λ-Grammars and the Syntax-Semantics Interface *

Reinhard Muskens

In this paper we discuss a new perspective on the syntax-semantics interface. Semantics, in this new set-up, is not 'read off' from Logical Forms as in mainstream approaches to generative grammar. Nor is it assigned to syntactic proofs using a Curry-Howard correspondence as in versions of the Lambek Calculus, or read off from f-structures using Linear Logic as in Lexical-Functional Grammar (LFG, Kaplan & Bresnan [9]). All such approaches are based on the idea that syntactic objects (trees, proofs, f-structures) are somehow prior and that semantics must be parasitic on those syntactic objects. We challenge this idea and develop a grammar in which syntax and semantics are treated in a strictly parallel fashion. The grammar will have many ideas in common with the (converging) frameworks of categorial grammar and LFG, but its treatment of the syntax-semantics interface is radically different. Also, although the meaning component of the grammar is a version of Montague semantics and although there are obvious affinities between Montague's conception of grammar and the work presented here, the grammar is not compositional, in the sense that composition of meaning need not follow surface structure.

λ-Grammars

We follow the tradition of Curry [6], Cresswell [4], and Oehrle [13, 14, 15] in representing syntactic information with the help of typed λ-terms.[1] For example, Oehrle [13, 14] considers multidimensional signs such as those in (1),

*From: Robert van Rooy and Martin Stokhof, editors, *Proceedings of the Thirteenth Amsterdam Colloquium*, University of Amsterdam, Amsterdam, 2001, pages 150–155.

[1] Curry considers expressions containing subscripted blanks, such as '$—_1$ is between $—_2$ and $—_3$' or '$—_1$ were eaten by the children'. Functors can apply to arguments and arguments are to be substituted for blanks in the order of the subscripts. Essentially

consisting of a λ-term over strings, a semantic λ-term, and a type. (1a) can be combined with (1b) by applying (1a)'s first term to (1b)'s first term, applying (1a)'s second term to (1b)'s second term, and applying Modus Ponens to the types. The result is (1c).

(1) a. λxλy. y likes x : $\lambda x \lambda y.like(y, x) : np \rightarrow (np \rightarrow s)$

b. John : $j : np$

c. λy. y likes John : $\lambda y.like(y, j) : np \rightarrow s$

Oehrle's work will be our point of departure, but we deviate from it in two respects. First, while [13, 14] combine signs such as the ones in (1) with the help of the undirected Lambek Calculus, our signs will be combined using linear combinators. Mathematically this boils down to the same thing, as proofs in the undirected Lambek Calculus are in 1-1 correspondence with the latter (Van Benthem [1, 2, 3]). But a move from proofs to combinators spares the working linguist much technical overhead and it will serve our purpose to stress the point that semantics need not be dependent on any form of syntax. A second divergence from Oehrle's work is that we move from λ-terms over *structures* (strings in [13, 14], but also trees and f-structures in [15]) to λ-terms over *descriptions* of structures. This is in accordance with a general tendency in linguistics (starting with [9]) to replace structures with descriptions of structures as the prime vehicles of representation.

Consider the three 3-dimensional signs in (2).

(2) a. $\mathsf{john} : \lambda f.arc(f, cat, N) \wedge arc(f, num, sg) \wedge arc(f, pers, 3) : john$

b. $\mathsf{mary} : \lambda f.arc(f, cat, N) \wedge arc(f, num, sg) \wedge arc(f, pers, 3) : mary$

c. $\lambda t_1 \lambda t_2.[t_2\ [\mathsf{loves}\ t_1]] :$
$\lambda F_1 \lambda F_2 \lambda f \exists f_1 f_2 [F_1(f_1) \wedge F_2(f_2) \wedge arc(f, cat, V) \wedge arc(f, tense, pres) \wedge$
$arc(f_1, cat, N) \wedge arc(f, obj, f_1) \wedge arc(f_2, cat, N) \wedge arc(f_2, num, sg) \wedge$
$arc(f_2, pers, 3) \wedge arc(f, subj, f_2)] :$
$\lambda x \lambda y \lambda i.love(y, x, i)$

---

then, although Curry does not explicitly mention this, these terms are lambda terms over syntactic objects.

syntax:	$k, f : \nu$	$t, F : \nu t$	$T, \mathcal{F} : (\nu t)(\nu t)$	
semantics:	$x, y : e$	$i, j : s$	$p : st$	$P : e(st)$

Table 1: Typographical conventions for variables used in this paper. *Var* : *Type* means that *Var* (with or without subscripts or superscripts) always has type *Type*.

These signs each consist of a c-structure component, an f-structure component, and a semantic component. Expressions in sans serif in the c-structure terms are of type νt, and denote sets of nodes. For example, john can be thought of as the set of nodes that are labeled 'John', whereas an expression such as [loves mary] can be thought of as the set of nodes k directly dominating a node k_1 labeled 'loves' and a node k_2 labeled 'Mary', with k_1 preceding k_2.

The f-components of our signs consist of λ-terms over the first order feature language of Johnson [8] and the semantics in the third component is in accordance with a streamlined form of Montague's [10] theory. Constants *john* and *mary* are of type e and *love* is of type $e(e(st))$. Constants *cat, num, pers*, etc. are of a type a (attributes), while N, *sg*, 3, ... are of type ν (nodes). More typing information is given in Table 1. We consider a grammar with three dimensions here, but in general the number of dimensions of a grammar is arbitrary (though fixed). The terms that we are interested in are all closed and we require that lexical elements have closed terms in each dimension.

Signs can be combined by means of *pointwise application*. In general, if $M = \langle M_1, \ldots, M_n \rangle$ and $N = \langle N_1, \ldots, N_n \rangle$ are sequences of λ-terms such that $M_i(N_i)$ is well-typed for each i, the pointwise application of M to N is just

$$\langle M_1(N_1), \ldots, M_n(N_n) \rangle \ .$$

Generalizing the notation for application, we denote this as $M(N)$. It is easily seen that the result of pointwise application of (2c) to (2a) equals (3a) modulo standard equivalences and that (3a)((2b)) reduces to (3b).

(3) a. $\lambda t_2.[t_2\ [\textsf{loves john}]] :$
$\lambda F_2 \lambda f \exists f_1 f_2 [F_2(f_2) \wedge arc(f, cat, V) \wedge arc(f, tense, pres) \wedge arc(f_1, cat, N)$
$\wedge\ arc(f, obj, f_1) \wedge arc(f_2, cat, N) \wedge arc(f_2, num, sg) \wedge arc(f_2, pers, 3) \wedge$
$arc(f, subj, f_2)] :$
$\lambda y \lambda i.love(y, john, i)$

b. [mary [loves john]] :
$\lambda f \exists f_1 f_2 [arc(f, cat, V) \wedge arc(f, tense, pres) \wedge arc(f_1, cat, N) \wedge$
$arc(f, obj, f_1) \wedge arc(f_2, cat, N) \wedge arc(f_2, num, sg) \wedge arc(f_2, pers, 3) \wedge$
$arc(f, subj, f_2)] :$
$\lambda i.love(mary, john, i)$

The three descriptions in sentential signs such as (3b) each denote a set in every possible model of the language; the first two denote sets of nodes (type νt), the third a set of possible worlds (a proposition, type st). The idea is that if the second set is non-empty in some model of the first four axioms in [8], then any node satisfying the first description should be connected to the truth conditions expressed in the third element. The requirement that the second component should be satisfiable provides for a subcategorization mechanism. E.g., combining (3a) with a plural subject would have led to an f-description that can only denote the empty set.

In (4) and (5) some more lexical signs are given with two results of their possible combinations in (6).

(4) a. man : $\lambda f.arc(f, cat, N) \wedge arc(f, num, sg) \wedge arc(f, pers, 3) : man$

b. woman : $\lambda f.arc(f, cat, N) \wedge arc(f, num, sg) \wedge arc(f, pers, 3) : woman$

(5) a. $\lambda t \lambda T.T([\text{a } t]) :$
$\lambda F \lambda \mathcal{F}.\mathcal{F}(\lambda f.F(f) \wedge arc(f, cat, N) \wedge arc(f, num, sg) \wedge arc(f, pers, 3)) :$
$\lambda P' P \lambda i \exists x [P'(x)(i) \wedge P(x)(i)]$

b. $\lambda t \lambda T.T([\text{every } t]) :$
$\lambda F \lambda \mathcal{F}.\mathcal{F}(\lambda f.F(f) \wedge arc(f, cat, N) \wedge arc(f, num, sg) \wedge arc(f, pers, 3)) :$
$\lambda P' P \lambda i \forall x [P'(x)(i) \rightarrow P(x)(i)]$

(6) a. $\lambda T.T([\text{every man}]) :$
$\lambda \mathcal{F}.\mathcal{F}(\lambda f.arc(f, cat, N) \wedge arc(f, num, sg) \wedge arc(f, pers, 3)) :$
$\lambda P \lambda i \forall x [man(x, i) \rightarrow P(x)(i)]$

b. $\lambda T.T([\text{a woman}]) :$
$\lambda \mathcal{F}.\mathcal{F}(\lambda f.arc(f, cat, N) \wedge arc(f, num, sg) \wedge arc(f, pers, 3)) :$
$\lambda P \lambda i \exists x [woman(x, i) \wedge P(x)(i)]$

abstract type	syntactic dimensions	semantic dimension
S	νt	st
NP	νt	e
N	νt	$e(st)$

Table 2: Concretizations of abstract types used in this paper.

The terms that our signs consist of are typed, but it is expedient to type the signs themselves as well. Types for signs will be called *abstract types*. Abstract types in this paper are built up from ground types S, NP and N with the help of implication, and thus have forms such as NP S, N((NP S)S), etc. A restriction on signs is that a sign of abstract type A should have a term of type A^i in its i-th dimension. The values of the function $.^i$ for ground types can be chosen on a per grammar basis and in this paper are as in Table 2. For complex types, the rule is that $(AB)^i = A^i B^i$. This means, for example, that $\textsc{np}(\textsc{np}\ \textsc{s})^1 = \textsc{np}(\textsc{np}\ \textsc{s})^2 = (\nu t)((\nu t)\nu t)$ and that $\textsc{np}(\textsc{np}\ \textsc{s})^3 = e(e(st))$. As a consequence, (2c) should be of type NP(NP S). Similarly, (2a) and (2b) can be taken to be of type NP, (3a) and (3b) are of types NP S and S respectively, etc. In general, if M has abstract type AB and N abstract type A, then the pointwise application $M(N)$ is defined and has type B.

Abstraction can also be lifted to the level of signs. Supposing that the variables in our logic have some fixed ordering and that the number of dimensions of the grammar under consideration is n, we define the k-th n-dimensional variable ξ of abstract type A as the sequence of variables $\langle \xi_1, \ldots, \xi_n \rangle$, where each ξ_i is the k-th variable of type A^i. The *pointwise abstraction* $\lambda \xi M$ is then defined as $\langle \lambda \xi_1 M_1, \ldots, \lambda \xi_n M_n \rangle$. A definition of *pointwise substitution* is left to the reader.

With the definitions of pointwise application, pointwise abstraction, and n-dimensional variable in place, we can consider complex terms built up with these constructions. (7a), for example, is the pointwise application of (6b) to the pointwise composition of (6a) and (2c). Here ζ is of type NP. (7a) can be expanded to (7b), where each dimension of a lexical sign is denoted with the help of an appropriate subscript (e.g. $(6b)_1$ is $\lambda T.T([\mathsf{a\ woman}])$). The terms here can be reduced and the result is as in (7c), a sign coupling the c-description in its first dimension to one of its possible readings. The other reading is obtained from (7d), which reduces to (7e).

(7) a. $(6b)(\lambda\zeta.(6a)((2c)(\zeta)))$

b. $(6b)_1(\lambda\zeta_1.(6a)_1((2c)_1(\zeta_1))) :$
$(6b)_2(\lambda\zeta_2.(6a)_2((2c)_2(\zeta_2))) :$
$(6b)_3(\lambda\zeta_3.(6a)_3((2c)_3(\zeta_3)))$

c. [[every man] [loves [a woman]]] :
$\lambda f \exists f_1 f_2[arc(f, cat, V) \wedge arc(f, tense, pres) \wedge arc(f_1, cat, N) \wedge$
$arc(f, obj, f_1) \wedge arc(f_2, cat, N) \wedge arc(f_2, num, sg) \wedge arc(f_2, pers, 3) \wedge$
$arc(f, subj, f_2)] :$
$\lambda i \exists y[woman(y, i) \wedge \forall x[man(x, i) \rightarrow love(x, y, i)]]$

d. $(6a)(\lambda\zeta_2.(6b)(\lambda\zeta_1.(2c)(\zeta_1)(\zeta_2)))$

e. [[every man] [loves [a woman]]] :
$\lambda f \exists f_1 f_2[arc(f, cat, V) \wedge arc(f, tense, pres) \wedge arc(f_1, cat, N) \wedge$
$arc(f, obj, f_1) \wedge arc(f_2, cat, N) \wedge arc(f_2, num, sg) \wedge arc(f_2, pers, 3) \wedge$
$arc(f, subj, f_2)] :$
$\lambda i \forall x[man(x, i) \rightarrow \exists y[woman(y, i) \wedge love(x, y, i)]]$

Let us call terms such as (7a) and (7d), which are built up from lexical signs with the help of n-dimensional variables, pointwise application and abstraction, *n-terms*. It is worth to note that n-terms are subject to the laws of α, β, and η-conversion, i.e. reasoning with them is as usual. But clearly, not every n-term makes for an acceptable coupling between syntax and semantics. We restrict ourselves to *linear combinations* of lexical elements. These are n-terms that are closed and conform to the condition that every abstractor $\lambda\zeta$, with ζ an n-dimensional variable, binds exactly one free ζ. n-terms conforming to this condition are called *generated signs*.[2] Conditions such as the requirement that the third component of a generated sign must be satisfiable are *admissibility* conditions and a generated sign obeying them is called *admissable*.

Multidimensional grammars that are set up in the way sketched here, with λ-terms in each dimension of the grammar and linear combination as a generative device, will be called *λ-grammars*.

Since any n-term M obeys the usual laws of λ-conversion, it can be written in the form $C(L_1)\cdots(L_m)$, where $L_1, \ldots, L_m$ are lexical signs and C is an n-term that does not contain any lexical material. If M is closed, C is a multi-dimensional (and typed) variant of a *combinator* in the sense of [5]. In case

[2]Note that any linear combination of generated signs is itself a generated sign.

M is a generated sign, C will correspond to a *linear* (or **BCI**) combinator. For example, (7a) can be rewritten as (8), with $\lambda Q_1 \lambda R \lambda Q_2 . Q_1(\lambda \zeta . Q_2(R(\zeta)))$ playing the role of the linear combinator combining (6b), (2c), and (6a).

(8) $\lambda Q_1 \lambda R \lambda Q_2 . Q_1(\lambda \zeta . Q_2(R(\zeta)))((6b))((2c))((6a))$

From the fact that linear combinators play an important underlying role we see that λ-grammars have obvious affinities not only with LFG and Lambek Categorial Grammar, but also with Combinatory Categorial Grammar (see e.g. [17, 18]). But λ-grammars should be distinguished from standard categorial grammars in that they are non-directional and do not use derivations.

Conclusion

It is a wideheld belief among linguists that semantics must in some sense be dependent upon syntax. Syntax first provides a scaffolding and semantics then follows the syntactic set-up, computing the meaning of a complex expression from the meanings of its syntactic parts. Such a compositional scheme works fairly well for English and related languages, although even in Montague's pivotal work it had to be assumed that a sentence can have as its parts (a) a noun phrase deep within that sentence and (b) the sentence lacking that noun phrase.

In Dalrymple et al. [7] it was observed that a problem arises when non-configurational languages such as Warlpiri are considered. In these languages combinations of words may form a semantic unit altough they are no syntactic unit. An example is (9) (see Simpson [16]), where the adjective *wita-jarra-rlu* is not adjacent to the noun (*kurdu-jarra-rlu*) that it modifies. The constituent structure of this sentence (and indeed of many Warlpiri sentences) is flat, with one S node dominating all preterminals.[3]

(9) Kurdu-jarra-rlu ka-pala maliki wajili-pi-nyi wita-jarra-rlu
child-dual-ergative pres-3ds dog chase-nonpast small-dual-ergative
'Two small children are chasing the dog', or
'Two children are chasing the dog and they are small'

Since *wita-jarra-rlu* and *kurdu-jarra-rlu* do not form a syntactic constituent but should still be considered a semantic whole, the idea of erecting a syntactic scaffold first and then using it in semantics breaks down. Constituent

[3]Any permutation of the words in (9) that leaves the 'auxiliary' element *ka-pala* in second position is also an acceptable Warlpiri sentence with the same two meanings.

structure simply does not provide enough structure for interpretation. The case is typical for a wide class of languages.

In [7] this observation motivates a development in which semantics is read off from functional structure with the help of the $\{-\circ, \otimes\}$ fragment of intuitionistic Linear Logic (= the undirected Lambek Calculus). In this paper we have taken the more radical course of using linear combinators directly for combining syntactic / semantic signs. This simplifies the grammatical set-up, as in our approach there is no need for a phrase structure component as a separate generative engine. Linear combinations suffice. The toy grammar presented here shows that it is possible to do syntax and semantics really in tandem. Interpretation does not need any previous syntactic scaffolding, whether it be a constituent structure, a functional structure, a Lambek / Linear Logic proof, or any other syntactic structure. There is no space here to give a detailed analysis of (9), but an essential element should be that when the sign of an adjective is combined with the sign of a noun, the result in the c-dimension need not be a description of an adjective-noun constituent. A description requiring sisterhood suffices.

The grammar in this paper also shows that it is possible to import many ideas of Lexical-Functional Grammar into an essentially categorial framework. In this we build upon Oehrle [15]. That our move from structures to descriptions allows the incorporation of more ideas from LFG (constraining equations, path constraints for long distance dependencies) is shown in Muskens [12]. A development of the theory that is more geared towards categorial grammar and the multimodal enterprise can be found in Muskens [11].

References

[1] J.F.A.K. van Benthem. *Essays in Logical Semantics*. Reidel, Dordrecht, 1986.

[2] J.F.A.K. van Benthem. The Semantics of Variety in Categorial Grammar. In W. Buszkowski, W. Marciszewski, and J.F.A.K. van Benthem, editors, *Categorial Grammar*, pages 37–55. John Benjamins, Amsterdam, 1988.

[3] J.F.A.K. van Benthem. *Language in Action*. North-Holland, Amsterdam, 1991.

[4] M.J. Cresswell. *Logics and Languages*. Methuen, London, 1973.

[5] H. Curry and R. Feys. *Combinatory Logic*, volume I. North-Holland, Amsterdam, 1958.

[6] H.B. Curry. Some Logical Aspects of Grammatical Structure. In *Structure of Language and its Mathematical Aspects: Proceedings of the Twelfth Symposium in Applied Mathematics*, pages 56–68. AMS, 1961.

[7] M. Dalrymple, J. Lamping, and V. Saraswat. LFG Semantics via Constraints. In *Proceedings of the Sixth Meeting of the European ACL*. European Chapter of the Association for Computational Linguistics, 1993.

[8] M. Johnson. Logic and Feature Structures. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991.

[9] R. Kaplan and J. Bresnan. Lexical-Functional Grammar: a Formal System for Grammatical Representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. The MIT Press, Cambridge, MA, 1982.

[10] R. Montague. The Proper Treatment of Quantification in Ordinary English. In *Formal Philosophy*, pages 247–270. Yale University Press, New Haven, 1973.

[11] R.A. Muskens. Language, Lambda's, and Logic. Submitted for publication, 2001.

[12] Reinhard Muskens. Categorial Grammar and Lexical-Functional Grammar. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of the LFG01 Conference, University of Hong Kong*, Stanford CA, 2001. CSLI Publications. http://csli-publications.stanford.edu/hand/miscpubsonline.html.

[13] R.T. Oehrle. Term-Labeled Categorial Type Systems. *Linguistics and Philosophy*, 17:633–678, 1994.

[14] R.T. Oehrle. Some 3-Dimensional Systems of Labelled Deduction. *Bulletin of the IGPL*, 3:429–448, 1995.

[15] R.T. Oehrle. LFG as Labeled Deduction. In M. Dalrymple, editor, *Semantics and Syntax in Lexical Functional Grammar*, chapter 9, pages 319–357. MIT Press, Cambridge, MA, 1999.

[16] J. Simpson. *Warlpiri Morpho-Syntax: A Lexicalist Approach*, volume 23 of *Studies in Natural Language and Linguistic Theory*. Kluwer, Dordrecht, 1991.

[17] M. Steedman. *Surface Structure and Interpretation*. MIT Press, 1996.

[18] M. Steedman. *The Syntactic Process*. MIT Press, 2000.