentence, so (9) is an appropriate translation.

(9) $\lambda P.\forall x(\textit{man } x \rightarrow P(x)).$

Combining (9) with (2) gives the expression in (10). In fact, for convenience we have replaced (2) with an alphabetic variant (an expression using different

variables to effect the same binding pattern). This kind of conversion is called α conversion.

(10) $\lambda P.\forall x(man\ x \rightarrow P(x))(\lambda y.y\ respects\ Bill)$.

Expression (10) is the result of combining the translation of *every man* with that of *respects Bill*. The result should be a sentential expression, i.e. an expression denoting a truth value. To see that this is indeed the case, a reduction of the expression is necessary. All expressions of the form $\lambda v.E(A)$ are reducible to a simpler form; they are called *redexes* (*red*ucible *ex*pressions).

5 Reducing Lambda Expressions

To reduce expression (10) from the previous section to its simplest form, two steps of so-called β conversion are needed. During β conversion of an expression consisting of a functor expression $\lambda v.E$ followed by an argument expression A, basically the following happens (we will state a *proviso* shortly). The prefix $\lambda v.$ is removed, the argument expression A is removed, and finally the argument expression A is substituted in E for all free occurrences of v. The free occurrences of v in E are precisely the occurrences which were bound by λv in $\lambda v.E$.

Here is the *proviso*. In some cases, the substitution process described above cannot be applied without further ado, because it will result in unintended capturing of variables within the argument expression A. Consider expression (11).

(11) $(\lambda x\lambda y.R(y)(x))(y)$.

In this expression, y is bound in the functional part $\lambda x\lambda y.R(y)(x)$ but free in the argument part y. Reducing (11) by β conversion would result in $\lambda y.R(y)(y)$, with capture of the argument y at the place where it is substituted for x. The problem is sidestepped if β conversion is performed on an alphabetic variant of the original expression, say on $(\lambda x\lambda z.R(z)(x))(y)$.

Another example where α conversion (i.e., switching to an alphabetic variant) is necessary before β conversion to prevent unintended capture of free variables is the expression (12).

(12) $\lambda p.\forall x(A(x) \equiv p)(B(x))$.

In (12), p is a variable of type truth value, and x one of type individual entity. Then x is bound in the functional part $\lambda p.\forall x(A(x) \equiv p)$ but free in the argument part $B(x)$. Substituting $B(x)$ for p in the function expression would cause x to be captured, with failure to preserve the original meaning. Again, the problem is sidestepped if β conversion is performed on an alphabetic variant of the original expression, say on $\lambda p.\forall z.(A(z) \equiv p)(B(x))$. The result of β conversion then is $\forall z.(A(z) \equiv B(x))$, with the argument of B still free, as it should be.

Using $[A/v]$ for the substitution operation we can express the β-reduction step formally as follows. Suppose $\lambda v.E(A)$ is an expression where all bound variables are different from the free variables. This condition constitutes a straightforward way of making sure that the problem mentioned above will not occur. Then $\lambda v.E(A)$ β-reduces to $[A/v]E$.

Applying the procedure of β reduction to (10), we see that a first β conversion step reduces (10) to (13), and a second, internal, β conversion step then yields (14).

(13) $\forall x(man\ x \rightarrow \lambda y.y\ respects\ Bill(x))$.

(14) $\forall x(man\ x \rightarrow x\ respects\ Bill)$.

The process of reducing lambda expressions has drastic consequences for their syntactic appearance. Historically, the syntactic form of logical expressions translating natural language sentences was taken to reflect the *logical form* of these sentences. In the next section it is pointed out that the metamorphosis of β conversion bears on certain historical problems of logical form.

6 Misleading Form and Logical Form

From *John walked* it follows that someone walked, but from *No-one walked* it does not follow that someone walked. Therefore, logicians such as Frege, Russell, Tarski and Quine have maintained that the structure of these two sentences must differ, and that it is not enough to say that they are both compositions of a subject and a predicate.

The logicians who used first order predicate logic to analyse the logical structure of natural language were struck by the fact that the logical translations of natural language sentences with quantified expressions did not seem to follow the linguistic structure. In the logical translations, the quantified expressions seemed to have disappeared. The logical translation of (15) does not reveal a constituent corresponding to the quantified subject noun phrase.

(15) *Every unmarried man courted Mary.*

(16) $\forall x((man\ x \wedge -married\ x) \rightarrow x\ courted\ mary)$.

In the translation (16) the constituent *every unmarried man* has disappeared: it is contextually eliminated. Frege remarks that a quantified expression like *every unmarried man* does not give rise to a concept by itself (*eine selbstandige Vorstellung*), but can only be interpreted in the context of the translation of the whole sentence. Applied to this particular example: the literal paraphrase of (16) is:

(17) *All objects in the domain of discourse have the property of either not being unmarried men or being objects who courted Mary.*

In restatement (17) of sentence (15) the phrase *every unmarried man* does not occur any more.

The logical properties of sentences involving quantified expressions (and descriptions, analyzed in terms of quantifiers) suggested indeed that the way a simple noun phrase such as a proper name combines with a predicate is logically different from the way in which a quantified noun phrase or a definite description combines with a predicate. This led to the belief that the linguistic form of natural language expressions was misleading.

The application of the logical tools of abstraction and reduction allow us to see that this conclusion was unwarranted. Using translation of natural language in expressions of typed logic we see that natural language constituents correspond to typed expressions that combine with one another as functions and arguments. After full reduction of the results, quantified expressions and other constituents may have been contextually eliminated, but this elimination is a result of the reduction process, not of the supposed misleading form of the original natural language sentence. Thus we see that, while fully reduced logical translations of natural language sentences may be misleading in some sense, the fully unreduced original expressions are not.

As an example of the way in which the λ tools smooth logical appearances, consider the logic of the combination of subjects and predicates. In the simplest cases (*John walked*) one could say that the predicate takes the subject as an

argument, but this does not work for quantified subjects (*no-one walked*). All is well, however, when we say that the subject always takes the predicate as its argument, and make this work for simple subjects by logically raising their status from argument to function. Using λ, this is easy enough: we translate *John* not as the constant j, but as the expression $\lambda P.P(j)$. This expression denotes a function from properties to truth values, so it can take a predicate translation as its argument. The translation of *no-one* is of the same type: $\lambda P. \neg \exists x.(person\ x \wedge P(x))$. Before reduction, the translations of *John walks* and *no-one walks* look very similar. These similarities disappear only after both translations have been reduced to their simplest forms.

7 Meaning in Natural Language

In a Montague style approach to natural language, one takes for the natural language syntax some version of categorial grammar enriched with quantifying in rules (to be discussed at the end of this section), and for the semantics some form of typed logic. The combination of categorial grammar and typed logic allows the link between syntax and semantics to be of the utmost simplicity. Lexical items get assigned categories such as CN for common nouns, IV for intransitive verbs, S/IV for noun phrases (these take intransitive verb phrases on their right to form sentences), (S/IV)/CN for determiners (these take common nouns on their right to form noun phrases), IV\IV for adverbial modifiers (these take intransitive verb phrases on their left to make new intransitive verb phrases). See the article CATEGORIAL GRAMMAR for further details.

Next, the lexical items get assigned translations with types matching the syntactic categories. In the simplest possible approach, the translation types are defined in terms of simplest types e (for entities) and t (for truth values). Formulae have type t, individual terms have type e. Expressions denoting functions from type A to type B have type (A, B). It follows from this rule that property denoting expressions have type (e, t). In general, if variable v has type A and expression E has type B, then $\lambda v.E$ has type (A, B). If E has type (A, B) and a has type A, then the expression $E(a)$ is well-formed and has type B.

It is not difficult to see which types are suitable for which syntactic categories. Sentences, category S, should translate into formulae, with type t. Intransitive verbs, category IV, translate into properties, type (e, t). Common nouns, category CN, also translate into properties. The rest of the category-to-type match is taken care of by a general rule. If category X translates into type A, and category Y translates into type B, then categories X/Y (takes a Y to the right to form an X) and Y\X (takes a Y to the left to form an X) translate into type (A, B).

The rules for syntactic function application are then matched by rules for semantic function application. The meaning of *John loves Mary* is derived in two steps. First *loves* and *Mary*, with categories IV/(S/IV) and S/IV respectively, combine to form *loves Mary*, with category IV. The meaning of this expression is derived from the meanings of the components by function–argument application. Next *John* and *loves Mary*, with categories S/IV and IV respectively, are combined to form *John loves Mary*, with category S. Again, the meaning is derived from the meanings of the components by function–argument application. Further details on how quantifiers and determiners are interpreted in this setup are provided, respectively, in the articles QUANTIFIERS and DETERMINERS.

Let us now briefly examine the notion of *ambiguity* for fragments of natural language with a compositional semantics. If a natural language expression E

is ambiguous, i.e. if E has several distinct meanings, then, under the assumption that these meanings are arrived at in a compositional way, there are three possible sources for the ambiguity (combinations are possible, of course):

1. The ambiguity is lexical: E contains a word with several distinct meanings. Example: *a splendid ball.*

2. The ambiguity is structural: E can be assigned several distinct syntactic structures. Example: *old [men and women]* versus *[old men] and women.*

3. The ambiguity is derivational: the syntactic structure that E exhibits can be derived in more than one way. Example: *Every prince sang some ballad.* is not structurally ambiguous, but in order to account for the $\exists\forall$ reading one might want to assume that one of the ways in which the structure can be derived is by combining *some ballad* with the incomplete expression *every prince sang* —.

Derivational ambiguities are very much a logician's ploy. In an essay on philosophical logic by P.T. Geach they are introduced as follows:

> [...] when we pass from "Kate / is loved by / Tom" to "Some girl / is loved by / every boy", it does make a big difference whether we first replace "Kate" by "some girl" (so as to get the predicable "Some girl is loved by —" into the proposition) and then replace "Tom" by "every boy", or rather first replace "Tom" by "every boy" (so as to get the predicable "— is loved by every boy" into the proposition) and then replace "Kate" by "some girl". Two propositions that are reached from the same starting point by the same set of logical procedures (e.g. substitutions) may nevertheless differ in import because these procedures are taken to occur in a different order. (Geach 1962, section 64.)

This is exactly the mechanism that has been proposed by Richard Montague to account for operator scope ambiguities in natural language. Montague introduces a rule for *quantifying in* of noun phrases in incomplete syntactic structures. The wide scope $\exists$ reading for the example *Every prince sang some ballad* is derived by using a rule Q_i to quantify in *some ballad* for syntactic index i in the structure *Every prince sang* PRO_i. In more complex cases, where more than one occurrence of PRO_i is present, the appropriate occurrence is replaced by the noun phrase, and the other occurrences are replaced by pronouns or reflexives with the right syntactic agreement features. See Montague (1973) for details.

The Montagovian approach to scope ambiguities does not account for restrictions on possible scope readings. It is not denied that such restrictions should be imposed, but they are relegated to constraints imposed by lexical features of determiner words. A problem here is that scope behaviour of certain natural language expressions seems to be influenced by the wider syntactic context. See the article SCOPE AMBIGUITIES for information.

8 Meaning, Truth and Inference

Typed logics are the proper logical tool for describing the semantics of natural language. One way to go about generalizing the model concept of predicate logic for languages of typed logic is as follows. A model for a typed logic based on individual objects and truth values starts out with a universe U for the domain of individual objects, and the set $\{1,0\}$ for the domain of truth values.

Next, more complex domains are construed in terms of those basic domains. The domain of properties is the set of functions $U \rightarrow \{1,0\}$, i.e. the set of all functions with U as domain and the truth values as co-domain. The domain of characteristic functions on properties (the type of things to which the quantifiers $\forall, \exists$ belong) is the set $(U \rightarrow \{1,0\}) \rightarrow \{1,0\}$, and so on. Another generalization is also possible, by defining so-called *general models*: see the article on TYPED LOGICS for details.

Models for typed logical languages can be used to define functions $[\![\cdot]\!]$ which map all expressions of the language to objects of the right types in the model: sentence type expressions to truth values, property type expressions to properties, object-to-property type expressions to functions from objects to properties, and so on. Let us assume for a moment that the basic domain U is the set of natural numbers, and that the predicate letter O stands for being an odd number. Then the interpretation of a property expression $\lambda x.O(x)$, notation $[\![\lambda x.O(x)]\!]$, is a function f which yields 1 for every $u \in U$ which is an odd number, and 0 for all even numbers. The property expression $\lambda x. -O(x)$ has the characteristic function of all even numbers as its interpretation. The interpretation of $\lambda P.\forall P$ (universal quantification over individuals), notation $[\![\lambda P.\forall P]\!]$, is the function $F \in (U \rightarrow \{1,0\}) \rightarrow \{1,0\}$ with $F(f) = 1$ for the function f in $U \rightarrow \{1,0\}$ with $f(u) = 1$ for all $u \in U$, and $F(g) = 0$ for all $g \neq f$. As a final example, $\lambda P.\forall x.(-O(x) \rightarrow P(x))$ has as its interpretation the characteristic function F which maps every function $f \in U \rightarrow \{1,0\}$ mapping every even number to 1, to 1, and all other functions in $U \rightarrow \{1,0\}$ to 0. Note that $\lambda P.\forall x.(-O(x) \rightarrow P(x))$ would be an appropriate translation for the natural language phrase *every non-odd thing*.

The model theoretic approach to meaning now equates the intuitive concept of *meaning* with the precise model theoretic concept of *interpretation*, i.e. with values of a function $[\![\cdot]\!]$ generated by an appropriate model. Note that $[\![\cdot]\!]$ is ultimately defined in terms of the interpretations of certain basic expressions in the model (for instance, predicates like O, universal quantification over individuals). The interpretation of these basic expressions can be described in terms of truth in the model, modulo appropriate assignments to free variables. The meaning of O reduces to the truth or falsity of $O(x)$ in the model, relative to assignments of individuals to x, the meaning of universal quantification over individuals reduces to truth or falsity of $\forall x.\varphi$ in the model, relative to assignments of properties to $\lambda x.\varphi$, and so on. Thus we can say that meaning is ultimately defined in terms of truth in a model.

Next, logical validity of inferences is defined in terms of truth, by saying that an inference from premisses φ_1 through φ_n to conclusion ψ is valid if and only if every model in which all of φ_1 through φ_n evaluate to true will make ψ true as well. In fact, for typed languages, the concept of logical validity can be extended to arbitrary expressions denoting characteristic functions. Let E_1 and E_2 be expressions for characteristic functions of the same type. Let us say that one characteristic function F_1 involves another one, F_2, if F_1 and F_2 have the same types and F_2 yields 1 for every argument for which F_1 yields 1. Then E_1 logically involves E_2 if and only if in every model, $[\![E_1]\!]$ involves $[\![E_2]\!]$. To give a rather trivial example, $\lambda x. -O(x)$ logically involves $\lambda x.(O(x) \vee -O(x))$. Typed languages are powerful enough to express involvement in a formula. E_1 logically involves E_2 if and only if the formula $\forall P.(E_1(P) \equiv E_2(P))$ is true in every model. Here P is a variable of the right type for arguments of E_1 and E_2.

Conversely, one may want to impose certain restrictions on the possible interpretations of the basic vocabulary by stipulating that certain concepts should involve others. For instance, one may want to ensure that the concept of walking involves the concept of moving relative to something. Assuming

that we have expressions $\lambda x.W(x)$ for walking and $\lambda y \lambda x.M(x,y)$ for moving with respect to, we can express the requirement as: $\lambda x.W(x)$ should involve $\lambda x.\exists y.(object(y) \wedge M(x,y))$. The semantic requirement is then imposed by restricting attention to models in which the first concept does indeed involve the second one. The desired involvements can be expressed as formulae. Such formulae, intented to constrain the class of possible models with the purpose of enforcing certain relations between elements in the vocabulary of the language, are called *meaning postulates*. Given a natural language fragment and a set of meaning postulates for that fragment, a sentence of the fragment is *analytic* if it is true in any model satisfying all the meaning postulates. A sentence of the fragment is *synthetic* if it does have counterexamples among the models satisfying all the meaning postulates. Given that the meaning postulates constrain the meanings of the vocabulary in the right way, we may assume that the real world (or some suitable aspect of it) will make all the meaning postulates true, so the synthetic sentences are precisely those that the world could be in disagreement with. The logically valid sentences are those that are true in any model of the language, irrespective of any meaning postulates.

This overview of models, interpretations, logical inference, logical involvement, and the analytic/synthetic distinction makes clear that truth is the tortoise which carries the whole edifice of semantics on its back. See the article TRUTH AND PARADOX for further discussion of this key concept.

9 Meaning in Context

The very simple account of meaning given in the previous sections breaks down if one wants to extend the treatment to intensional phenomena. Consider example (18).

(18) *John seeks a girlfriend.*

Example (18) might mean that John is looking for Sue, who happens to be his girlfriend, but it might also mean that John is answering small ads in the lonely hearts column because he wants to create a situation in which he has a girlfriend.

The setup of the previous sections would only give us the first sense of the sentence. A standard way to get the second sense is by making a distinction between *extensional* and *intensional* interpretations of phrases. The extensional interpretation of a phrase is its interpretation in a given context. The intensional interpretation is somewhat more involved: it is a function from contexts to (extensional) interpretations in those contexts. Under the second reading of (18), John is related, not to an individual, but to the set of possible contexts which contain a girlfriend for him. Such possible contexts are often called possible worlds; see the article POSSIBLE WORLDS. So how do we characterize a context which contains a girlfriend? Following Lewis (1970), we call the set of all properties of a thing the *character* of the thing. Some reflection will show that expression (19) denotes the characteristic function of the character of a girlfriend:

(19) $\lambda P.\exists y(girlfriend(y) \wedge P(y))$.

Expression (19) gives the character of a girlfriend in the actual context, but that is not quite what we want. To achieve what we want we must interpret basic expressions such as common nouns relative to contexts, with $girlfriend(y,c)$ meaning that y is a girlfriend in context c. Expression (20) gives a mapping

from contexts to characters of girlfriends in these contexts.

(20) $\lambda c \lambda P. \exists y(\mathit{girlfriend}(y, c) \wedge P(y, c))$.

When John is seeking a girlfriend in the intensional sense he is related to the item specified by (20).

Intensional interpretations are also useful for interpreting propositional attitude sentences, such as the example in (21).

(21) *John believes that a ghost is haunting his house.*

The embedded that–clause in this example cannot have its extensional interpretation, for if in fact no ghost is haunting John's house (let us assume for convenience that ghosts do not exist), then the extensional interpretation is just the truth value *false*, and John's beliefs are not as closely tied to the actual world as that. Rather, (21) is true just in case any situation or context compatible with John's belief is such that in that context a ghost is haunting his house. The translation for the embedded clause that we need to get this is given in (22).

(22) $\lambda c. \exists y(\mathit{ghost}(y, c) \wedge \mathit{haunt}(y, \mathit{house\text{-}of}(\mathit{john}, c), c))$.

Note that the common noun translations and the translation of the transitive verb *haunt* all have an extra argument for the context. The translation of the proper name does not, for we take it that proper names denote the same individual in any context. One way to tackle Frege's famous Morning Star/Evening Star paradox in terms of intensions would be to make names context-sensitive too We will not spell out details, as such a solution is not without its philosophical difficulties. The source of the paradox is not so much change of reference of proper names in other contexts, but incomplete information about the identity relation in those contexts (see the article IDENTITY).

It should be noted that the shift from extensional to intensional interpretations by no means solves all problems of sense and reference (see the article SENSE AND REFERENCE). Intensions are still not fine–grained enough to distinguish between equivalent statements of logic. Because logical truths are true independent of context, $2 + 2 = 4$ is true in all contexts, and so is *Zorn's lemma is equivalent to the Axiom of Choice.* Still, *John knows that* $2 + 2$ *equals* 4 hardly warrants the conclusion *John knows that Zorn's lemma is equivalent to the Axiom of Choice.* John may never have heard about set theory in the first place. One possible way out is to make meanings still more fine–grained, by taking them to be structured trees with intensions at their leaves. See Lewis (1970) for details. Indeed, Lewis reserves the term *meaning* for such structured objects.

Now that we have mentioned contexts, we might as well acknowledge the fact that the context in which a natural language statement is made—let us call it the current context—plays a very special rôle in the interpretation of the sentence. Tense operators are interpreted with respect to the time of utterance, personal pronouns *I*, *you*, are interpreted as speaker and addressee in the current context, demonstratives can be used to *anchor* the discourse to items in the current context. The anchoring mechanism has to be defined with some care, for it should be able to account for the anchoring of sentences like *This is cheaper than this, but this is nicer than this* (with four acts of pointing to different objects while the sentence is being uttered). Again, see Lewis (1970) for some suggestions.

In the above we have more or less equated contexts with possible worlds, i.e. alternative complete pictures of what the world might have been like. It

is argued in Barwise (1981) that some contexts of linguistic utterance are essentially incomplete. This observation has led to the development of strategies for interpreting natural language with respect to *partial models* or *situations*. See the article SITUATION SEMANTICS for a full–fledged theory along these lines, and Muskens (1989) for an attempt to incorporate partiality in a more traditional account.

10 The Meanings of Non-Indicative Expressions

We have seen that the extensional interpretation of a declarative sentence is a truth value, and its intensional interpretation a set of contexts. The extensional interpretation of *John loves Mary* in a model is either the value *true* or the value *false*. The intensional interpretation of this sentence is the set of contexts where the sentence has the value *true*. The intensional interpretation is needed in cases where the sentence occurs in embedded contexts, such as *Bill believes that John loves Mary*, which is standardly interpreted as true just in case *John loves Mary* is true in any context which is compatible with everything that Bill believes. So much for the semantics of the indicative mood. How about such non-indicative moods as questions and commands? Can their semantics be related to the semantics of the indicative mood?

Broadly speaking, the indicative mood is for *describing* situations, the interrogative mood for *checking* situations, and the imperative mood for (giving directions for) *changing* situations. A declarative sentence picks out a set of contexts where the sentence is true; its (intensional) meaning has the form $\lambda i.Pi$, where P is a predicate of contexts. Now take simple yes/no questions, for example. A yes/no question such as *Does John love Mary?* is an invitation to check whether the indicative root of the question, namely the statement *John loves Mary*, is true in the situation we are in. A check is an action, and actions are transformations from situations to other situations. Thus, a yes/no question $P?$ denotes a relation $\lambda i\lambda j.(i = j \wedge Pj)$. In other words, a yes/no question relates the set of states of the world to the set of states where the answer to the question is *yes*. This dynamic view of questions can be related to a more static picture. See Hintikka (1976) or Groenendijk & Stokhof (1984, 1988) for the static semantics of questions, Van Benthem (1990) for relations between a static and a dynamic view.

Utterances in imperative mood can be interpreted as commands to change the context, i.e. as mappings from contexts to intended contexts which are the result if the command is obeyed. Again, a dynamic perspective naturally presents itself. The command *Close the door!* relates situations to new situations where the door is closed. A command $P!$ denotes a relation $\lambda i\lambda j.Pj$. In other words, a command relates the set of states of the world to the set of states of the world where the command is fulfilled. Of course, there is much to be said about felicity conditions of imperatives (*Close the door!* only makes sense when the door is open), but likewise there is much to be said about felicity conditions of questions and declaratives. See the article PRAGMATICS for further illumination.

What matters here is that a command like *Close the door!* can be interpreted as a relation between the current context c_0 and the set of all contexts which are the result of performing the action of closing the door in c_0 in some way or other. The result of uttering the command need not be that one ends up in a context like c_0 but with a closed door (not all commands are obeyed, fortunately), but that does not matter to the principle of the account. Note that, just as in the dynamic account of questions, the concept of truth continues to play

an important rôle. The contexts that are like the current context but where the command has been obeyed are contexts where the sentence which is the declarative root of the imperative is true.

11 The Dynamics of Meaning

In the previous section we have started to look at meaning in a dynamic way. Instead of focussing on the question 'How are linguistic expressions semantically linked to a (static) world or model?' we have switched to a new question: 'How do linguistic messages viewed as actions change the current situation?' Not only a question or a command, but every linguistic utterance can be viewed as an action: it has an effect on the state of mind of an addressee, so one could say that the dynamic meaning of a natural language utterance is a map from states of mind to states of mind. Such talk about influencing states of mind is no more than a metaphor, of course; to make this precise one needs to replace 'state of mind' by a more precise concept of *state*. An obvious place to look for inspiration is the semantics of programming languages, where the meaning of a program is taken to be the effect that it has on the memory state of a machine: the dynamic meaning of a computer program is a mapping from memory states to memory states. Van Benthem (1990) looks at the link between programming language semantics and natural language semantics in some detail and presents a uniform picture of how static and dynamic views of language are related.

In imperative programming, e.g. in a language like Pascal, on program startup part of the storage space of the computer is divided up in segments with appropriate names. These are the names of the so–called *global variables* of a program, but in order to avoid confusion with logical variables we will call these *store names*. The effect of a program can be specified as a relation between the states of the stores before the execution of the program and the states of the stores afterward. A memory state is a specification of the contents of all stores, in other words, it is a function from the set of stores to appropriate values for the contents of these stores. Equivalently, we can look at each individual store as a function from states to appropriate values for that store. In this perspective on stores as functions from states to values, we can say things like $v_1(i) = 3$, meaning that the content of store v_1 in state i is 3.

Suppose a program consists of one command, $v_1 := 3$, the command to put the value 3 in the store with name v_1. Then the effect of this program on a given state i is a new state j which is just like i except for the fact that $v_1(j) = 3$ (where the value for v_1 in i might have been different). We will use the abbreviation $i[v]j$ for 'state i and state j differ at most in the value store v assigns to them'. We will assume that if i is a state and v a store, then there will always exist a state j with $i[v]j$.

In a language like Pascal every store has a specific type: some stores are reserved for storing strings, others for integer numbers, others for real numbers, and so on. For a rough sketch of how to use the dynamics metaphor for an account of anaphoric linking in natural language, these storage types do not matter. We will assume all stores to be of the same type: we take it that they are all used to store (names of) entities. We can again use typed logic as a medium of translation, but now we need an extra basic type s. For ease of exposition, we forget about contexts and intensionality again, and go back to an extensional treatment. Using s for the type of states and e for the type of entities, we can express our assumption that all stores store entities as follows: every store is of type (s, e).

Suppose we have a list of store names $V = v_1, v_2, \ldots$. We want to account for

anaphoric links as in example (23). The intended anaphoric links are indicated by subscripts.

(23) A_1 *man loves* a_2 *woman.* He_1 *kisses* her_2.

The difficulty with the traditional account of the semantics of (23), where the anaphoric links are established by a variable binding mechanism, is that the pronouns in the second sentence can only be translated as variables bound by the existential quantifiers in the first sentence if the quantifiers extend their scopes beyond the sentence limit. The scope of an existential quantifier has to be closed off somewhere, but there appears to be no natural place for a boundary. Wherever one puts the closing–off point, beyond it pronouns might still turn up that have to be linked to the same antecedent. Some theories have tried to solve this problem by translating indefinite and definite noun phrases as some sort of free variables (see the article DISCOURSE REPRESENTATION THEORY). We will sketch now how a dynamic approach to pronoun–antecedent linking can be integrated in a traditional Montague style grammar.

In the dynamic approach, sentences are translated as relations between states, in other words, sentences are of type $(s,(s,t))$: a sentence interpretation takes two states and then gives a truth value. Sentences that do not have a dynamic effect will not change the state, for instance the 'dynamic' translation of *John loves Mary*, in typed logic, will be something like (24).

(24) $\lambda i \lambda j.(i = j \wedge love(john, mary))$.

The translation of A_1 *man loves* a_2 *woman*, on the other hand, does involve state changes, because the dynamic interpretation of the indefinite noun phrases involves assigning values to stores. The interpretation of a_1 *man* is a relation between states i and j which holds just in case $i[v_1]j$ and the value of v_1 in state j is indeed a man.

(25) $\lambda P \lambda i \lambda j.(i[v_1]j \wedge man(v_1 j) \wedge P(v_1 j))$.

Note that $i[v_1]j$ is used here as abbreviation for the appropriate expression of typed logic that we will not bother to spell out. The translation of the common noun *man* in (25) does not involve states, and neither does the variable P that is proxy for the translation of the verb phrase. This is not quite right, for common nouns can contain relative clauses with dynamic effects, and verb phrases can have dynamic effects as well, so translations of common nouns and verb phrases must contain the means to accommodate these. In other words, the states must be 'threaded' through all these constituents. In case of a lexical noun such as *man* the net effect of the threading is nil, but the variable P must be of type $(s,(s,(e,t)))$ to cater for state switches in the verb phrase. The translation for a_1 *man* now looks like (26).

(26) $\lambda P \lambda i \lambda k.\exists j(i[v_1]j \wedge man(v_1 j) \wedge P(v_1 j, j, k))$.

Here is the dynamic translation for A_1 *man loves* a_2 *woman.*

(27) $\lambda i \lambda k.\exists j(i[v_1]j \wedge man(v_1 j) \wedge j[v_2]k \wedge love(v_1 k, v_2 k) \wedge woman(v_2 k))$.

The interpretation for the pronoun he_1 makes use of the contents of store v_1. Thus, the anaphoric links are provided by the retrieval of stored values. Expression (28) give the translation for the pronoun he_1 that has the desired effect; P is again a variable of type $(s,(s,(e,t)))$.

(28) $\lambda P \lambda i \lambda j.P(v_1 i, i, j)$.

The translation of the whole discourse (23) is left to the reader. A pioneer paper on dynamic interpretation of natural language is Barwise (1987). Groenendijk & Stokhof (1990) and Muskens (1990) contain worked–out proposals. The article DYNAMIC INTERPRETATION also provides further information.

References

[1] Barwise J 1981 'Scenes and Other Situations', *The Journal of Philosophy*, 78, 369–397.

[2] Barwise J 1987 'Noun Phrases, Generalized Quantifiers and Anaphora', in P Gärdenfors (ed.), *Generalized Quantifiers / Linguistic and Logical Approaches*, 1–29, Reidel, Dordrecht.

[3] Barwise J & R Cooper 1981 'Generalized Quantifiers and Natural Language', *Linguistics and Philosophy*, 4, 159-219.

[4] Church A 1940 'A Formulation of the Simple Theory of Types', *Journal of Symbolic Logic*, 5, 56–68.

[5] Dowty D 1976 *Word Meaning and Montague Grammar*, Reidel, Dordrecht.

[6] Dowty D R, R E Wall and S Peters 1981, *Introduction to Montague Semantics*, Reidel, Dordrecht.

[7] Frege G 1879 *Begriffsschrift*, Jena.

[8] Geach P T 1962 *Reference and Generality / an Examination of some Medieval and Modern Theories*, Cornell University Press, Ithaca & London (third revised edition: 1980).

[9] Groenendijk J & M Stokhof 1984 *Studies on the Semantics of Questions and the Pragmatics of Answers*, Ph. D. Thesis, University of Amsterdam.

[10] Groenendijk J & M Stokhof 1988 'Type Shifting Rules and the Semantics of Interrogatives', in G Chierchia, B H Partee & R Turner (eds.), *Properties, Types and Meaning, Volume II*, 21–68, Kluwer, Dordrecht.

[11] Groenendijk J & M Stokhof 1990 'Dynamic Montague Grammar', in L Kálman & L Pólos (eds.), *Papers from the Second Symposium on Logic and Language*, 3–48, Akadémiai Kiadó, Budapest.

[12] Hintikka J 1976 *The Semantics of Questions and the Question of Semantics*, Acta Philosophica Fennica, 28, North Holland, Amsterdam.

[13] Kamp H 1981 'A Theory of Truth and Semantic Representation', in Groenendijk, Janssen, Stokhof (eds.), *Formal Methods in the Study of Language*, Mathematisch Centrum, Amsterdam

[14] Lewis D 1970 'General Semantics', *Synthese*, 22, 18–67.

[15] Montague R 1973 'The Proper Treatment of Quantification in Ordinary English', in Hintikka e.a. (eds.), *Approaches to Natural Language*, Reidel, Dordrecht, 221–242.

[16] Muskens R 1989 *Meaning and Partiality*, Ph D Thesis, University of Amsterdam.

[17] Muskens R 1990 'Meaning, Context and the Logic of Change', manuscript, Tilburg University.

[18] Partee B H 1975 'Montague Grammar and Transformational Grammar', *Linguistic Inquiry*, 6, 203–300

[19] Partee B H (ed) 1975 *Montague Grammar*, Academic Press, New York, San Francisco and London.

[20] Percy W 1954 *The Message in the Bottle*, Farrar, Straus and Giroux, New York.

[21] Pereira F & S Shieber 1985 *Prolog and Natural Language Analysis*, CSLI Lecture Notes 10, CSLI, Stanford (distributed by University of Chicago Press).

[22] Tarski A 1956 'The Concept of Truth in Formalized Languages', in A. Tarski, *Logic, Semantics, Metamathematics*, Oxford University Press, Oxford, 152–278 (original Polish publication in 1934).

[23] Van Benthem J 1986 *Essays in Logical Semantics*, Reidel, Dordrecht.

[24] Van Benthem J 1990 'General Dynamics', *Theoretical Linguistics*.