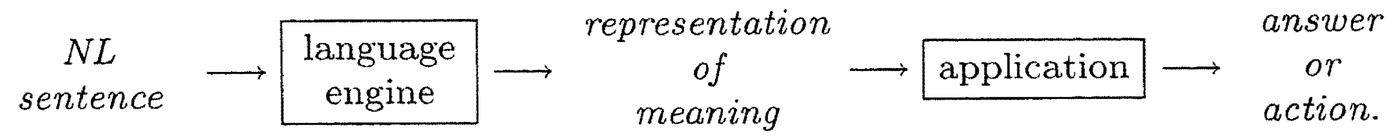
On the Borderline of Logic, Language and Computation

D.J.N. van Eijck

1. What is Computational Linguistics?

Formal language theory, as a branch of computer science, is concerned with the study of formal languages such as *Pascal*, *Prolog*, *LISP*, and various other formalisms designed for human–computer interaction. It has become clear in the past two decades, however, that the tools developed for the analysis of formal languages can fruitfully be applied to the study of natural languages such as English, Russian or Dutch. Recently, a new branch of linguistics has emerged which uses insights from formal language theory, empirical linguistics, and logic, with the overall aim to implement natural language understanding systems on computers. This new discipline is called computational linguistics.

Here is a schematic view of a typical system for human–computer interaction by means of natural language:



A system which supports voice interaction would have extra components for speech recognition and generation. In the scheme above these components are omitted. If written input is assumed, a natural language engine for English should be able to recognize a substantial fragment of grammatical

written English sentences, and generate their literal meanings; it should also be able to generate grammatical English expressions on the basis of symbolic inputs from an application program. In the scheme it is assumed that the answers given by the system are displayed as English expressions on the screen. The language engine should contain an extensive lexicon for English, a set of grammar rules for a considerable fragment of English, and a set of translation rules matching the grammar rules for translating the English expressions into unambiguous formal expressions that represent their meanings and that can be handled by the application [1]. Typical applications are expert systems or knowledge bases.

2. FRAGMENTS OF ENGLISH

Probably the best way of forming an accurate picture of what is involved in natural language processing is to develop a toy natural language application. In this contribution we aim to give a basic idea of some aspects of designing a system for natural language understanding by developing a very simple toy fragment of natural language. The method of fragments has been first advocated in the work of R. Montague [5]. Our fragment will be a lot simpler still than the simplest Montague grammar fragment. We want to be able to process sentences like the following:

A woman smiles. (2.1)

John loves a woman. (2.2)

If a woman smiles, John loves her. (2.3)

These examples may seem embarrassingly simple, but they are not quite as trivial as may appear at first sight. Example (2.3) exhibits a logical puzzle that has bothered natural language researchers for a long time. If one compares examples (2.1) and (2.2) with (2.3) then it appears that the first two can be understood as statements about a particular woman, while the third seems to express a general statement about women. This poses a genuine problem for natural language understanding, for the process of building meaning representations for sentences has to proceed in a *compositional* fashion: the meaning of a complex expression of natural language is built from the meanings of its components. This compositionality requirement is one of the cornerstones of the enterprise of building meaning representations in a systematic way.

As the example sentences indicate, indefinite noun phrases such as *a woman* seem to require a meaning representation as existential expressions when they appear in simple contexts and a representation as universal expressions when they appear in the antecedents of *if then* contexts. A first attempt at solving this problem would use ordinary predicate logic. Take example (2.4).

If a woman smiles, John smiles. (2.4)

Here one can translate the noun phrase *a woman* using an existential quantifier and still get the right meaning: $\exists x(Wx \wedge Sx) \rightarrow Sj$. To the logician it is clear immediately that this expresses a universal statement: the existential quantifier occurs in a negative position, so the translation is equivalent to $\forall x((Wx \wedge Sx) \rightarrow Sj)$. Unfortunately, this observation does not work for example (2.3). A straightforward translation of (2.3) in predicate logic would yield $\exists x(Wx \wedge Sx) \rightarrow Ljx$. But this cannot be correct, for the variable x in Ljx is left unbound.

A systematic solution of this unbound variable problem becomes possible if one translates to a representation language where variable binding proceeds in a dynamic fashion, as in imperative programming languages; such a move was first proposed in [2]. To make this work one has to replace existential quantifiers with indeterministic instructions for storing values with specified properties in the memory locations associated with variables. So instead of $\exists x(Wx \wedge \ldots)$ one writes $\eta x : Wx; \ldots$, with the intended meaning: 'fill location x with (a representation of) an object satisfying W and proceed with the ... processing'. The switch to dynamic interpretation will make it necessary to relate dynamic meaning representations to the old fashioned static representations that can be expressed in predicate logic, for example. This problem has been addressed in the research in natural language analysis at CWI [4].

What we will do in the remainder of this contribution is give a very rudimentary sketch of the syntactic processing of natural language by means of a tool called *categorial grammar*. We will then show how the syntactic analysis can be used to build meaning representations in a dynamic representation language. Finally, we will discuss the problem of relating the dynamic representations to static representations phrased in ordinary predicate logic.

3. Categorial Grammar with Features

A categorial grammar is a grammar combined with a lexicon in such a way that the lexical information comprises virtually all the information one needs for syntactic processing. These grammars are called categorial because they proceed by assigning *categories* to expressions. Simple categories such as S for 'sentence', CN for 'common noun' and IV for 'intransitive verb' are taken as basic.

For our fragment we have the following expressions in basic categories: *smiles:*IV, *man:*CN and *woman:*CN. Next, a proper name can be viewed as an expression which combines with an intransitive verb to its right to form a sentence; in categorial notation: S/IV. So for our fragment: *John:*S/IV. The

indefinite article combines with a common noun to form a noun phrase; noun phrases, as we have seen, have category *S/IV*, so we have: *a:(S/IV)/CN*. Transitive verbs combine with noun phrases to form intransitive verbs, which gives: *loves:IV/(S/IV)*. Finally, the sentential operator *if* takes an antecedent sentence and forms an expression which combines with a consequent sentence to form a new sentence, which gives: *if:(S/S)/S*.

The basic strategy for putting categorial expressions together is very simple: put an expression of category CAT_1/CAT_2 in front of an expression of category CAT_2 to form a new expression of category CAT_1. Thus, putting *John:(S/IV)* in front of *smiles:IV* gives *John smiles : S*, and so on.

Of course, this grammar is too simple as it stands. For example, the pronoun *her* will have the same category as any other noun phrase, namely *S/IV*, and this would enable the derivation of *her smiles : S*. To remedy this, it is customary to enrich the categories with *feature information*. For instance, if one assumes that a noun phrase has features for case, gender, number and a coreference index, the category for *her* could look like *(S/IV):[-nom,f,sg,213]*, to indicate that its case is not nominative, its gender is feminine, its number is singular, and it has coreference index 213 (which means that it is intended to be linked to an antecedent which also has index 213). This information can be used to enrich the category of noun phrases to force agreement in case and number with an intransitive verb phrase, as follows : *(S/IV:[Case,Number]):[Case,_ ,Number,_]*. The upper case letters are used to indicate feature constraints; _ indicates that any value will do for the feature at that position. If an item of this category combines with an *IV* with given features for case and number, then these features should agree. Enriching the category of *walks* to *IV:[nom,sg]* now blocks the derivation of *her walks*. It is possible to encode very complex and detailed information in syntactic features.

Toy Categorial Grammar

John_i	(S/IV[Case,sg]):[Case,sg,m,i]
her_i	(S/IV:[-nom,sg]):[-nom,f,sg,i]
smiles	IV:[nom,sg]
loves	(IV:[nom,sg]/(S/IV)):[-nom,_ ,_ ,_]
man	CN:[sg,m]
woman	CN:[sg,f]
a_i	((S/IV):[_ ,sg,Gender,i]/CN:[sg,Gender]
if	(S/S)/S

In the next section we will present a logical perspective on the process of parsing. In Section 5 we will see how a procedure for building meaning representations can be hooked to a categorial grammar.

4. Parsing as Deduction

We can look at the process of parsing natural language sentences as a kind of deduction. A rule of inference of a parsing algorithm has the following general form:

$$\frac{A_1 \ \cdots \ A_n}{B} \quad \textit{side conditions on } A_1, \ldots, A_n, B \tag{4.1}$$

A deduction system is given by a set of such rules, plus a set of axioms. A derivation of a formula B from premisses $A_1, \ldots, A_n$ is defined as usual: a sequence of formulas with B at the end, and each member of the sequence is either an axiom or the result of applying a deduction rule to previous members of the sequence. In the parsing application, we assume that formulas may refer to positions in the input string, and that derivation rules may mention grammar rules in their side conditions. In the case of parsing categorial grammars, items have the form $[X, i, j]$, where X is a grammar category and i, j refer to positions in the input string. The intended meaning is: category X spans word sequence $w_i, \ldots, w_j$ in the input string. Axioms have the form $[X, i, i+1]$, where X is a category assigned by the lexicon to word w_{i+1} in the input string. If the input string has length n, the parsing goal has the form $[S, 0, n]$. The inference rule looks as follows:

$$\frac{[X/Y, i, j] \ [Y, j, k]}{[X, i, k]} \tag{4.2}$$

John loves a woman. (4.3)

The proof that (4.3) is a sentence of the fragment is given in the box below. More information on parsing as deduction can be found in [6].

Proof of sentence in the fragment.

1.	[S/IV, 0, 1]	axiom
2.	[IV/(S/IV),1 , 2]	axiom
3.	[(S/IV)/CN,2 , 3]	axiom
4.	[CN, 3, 4]	axiom
5.	[S/IV, 2, 4]	rule application on 3, 4
6.	[IV, 1, 4]	rule application on 2, 5
7.	[S, 0, 4]	rule application on 1, 6

5. Building Meaning Representations

If a categorial grammar is given, meaning representations for the expressions recognized by the grammar can be given using the tools of lambda abstraction. The semantic operation corresponding to the syntactic combination of an expression of category CAT_1/CAT_2 and one of category CAT_2

will be the functional application of a typed lambda expression corresponding to the functor expression to a typed lambda expression corresponding to the argument. The basic categories provide the clue for the types of the translations: S expressions should translate as formulae, IV and CN expressions as one-place predicates. Thus, an appropriate translation for *woman* is $\lambda x.Wx$, which does indeed denote a one place predicate. Similarly for IV expressions: *smiles* can be translated as $\lambda x.Sx$. Expressions of category S/IV must be lambda expressions that can take things like $\lambda x.Sx$ as arguments, so an appropriate translation for *John* would be $\lambda X.Xj$, where X is a variable for one-place predicates. Given that we want to translate *a woman* as $\eta x : Wx; \ldots$, an appropriate translation for this noun phrase is $\lambda Y.\eta x : Wx; Yx$, which means that *a* can be translated as $\lambda X \lambda Y.\eta x : Xx; Yx$, and so on. If one wants to use the noun phrase indices to establish links of pronouns to their antecedents, individual variables with the same index must be used in the translations. These considerations are all taken into account in the example grammar with semantic component below.

Toy Categorial Grammar with Semantic Component

John$_i$	(S/IV[Case,sg]):[Case,sg,m,i]	$\lambda X.(\eta v_i : v_i = j; Xv_i)$
her$_i$	(S/IV:[-nom,sg]):[-nom,f,sg,i]	$\lambda X.Xv_i$
smiles	IV:[nom,sg]	$\lambda x.Sx$
loves	(IV:[nom,sg]/(S/IV):[-nom,_ ,_ ,_]	$\lambda \mathbf{X} \lambda x.\mathbf{X}(\lambda y.Lxy)$
man	CN:[sg,m]	$\lambda x.Mx$
woman	CN:[sg,f]	$\lambda x.Wx$
a$_i$	((S/IV):[_ ,sg,Gender,i]/CN:[sg,Gender]	$\lambda X \lambda Y.(\eta v_i : Xv_i; Yv_i)$
if	(S/S)/S	$\lambda p \lambda q.(p \Rightarrow q)$

Our fragment enables us to construct meaning representations for the example sentences we started out with. The representation for *John smiles* becomes $\lambda X.(\eta v_i : v_i = j; Xv_i)(\lambda x.Sx)$, which reduces in two steps to $\eta v_i : v_i = j; Sv_i$. The representation for *John loves a woman* is a fairly complex expression which reduces in several steps to $\eta v_i : v_i = j; \eta v_k : Wv_k; Lv_iv_k$. We have assumed that the indices of subject and object are different; in fact, a procedure for checking co-indexings should be invoked to rule out all co-indexings with clashes in the gender or number feature. The representation for *If a woman smiles, John loves her*, in the reading where the pronoun is linked to *a woman*, will, after several reductions, boil down to $(\eta v_i : (Wv_i; Sv_i) \Rightarrow \eta v_k : (v_k = j; Lv_kv_i))$. Our final problem is to make sense of this representation.

6. Axioms for Dynamic Interpretation

The dynamic interpretation strategy treats meaning representations for natural language as imperative programs. This entails that the tools for anal-

ysis of imperative programming languages can be put to use. In particular, an axiom system for dynamic interpretation can be given in terms of Hoare style pre- and postconditions of programs [3]. A convenient way to express such conditions is by means of dynamic logic [?]. Axioms for dynamic interpretation have natural language meaning representations as modalities. $\langle\pi\rangle\phi$ expresses that ϕ holds in some π output state of the current state; $[\pi]\phi$ expresses that ϕ holds in every π output state of the current state. We can find the static meaning of a representation program by checking the conditions under which it will terminate successfully. Some examples of the dynamic logic axioms involved are given below; more details can be found in Van Eijck's contribution in [4].

Axioms for Dynamic Interpretation

$$
\begin{array}{lcl}
\langle R(t_1 \cdots t_n)\rangle\phi & \leftrightarrow & Rt_1 \cdots t_n \wedge \phi \\
\langle t_1 = t_2\rangle\phi & \leftrightarrow & t_1 = t_2 \wedge \phi \\
\langle \pi_1;\pi_2\rangle\phi & \leftrightarrow & \langle\pi_1\rangle\langle\pi_2\rangle\phi \\
\langle \pi_1 \Rightarrow \pi_2\rangle\phi & \leftrightarrow & [\pi_1]\langle\pi_2\rangle\top \wedge \phi \\
\langle \eta x : \pi\rangle\phi & \leftrightarrow & \exists x\langle\pi\rangle\phi
\end{array}
$$

Using the axioms, the static meaning of the representation (6.4) can be derived by a simple calculation.

$$\eta v_i : (Wv_i; Sv_i) \Rightarrow \eta v_k : (v_k = j; Lv_k v_i). \tag{6.4}$$

This turns out to be (6.5), which is indeed an appropriate meaning representation for example sentence (2.3).

$$\forall v_i(Wv_i \rightarrow (Sv_i \rightarrow \exists v_k(v_k = j \wedge Lv_k v_i))). \tag{6.5}$$

7. Conclusion

Natural language understanding research at CWI concentrates on theoretical issues and emphasises the use of tools from programming language analysis for the analysis of natural language. It is expected, however, that the insights thus gained will greatly facilitate the task of building practically useful natural language interfaces in the not too distant future.

References

1. H. Alshawi (ed.) (1992). *The Core Language Engine*. MIT Press, Cambridge Mass, Cambridge, Mass., and London, England.

2. J. BARWISE (1987). Noun phrases, generalized quantifiers and anaphora. P. GÄRDENFORS (ed.), *Generalized Quantifiers: linguistic and logical approaches*, Reidel, Dordrecht, 1-30.

3. J. VAN EIJCK, F.J. DE VRIES (1992). Dynamic interpretation and Hoare deduction. *Journal of Logic, Language, and Information 1*, 1-44.

4. J. VAN EIJCK, A. VISSER (eds.) (1994). *Logic and Information Flow.* MIT Press, Cambridge Mass.

5. R. MONTAGUE (1973). The proper treatment of quantification in ordinary English. J. HINTIKKA E.A. (ed.), *Approaches to Natural Language*, Reidel, 221-242.

6. S.M. SHIEBER, Y. SCHABES, F.C.N. PEREIRA (1995). Principles and implementation of deductive parsing. *Journal of Logic Programming 1&2*, 3-56.