# Intensional Verbs Without Type-Raising or Lexical Ambiguity

Mary Dalrymple*
John Lamping*
Fernando Pereira†
Vijay Saraswat*

July 24, 2011

## 1 Introduction

We present an analysis of the semantic interpretation of intensional verbs such as *seek* that allows them to take direct objects of either individual or quantifier type, producing both *de dicto* and *de re* readings in the quantifier case, all without needing to stipulate type-raising or quantifying-in rules. This simple account follows directly from our use of logical deduction in linear logic to express the relationship between syntactic structures and meanings. While our analysis resembles current categorial approaches in important ways ((Moortgat , 1988; Moortgat , 1992a; Morrill , 1993; Carpenter , 1993)), it differs from them in allowing the greater type flexibility of categorial semantics ((van Benthem , 1991)) while maintaining a precise connection to syntax. As a result, we are able to provide derivations for certain readings of sentences with intensional verbs and complex direct objects that are not derivable in current purely categorial accounts of the syntax-semantics interface. The analysis forms a part of our ongoing work on semantic interpretation within the framework of Lexical-Functional Grammar.

## 2 Theoretical Background

As a preliminary to presenting our analysis of intensional verbs, we outline our approach to semantic interpretation in LFG.

*Xerox PARC, Palo Alto CA 94304; {dalrymple,lamping,saraswat}@parc.xerox.com
†AT&T Bell Laboratories, Murray Hill NJ 07974; pereira@research.att.com

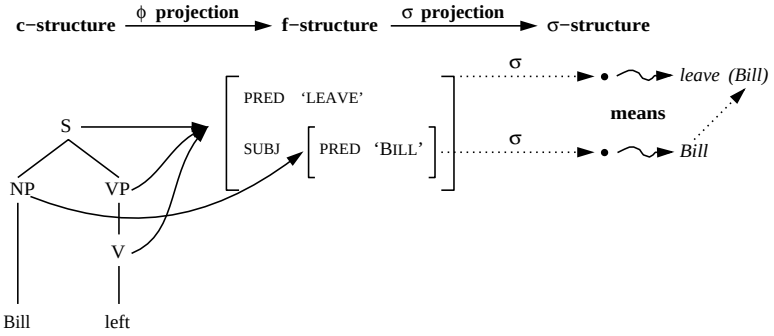


Figure 1: Semantic Interpretation Architecture

It is well known that surface constituent structure does not always provide the optimal set of constituents or hierarchical structure to guide semantic interpretation. This has led to efforts to develop more abstract structures for the representation of relevant syntactic information. We follow Kaplan and Bresnan (1982) and Halvorsen (1983) in taking the functional structure or *f-structure* of LFG as the primary input to semantic interpretation. The syntactic structures of LFG, the constituent structure or *c-structure* and the f-structure, are related by means of a functional correspondence, represented in Figure 1 by solid lines leading from nodes of the c-structure tree to f-structures ((Kaplan and Bresnan , 1982)).[1] In more recent work, Kaplan (1987) and Halvorsen and Kaplan (1988) have proposed to extend the theory of correspondences to other structures, called *projections*. Here, we will appeal to a semantic projection $\sigma$, relating f-structures and their meanings. Notationally, a subscript $\sigma$ will indicate the semantic or $\sigma$ projection of an f-structure $f$, so that the semantic projection of $f$ will be written $f_\sigma$. In Figure 1, dotted lines represent the relation between f-structures and their semantic projections. Finally, as shown in the figure, the semantic projection $f_\sigma$ of an f-structure $f$ can be put in correspondence with a meaning $\phi$:

(1) $f_\sigma \rightsquigarrow \phi$

Informally, we read this expression as "the meaning of $f$ is $\phi$". We use expressions of this sort to lexically associate meanings with f-structures, as in the following lexical entry for the word *Bill*:

(2) Bill $(\uparrow \text{PRED}) = \text{'BILL'}$
$\uparrow_\sigma \rightsquigarrow Bill$

[1] The c-structure and f-structure presented here have been simplified to show only the detail necessary for the semantic issues addressed here. We also do not address a number of orthogonal semantic issues (tense and aspect, for example), providing only enough details of the representation of the meaning of a sentence to illustrate the points relevant to the discussion at hand.

The first line of this lexical entry:

$$(\uparrow \textsc{pred}) = \text{`}\textsc{Bill}\text{'}$$

says (roughly) that the word *Bill* introduces an f-structure $\uparrow$ whose PRED is 'Bill'. The second line:

$$\uparrow_\sigma \rightsquigarrow Bill$$

says that the meaning of that f-structure is *Bill*. When a lexical entry is used, the metavariable '$\uparrow$' is instantiated with some constant $f$ denoting an f-structure ((Kaplan and Bresnan , 1982, page 183)). In particular, the term $\uparrow_\sigma$ is instantiated to the semantic projection $f_\sigma$ of $f$ and the formula $\uparrow_\sigma \rightsquigarrow Bill$ is instantiated to $f_\sigma \rightsquigarrow Bill$, asserting that the meaning of $f$ is *Bill*.

More complicated lexical entries give not only meanings for f-structures, but also instructions for assembling f-structure meanings from the meanings of other f-structures. We distinguish a *meaning language*, in which we represent the meanings of f-structures, and a *composition language* or *glue language*, in which we describe how to assemble the meanings of f-structures from the meanings of their substructures. Each lexical entry will contain a composition language formula, its *meaning constructor*, specifying how a lexical entry contributes to the meaning of any structure in which it participates.

In principle, the meaning language can be any suitable logic. Here, since we are concerned with the semantics of intensional verbs, we will use Montague's higher-order intensional logic IL. The expressions that appear on the right side of the $\rightsquigarrow$ connective in lexical entries like (2) above are (usually open) terms in the meaning language.

Our composition language is a fragment of linear logic with higher-order quantification. While the resource sensitivity of linear logic is crucial to our overall interpretation framework, it does not play a central role in the analysis discussed here, so the linear connectives can be informally read as their classical counterparts.[2] In contrast to standard approaches, which use the $\lambda$-calculus to combine fragments of meaning via ordered applications, we combine fragments of meaning through unordered conjunction and implication. Rather than using $\lambda$-reduction to simplify meanings, we rely on deduction, as advocated by Pereira ((1990; 1991)).

[2]We make use of *linear logic* ((Girard , 1987)) because its resource sensitivity turns out to be a good match to natural language. This property of linear logic nicely captures, among other things, the LFG requirements of *coherence* and *consistency*, and enables a nice treatment of modification ((Dalrymple et al. , 1993)), quantification ((Dalrymple et al. , 1994)), and complex predicates ((Dalrymple et al. , 1993)). We make use only of the *tensor fragment* of linear logic. The fragment is closed under conjunction, universal quantification and implication. It arises from transferring to linear logic the ideas underlying the concurrent constraint programming scheme of Saraswat ((1989)). A nice tutorial introduction to linear logic itself may be found in Scedrov ((1993)).

# 3 A Simple Example

We now turn to a simple example to illustrate the framework. The lexical entries necessary for the example in Figure 1 are:[3]

(3) Bill $(\uparrow \textsc{pred}) = \text{`}\textsc{Bill}\text{'}$
$\uparrow_\sigma = \mathit{Bill}$

left $(\uparrow \textsc{pred}) = \text{`}\textsc{leave}\text{'}$
$\forall X.\ (\uparrow \textsc{subj})_\sigma \rightsquigarrow X \multimap \uparrow_\sigma \rightsquigarrow \mathit{leave}(X)$

The symbol ' $\multimap$ ' is the *linear implication* operator of linear logic; for this paper, ' $\multimap$ ' can be thought of as analogous to classical implication '$\rightarrow$'. The formula

$$\forall X.\ (\uparrow \textsc{subj})_\sigma \rightsquigarrow X \multimap \uparrow_\sigma \rightsquigarrow \mathit{leave}(X)$$

states that the verb *left* requires a meaning $X$ for its subject, $(\uparrow \textsc{subj})$; when that meaning is provided, the meaning for the sentence will be $\mathit{leave}(X)$. When the words *Bill* and *left* are used in a sentence, the metavariable $\uparrow$ will be instantiated to a particular f-structure, and the meaning given in the lexical entry will be used as the meaning of that f-structure.

Here we repeat the f-structure in Figure 1, including labels for ease of reference:

(4) $$f\!:\begin{bmatrix} \textsc{pred} & \text{`}\textsc{leave}\text{'} \\ \textsc{subj} & g\!:\begin{bmatrix} \textsc{pred} & \text{`}\textsc{Bill}\text{'} \end{bmatrix} \end{bmatrix}$$

Annotated phrase structure rules provide instructions for assembling this f-structure by instantiating the metavariables '$\uparrow$' in the lexical entries above. For instance, the metavariable '$\uparrow$' in the lexical entry for *Bill* is instantiated to the f-structure labeled $g$.

From the instantiated lexical entries of *Bill* and *left*, we have the following semantic information:

(5) **leave**: $\forall X.\ g_\sigma \rightsquigarrow X \multimap f_\sigma \rightsquigarrow \mathit{leave}(X)$
**Bill**: $g_\sigma \rightsquigarrow \mathit{Bill}$

where **leave** and **Bill** are names for their respective formulas. By modus ponens, we deduce:

$$\mathbf{Bill}, \mathbf{leave} \vdash f_\sigma \rightsquigarrow \mathit{leave}(\mathit{Bill})$$

The elements of the f-structure provide a set of formulas in the composition logic that introduce semantic elements and describe how they can be combined.

[3] In the composition language, we use upper-case letters for *essentially existential* variables, that is, Prolog-like variables that become instantiated to particular terms during a derivation, and lower-case letters for *essentially universal variables* that stand for new local constants (also called eigenvariables) during a derivation.

For example, lexical items for words that expect arguments, like verbs, typically contribute a formula for combining the meanings of their arguments into a result. Once all the formulas are assembled, deduction in the logic is used to infer the meaning of the entire structure. Throughout this process we maintain a clear distinction between meanings proper and assertions about meaning combinations.

# 4 Quantification

We now turn to an overview of our analysis of quantification (Dalrymple, Lamping, Pereira, and Saraswat 1993). As a simple example, consider the sentence

(6) Every man left.

For conciseness, we will not illustrate the combination of the meaning constructors for *every* and *man*; instead, we will work with the derived meaning constructor for the subject *every man*, showing how it combines with the meaning constructor for *left* to produce a meaning constructor giving the meaning of the whole sentence.

The basic idea of our analysis of quantified NPs can be seen as a logical counterpart at the semantic composition level of the generalized-quantifier type assignment for (quantified) NPs ((Barwise and Cooper , 1981)). Under that assignment, a NP meaning $Q$ has type

$$(e \to t) \to t$$

—that is, a function from a property, the scope of quantification, to a proposition. At the semantic composition level, we can understand that type as follows. If by assuming that $x$ is the entity referred to by the NP we can derive $Sx$ as the meaning of the scope of quantification, where $S$ is a property (a function from entities to propositions), then we can derive $QS$ as the meaning of the whole clause containing the NP.

The f-structure for the sentence *Every man left* is:

(7)
$$f\!:\begin{bmatrix} \text{PRED} & \text{`LEAVE'} \\ \text{SUBJ} & g\!:\begin{bmatrix} \text{SPEC} & \text{`EVERY'} \\ \text{PRED} & \text{`MAN'} \end{bmatrix} \end{bmatrix}$$

The meaning constructors for *every man* and *left* are:[4]

[4]We use throughout the convenient abbreviation $Q(x, Rx, Sx)$ for the application of the generalized quantifier $Q$ to restriction $R$ and scope $S$.

(8) **leave**: $\forall X.\ g_\sigma \rightsquigarrow X \multimap f_\sigma \rightsquigarrow leave(X)$

**every-man**: $\forall H, S.\ (\forall x. g_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Sx) \multimap H \rightsquigarrow every(z, man(z), Sz)$

The meaning constructor for *left* is as before. The meaning constructor for *every man* quantifies over semantic projections $H$ which constitute possible quantification scopes; its propositional structure corresponds to the standard type assignment, $(e \to t) \to t$. It can be paraphrased as:

$\forall H, S.$

$(\forall x.\ g_\sigma \rightsquigarrow x$ — if, by assuming an arbitrary meaning $x$ for $g$,

$\multimap H \rightsquigarrow Sx)$ — a meaning $Sx$ for some scope $H$ can be derived,

$\multimap H \rightsquigarrow every(z, man(z), Sz)$ — then a possible complete meaning for $H$ can be derived.

In the case at hand, the semantic projection $f_\sigma$ will be chosen to provide the scope of quantification.[5] It has exactly the form that the antecedent of **every-man** expects. The meaning $S$ will be instantiated to $\lambda x.leave(x)$. From the premises in (8), we can deduce the meaning of the scope f-structure $f$:

$$\textbf{every-man}, \textbf{leave} \vdash f_\sigma \rightsquigarrow every(z, man(z), leave(z))$$

The resource sensitivity of linear logic ensures that the scope of quantification is constructed and used exactly once.

# 5 Intensional Verbs

We follow Montague ((1974)) in requiring intensional verbs like *seek* to take an object of NP type. What is interesting is that this is the only step required in our setting to obtain the appropriate ambiguity predictions for intensional verbs. The *de re*/*de dicto* ambiguity of a sentence like *Bill seeks a unicorn*:

*de dicto* reading: $seek(Bill, \hat{}\lambda Q.a(x, unicorn(x), [\check{}Q](x)))$
*de re* reading: $a(x, unicorn(x), seek(Bill, \hat{}\lambda Q.[\check{}Q](x)))$

is a natural consequence, in our setting, of *seek* taking an NP-type argument.

We assign the following lexical entry to the verb *seek*:

[5] From what has been said so far, $g_\sigma$ could also be chosen to provide the scope, leading to a nonsensical result. As explained in our full analysis of quantifiers (Dalrymple, Lamping, Pereira, and Saraswat 1993), that problem is avoided by using a family of $\rightsquigarrow$ relations indexed by the type of their second argument. The relation for the meaning of the scope of quantification is the one that expects a proposition meaning, so $g_\sigma$ can not provide a scope.

(9) seek $(\uparrow \text{PRED}) = \text{`SEEK'}$
$\forall Z, Y.\ (\uparrow \text{SUBJ})_\sigma \rightsquigarrow Z$
$\otimes(\forall s, p.(\forall X.(\uparrow \text{OBJ})_\sigma \rightsquigarrow X \multimap s \rightsquigarrow p(X)) \multimap s \rightsquigarrow Y(\hat{}p))$
$\multimap \uparrow_\sigma \rightsquigarrow \mathit{seek}(Z, \hat{}Y)$

The significant fact here is that *seek* differs from an extensional verb such as *find* below (corresponding to the type $e \to e \to t$) in its specification of requirements on its object:

(10) find $(\uparrow \text{PRED}) = \text{`FIND'}$
$\forall Z, Y.(\uparrow \text{SUBJ})_\sigma \rightsquigarrow Z \otimes (\uparrow \text{OBJ})_\sigma \rightsquigarrow Y \multimap \uparrow_\sigma \rightsquigarrow \mathit{find}(Z, Y)$

Note also the use of the operators "$\hat{}$" and "$\check{}$" of IL. Computationally, this implies that our proofs have to be carried out in a logic whose terms are (typed) lambda-expressions that satisfy $\alpha-$, $\beta-$ and $\eta$-equality and also the law $\check{}(\hat{}P) = P$, for all $P$.

The lexical entry for *seek* can be paraphrased as follows:

| | |
|---|---|
| $\forall Z, Y.\ (\uparrow \text{SUBJ})_\sigma \rightsquigarrow Z \otimes$ | The verb *seek* requires a meaning $Z$ for its subject and |
| $(\forall s, p.$ $(\forall X.\ (\uparrow \text{OBJ})_\sigma \rightsquigarrow X$ $\multimap s \rightsquigarrow p(X))$ $\multimap s \rightsquigarrow Y(\hat{}p))$ | a meaning $\hat{}Y$ for its object, where $Y$ is an NP meaning applied to the meaning $p$ of an arbitrarily-chosen 'scope' $s$, (∗) |
| $\multimap \uparrow_\sigma \rightsquigarrow \mathit{seek}(Z, \hat{}Y)$ | to produce the clause meaning $\mathit{seek}(Z, \hat{}Y)$. |

Rather than looking for an entity type meaning for its object, the requirement expressed by the subformula labeled (∗) exactly describes the form of quantified NP meanings discussed in the previous section. In this case, a quantified NP in the object position is one that can accept anything that takes a meaning for $(\uparrow \text{OBJ})_\sigma$ to a meaning for any $s$, and convert that into a meaning for the $s$. In particular, the quantified NP *a unicorn* will fill the requirement, as we now demonstrate.

The f-structure for *Bill seeks a unicorn* is:

(11)
$$f\!:\begin{bmatrix} \text{PRED} & \text{`SEEK'} \\ \text{SUBJ} & g\!:\begin{bmatrix} \text{PRED} & \text{`BILL'} \end{bmatrix} \\ \text{OBJ} & h\!:\begin{bmatrix} \text{SPEC} & \text{`A'} \\ \text{PRED} & \text{`UNICORN'} \end{bmatrix} \end{bmatrix}$$

The semantic information associated with this f-structure is:

$$\begin{array}{ll}
\textbf{seeks}: & \forall Z, Y. \quad g_\sigma \rightsquigarrow Z \\
& \qquad \otimes(\forall s, p.(\forall X. h_\sigma \rightsquigarrow X \multimap s \rightsquigarrow p(X)) \multimap s \rightsquigarrow Y(\hat{}p)) \\
& \quad \multimap f_\sigma \rightsquigarrow seek(z, \hat{}Y) \\
\textbf{Bill}: & g_\sigma \rightsquigarrow Bill \\
\textbf{a-unicorn}: & \forall H, S.(\forall x. h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Sx) \multimap H \rightsquigarrow a(z, unicorn(z), Sz)
\end{array}$$

These are the premises for the deduction of the meaning of the sentence *Bill seeks a unicorn*. From the premises **Bill** and **seeks**, we can conclude by modus ponens:

$$\begin{array}{rl}
\textbf{Bill-seeks}: & \forall Y.(\forall s, p.(\forall X. h_\sigma \rightsquigarrow X \multimap s \rightsquigarrow p(X)) \multimap s \rightsquigarrow Y(\hat{}p)) \\
& \quad \multimap f_\sigma \rightsquigarrow seek(Bill, \hat{}Y)
\end{array}$$

Different derivations starting from the premises **Bill-seeks** and **a-unicorn** will yield the different readings of *Bill seeks a unicorn* that we seek.

## 5.1 De Dicto Reading

The formula **a-unicorn** is exactly what is required by the antecedent of **Bill-seeks** provided that the following substitutions are performed:

$$\begin{array}{l}
H \mapsto s \\
S \mapsto p \\
X \mapsto x \\
Y \mapsto \lambda P. a(z, unicorn(z), [\check{}P](z))
\end{array}$$

We can thus conclude the desired *de dicto* reading:

$$f_\sigma \rightsquigarrow seek(Bill, \hat{}\lambda P. a(z, unicorn(z), [\check{}P](z)))$$

To show how the premises also support a *de re* reading, we take first a short detour through a simpler example.

## 5.2 Nonquantified Objects

The meaning constructor for *seek* also allows for nonquantified objects as arguments, without needing a special type-raising rule. Consider the f-structure for the sentence *Bill seeks Al*:

(12)
$$f\!:\!\begin{bmatrix} \textsc{pred} & \text{`SEEK'} \\ \textsc{subj} & g\!:\!\begin{bmatrix} \textsc{pred} & \text{`BILL'} \end{bmatrix} \\ \textsc{obj} & h\!:\!\begin{bmatrix} \textsc{pred} & \text{`AL'} \end{bmatrix} \end{bmatrix}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{h_\sigma \rightsquigarrow Al \vdash h_\sigma \rightsquigarrow Al \qquad s \rightsquigarrow P(Al) \vdash s \rightsquigarrow P(Al)}{h_\sigma \rightsquigarrow Al, h_\sigma \rightsquigarrow Al \multimap s \rightsquigarrow P(Al) \vdash s \rightsquigarrow P(Al)}}{h_\sigma \rightsquigarrow Al, (\forall x.h_\sigma \rightsquigarrow x \multimap s \rightsquigarrow P(x)) \vdash s \rightsquigarrow P(Al)}}{h_\sigma \rightsquigarrow Al \vdash (\forall x.h_\sigma \rightsquigarrow x \multimap s \rightsquigarrow P(x)) \multimap s \rightsquigarrow P(Al)}}{h_\sigma \rightsquigarrow Al \vdash \forall P.(\forall x.h_\sigma \rightsquigarrow x \multimap s \rightsquigarrow P(x)) \multimap s \rightsquigarrow P(Al)}$$

Figure 2: Proof that **Al** can function as a quantifier

The lexical entry for $Al$ is analogous to the one for *Bill*. We begin with the premises **Bill-seeks** and **Al**:

**Bill-seeks**: $\forall Y.(\forall s, p.(\forall X.h_\sigma \rightsquigarrow X \multimap s \rightsquigarrow p(X)) \multimap s \rightsquigarrow Y(\hat{}p)) \multimap f_\sigma \rightsquigarrow seek(Bill, \hat{}Y)$

**Al**: $h_\sigma \rightsquigarrow Al$

For the derivation to proceed, **Al** must supply the NP meaning constructor that **Bill-seeks** requires. This is possible because **Al** can map a proof $\Pi$ of the meaning for $s$ from the meaning for $h$ into a meaning for $s$, simply by supplying $h_\sigma \rightsquigarrow Al$ to $\Pi$. Formally, from **Al** we can prove (Figure 2):

(13) $\forall P.(\forall x.h_\sigma \rightsquigarrow x \multimap s \rightsquigarrow P(x)) \multimap s \rightsquigarrow P(Al)$

This corresponds to the Montagovian type-raising of a proper name meaning to an NP meaning, and also to the undirected Lambek calculus derivation of the sequent $e \Rightarrow (e \rightarrow t) \rightarrow t$.

Formula (13) with the substitutions

$$P \mapsto p, Y \mapsto \lambda P.[\check{}P](Al)$$

can then be used to satisfy the antecedent of **Bill-seeks** to yield the desired result:

$$f_\sigma \rightsquigarrow seek(Bill, \hat{}\lambda P.[\check{}P](Al))$$

It is worth contrasting the foregoing derivation with treatments of the same issue in the lambda calculus. The function $\lambda x.\lambda P.Px$ raises a term like $Al$ to the quantified NP form $\lambda P.P(Al)$, so it is easy to modify $Al$ to make it suitable for **seek**. But the conversion must be explicitly applied somewhere, either in a meaning postulate or in an alternate definition for *seek*, in order to carry out the derivation. This is because a lambda calculus function must specify exactly what is to be done with its arguments and how they will interact. It must presume some functional form of its arguments in order to achieve its own function. Thus, it is impossible to write a function that is indifferent with respect to whether its argument is $Al$ or $\lambda P.P(Al)$.

In the deductive framework, on the other hand, the exact way in which different propositions can interact is not fixed, although it is constrained by

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{I\rightsquigarrow Z \vdash I\rightsquigarrow Z \qquad S\rightsquigarrow P(Z) \vdash S\rightsquigarrow P(Z)}{I\rightsquigarrow Z, I\rightsquigarrow Z \multimap S\rightsquigarrow P(Z) \vdash S\rightsquigarrow P(Z)}}{I\rightsquigarrow Z, (\forall x.\ I\rightsquigarrow x \multimap S\rightsquigarrow P(x)) \vdash S\rightsquigarrow P(Z)}}{I\rightsquigarrow Z \vdash (\forall x.\ I\rightsquigarrow x \multimap S\rightsquigarrow P(x)) \multimap S\rightsquigarrow P(Z)}}{I\rightsquigarrow Z \vdash \forall S, P.\ (\forall x.\ I\rightsquigarrow x \multimap S\rightsquigarrow P(x)) \multimap S\rightsquigarrow P(Z)}}{\vdash I\rightsquigarrow Z \multimap \forall S, P.\ (\forall x.\ I\rightsquigarrow x \multimap S\rightsquigarrow P(x)) \multimap S\rightsquigarrow P(Z)}}{\vdash \forall I, Z.\ I\rightsquigarrow Z \multimap \forall S, P.\ (\forall x.\ I\rightsquigarrow x \multimap S\rightsquigarrow P(x)) \multimap S\rightsquigarrow P(Z)}$$

Figure 3: General Type-Raising Theorem

their (logical and quantificational) propositional structure. Thus $h_\sigma \rightsquigarrow Al$ can function as any logical consequence of itself, in particular as:

$$\forall S, P.\ (\forall x.\ h_\sigma \rightsquigarrow x \multimap S\rightsquigarrow P(x)) \multimap S\rightsquigarrow P(Al)$$

This flexibility, which is also found in syntactic-semantic analyses based on the Lambek calculus and its variants ((Moortgat , 1988; Moortgat , 1992b; van Benthem , 1991)), seems to align well with some of the type flexibility in natural language.

## 5.3 Type Raising and Quantifying In

The derivation in Figure 2 can be generalized as shown in Figure 3 to produce the general type-raising theorem:

(14) $\forall I, Z.\ I\rightsquigarrow Z \multimap (\forall S, P.\ (\forall x. I\rightsquigarrow x \multimap S\rightsquigarrow P(x)) \multimap S\rightsquigarrow P(Z))$

This theorem can be used to raise meanings of $e$ type to $(e \to t) \to t$ type, or, dually, to quantify into verb argument positions. For example, with the variable instantiations

$$\begin{aligned} I &\mapsto h_\sigma \\ X &\mapsto x \\ P &\mapsto p \\ S &\mapsto s \\ Y &\mapsto \lambda R.[\check{}R](Z) \end{aligned}$$

we can use transitivity of implication to combine (14) with **Bill-seeks** to derive:

$$\textbf{Bill-seeks}': \forall Z.\ h_\sigma \rightsquigarrow Z \multimap f_\sigma \rightsquigarrow \mathit{seek}(\mathit{Bill}, \hat{}\lambda R.[\check{}R](Z))$$

This formula can then be combined with arguments of type $e$ to produce a meaning for $f_\sigma$. For instance, it will take the non-type-raised $h_\sigma \rightsquigarrow Al$ to yield the same result

$$f_\sigma \rightsquigarrow \mathit{seek}(\mathit{Bill}, \hat{}\lambda R.[\check{}R](Al))$$

as the combination of **Bill-seeks** with the type-raised version of **Al**. In fact, **Bill-seeks′** corresponds to type $e \to t$, and can thus be used as the scope of a quantifier, which would then quantify into the intensional direct object argument of *seek*. As we will presently see, that is exactly what is needed to derive *de re* readings.

## 5.4 De Re Reading

We have just seen how theorem (14) provides a general mechanism for quantifying into intensional argument positions. In particular, it allowed the derivation of **Bill-seeks′** from **Bill-seeks**. Now, given the premises

**Bill-seeks′**: $\forall Z.\ h_\sigma \rightsquigarrow Z \multimap f_\sigma \rightsquigarrow seek(Bill, \hat{}\lambda R.[\check{}R](Z))$
**a-unicorn**: $\forall H, S.\ (\forall x.\ h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Sx) \multimap H \rightsquigarrow a(z, unicorn(z), Sz)$

and the variable substitutions

$$\begin{aligned} Z &\mapsto x \\ H &\mapsto f_\sigma \\ S &\mapsto \lambda z.seek(Bill, \hat{}\lambda R.[\check{}R](z)) \end{aligned}$$

we can apply modus ponens to derive the *de re* reading of *Bill seeks a unicorn*:

$$f_\sigma \rightsquigarrow a(z, unicorn(z), seek(Bill, \hat{}\lambda R.[\check{}R](z)))$$

## 5.5 A Comparison with Categorial Approaches

The analysis presented here has strong similarities to analyses of the same phenomena discussed by Morrill (1993) and Carpenter (1993). Following Moortgat (1992a), they add to an appropriate version of the Lambek calculus ((Lambek , 1958)) the *scope* connective $\Uparrow$, subject to the following proof rules:

$$\frac{\Gamma, v : A, \Gamma' \Rightarrow u : B \qquad \Delta, t(\lambda v.u) : B, \Delta' \Rightarrow C}{\Delta, \Gamma, t : A \Uparrow B, \Gamma', \Delta' \Rightarrow C} \quad [\text{QL}]$$

$$\frac{\Gamma \Rightarrow u : A}{\Gamma \Rightarrow \lambda v.v(u) : A \Uparrow B} \quad [\text{QR}]$$

In terms of the scope connective, a quantified noun phrase is given the category N $\Uparrow$ S, which semantically corresponds to the type $(e \to t) \to t$ and agrees with the propositional structure of our linear formulas for quantified noun phrases, for instance (8). A phrase of category $N \Uparrow S$ is an infix functor that binds a variable of type $e$, the type of individual noun phrases N, within a scope of type $t$, the type of sentences S. An intensional verb like 'seek' has, then, category

$(\mathrm{N} \setminus (\mathrm{S})/(\mathrm{N} \Uparrow \mathrm{S})$, with corresponding type $(((e \to t) \to t) \to e \to t)$. [6] Thus the intensional verb will take as direct object a quantified noun phrase, as required.

A problem arises, however, with sentences such as

(15) Bill seeks a conversation with every unicorn.

This sentence has five possible interpretations:

(16) a. $\mathit{seek}(\mathit{Bill}, \hat{}\lambda P.\mathit{every}(u, \mathit{unicorn}(u), a(z, \mathit{conv\text{-}with}(z, u), [\check{}P](z))))$

b. $\mathit{seek}(\mathit{Bill}, \hat{}\lambda P.a(z, \mathit{every}(u, \mathit{unicorn}(u), \mathit{conv\text{-}with}(z, u)), [\check{}P](z)))$

c. $\mathit{every}(u, \mathit{unicorn}(u), \mathit{seek}(\mathit{Bill}, \hat{}\lambda P.a(z, \mathit{conv\text{-}with}(z, u), [\check{}P](z))))$

d. $\mathit{every}(u, \mathit{unicorn}(u), a(z, \mathit{conv\text{-}with}(z, u), \mathit{seek}(\mathit{Bill}, \hat{}\lambda P.[\check{}P](z))))$

e. $a(z, \mathit{every}(u, \mathit{unicorn}(u), \mathit{conv\text{-}with}(z, u)), \mathit{seek}(\mathit{Bill}, \hat{}\lambda P.[\check{}P](z)))$

Both our approach and the categorial analysis using the scope connective have no problem in deriving interpretations (16b), (16c), (16d) and (16e). In those cases, the scope of 'every unicorn' is interpreted as an appropriate term of type $e \to t$. However, the situation is different for interpretation (16a), in which both the conversations and the unicorn are *de dicto*, but the conversations sought may be different for different unicorns sought. As we will show below, this interpretation can be easily derived within our framework. However, a similar derivation does not appear possible in terms of the categorial scoping connective.

The difficulty for the categorial account is that the category $\mathrm{N} \Uparrow \mathrm{S}$ represents a phrase that plays the role of a category N phrase where it appears, but takes an S (dependent on the N) as its scope. In the derivation of (16a), however, the scope of 'every unicorn' is 'a conversation with', which is not of category S. Semantically, 'a conversation with' is represented by:

(17) $\lambda P.\lambda u.a(z, \mathit{conv\text{-}with}(z, u), [\check{}P](z)) : (e \to t) \to (e \to t)$

The *undirected* Lambek calculus ((van Benthem , 1991)) allows us to compose (17) with the interpretation of 'every unicorn':

(18) $\lambda Q.\mathit{every}(u, \mathit{unicorn}(u), Q(u)) : (e \to t) \to t$

to yield:

(19) $\lambda P.\mathit{every}(u, \mathit{unicorn}(u), a(z, \mathit{conv\text{-}with}(z, u), [\check{}P](z))) : (e \to t) \to t$

As we will see below, our linear logic formulation also allows that derivation step.

In contrast, as Moortgat (1992a) points out, the categorial rule [QR] is not powerful enough to raise $N \Uparrow S$ to take as scope any functor whose result is a S. In particular, the sequent

[6] These category and type assignments are an oversimplification since intensional verbs like 'seek' require a direct object of type $s \to ((e \to t) \to t)$, but for the present discussion the simpler category and type are sufficient. Morrill (1993) provides a full treatment.

(20) $\mathrm{N} \Uparrow \mathrm{S} \Rightarrow \mathrm{N} \Uparrow (\mathrm{N} \Uparrow \mathrm{S})$

is not derivable, whereas the corresponding "semantic" sequent (up to permutation)

(21) $q : (e \to t) \to t \Rightarrow$
$\lambda R.\lambda P.q(\lambda x.R(P)(x)) : ((e \to t) \to (e \to t)) \to ((e \to t) \to t)$

is derivable in the undirected Lambek calculus. Sequent (21) will in particular raise (18) to a function that, applied to (17), produces (19), as required.

Furthermore, the solution proposed by Morrill (1993) to make the scope calculus complete is to restrict the intended interpretation of $\Uparrow$ so that (20) is not valid. Thus, *contra* Carpenter (1993), Morrill's logically more satisfying account of $\Uparrow$ is not a step towards making reading (16a) available.

We now give the derivation of the interpretation (16a) in our framework. The f-structure for (15) is:

(22)
$$f:\begin{bmatrix} \textsc{pred} & \text{'SEEK'} \\ \textsc{subj} & g:\begin{bmatrix} \textsc{pred} & \text{'Bill'} \end{bmatrix} \\ \textsc{obj} & h:\begin{bmatrix} \textsc{spec} & \text{'A'} \\ \textsc{pred} & \text{'CONVERSATION'} \\ \textsc{obl}_{\textsc{with}} & i:\begin{bmatrix} \textsc{spec} & \text{'EVERY'} \\ \textsc{pred} & \text{'UNICORN'} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The two formulas **Bill-seeks** and **every-unicorn** can be derived as described before:

**Bill-seeks**: $\forall Y.(\forall s, p.(\forall X.h_\sigma \rightsquigarrow X \multimap s \rightsquigarrow pX) \multimap s \rightsquigarrow Y(\hat{}p)) \multimap f_\sigma \rightsquigarrow \mathit{seek}(\mathit{Bill}, \hat{}Y)$

**every-unicorn**: $\forall G, S.(\forall x.i_\sigma \rightsquigarrow x \multimap G \rightsquigarrow Sx) \multimap G \rightsquigarrow \mathit{every}(u, \mathit{unicorn}(u), Su)$

As explained in more detail in Dalrymple, Lamping, Pereira, and Saraswat (1993), the remaining lexical premises for (22) are:

**a**: $\forall H, R, T.((\forall x.(h_\sigma \textsc{var}) \rightsquigarrow x \multimap (h_\sigma \textsc{restr}) \rightsquigarrow Rx) \otimes (\forall x.h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Tx)) \multimap H \rightsquigarrow a(z, Rz, Tz)$

**conv-with**: $\forall Z, X.\ (h_\sigma \textsc{var}) \rightsquigarrow Z \otimes i \rightsquigarrow X \multimap (h_\sigma \textsc{restr}) \rightsquigarrow \mathit{conv\text{-}with}(Z, X)$

From these premises we immediately derive

$$\forall X, H, T.i_\sigma \rightsquigarrow X \otimes (\forall x.h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Tx)) \multimap H \rightsquigarrow a(z, \mathit{conv\text{-}with}(z, X), Tz)$$

which can be rewritten as:

$$(23)\quad \forall H,T.(\forall x.h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Tx) \multimap \\ \forall X.\ (i_\sigma \rightsquigarrow X \multimap H \rightsquigarrow a(z, \textit{conv-with}(z,X), Tz))$$

With the substitutions

$$X \mapsto x, G \mapsto H, S \mapsto \lambda v.a(z, \textit{conv-with}(z,v), Tz))$$

formula (23) can be combined with **every-unicorn** to yield the required quantifier-type formula:

$$(24)\quad \forall H,T.(\forall x.h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Tx) \multimap \\ H \rightsquigarrow \textit{every}(u, \textit{unicorn}(u), a(z, \textit{conv-with}(z,u), Tz))$$

Using substitutions

$$\begin{array}{lcl} H & \mapsto & s \\ T & \mapsto & p \\ Y & \mapsto & \lambda R.\textit{every}(u, \textit{unicorn}(u), a(z, \textit{conv-with}(z,u), [\check{}R](z)))) \end{array}$$

and modus ponens, we then combine (24) with **Bill-seeks** to obtain the desired final result:

$$f_\sigma \rightsquigarrow \textit{seek}(\textit{Bill}, \hat{}\lambda R.\textit{every}(u, \textit{unicorn}(u), a(z, \textit{conv-with}(z,u), [\check{}R](z))))$$

We see thus that our more flexible connection between syntax and semantics permits the full range of type flexibility provided categorial *semantics* without losing the rigorous connection to syntax. In contrast, current categorial accounts of the syntax-semantics interface do not appear to offer the needed flexibility when syntactic and semantic composition are more indirectly connected, as in the present case.

# 6 Conclusion

We have shown that our deductive framework allows us to predict the correct set of readings for intensional verbs with quantified and nonquantified direct objects if we make a single assumption: that intensional verbs require a quantified direct object. This assumption is, of course, the starting point of the standard Montagovian treatment of intensional verbs. But that treatment depends on the additional machinery of quantifying in to generate *de re* readings of quantified direct objects, and that of explicit type raising to accommodate unquantified direct objects. In our approach those problems are handled directly by the deductive apparatus without further stipulation.

These results, as well as our previous work on quantifier scope, suggest the advantages of a generalized form of compositionality in which the semantic contributions of phrases are represented by logical formulas rather than by

functional abstractions as in traditional compositionality. The fixed application order and fixed type requirements of lambda terms are just too restrictive when it comes to encoding the freer order of information presentation in natural language. In this observation, our treatment is closely related to systems of syntactic and semantic type assignment based on the Lambek calculus and its variants. However, we differ from those categorial approaches in providing an explicit link between functional structures and semantic derivations that does not depend on linear order and constituency in syntax to keep track of predicate-argument relations. Thus we avoid the need to force both syntax and semantics into an uncomfortably tight categorial embrace.

## Acknowledgments

We thank David Israel, Michael Moortgat and Stanley Peters for discussions on the subject of this paper.

## References

[Barwise and Cooper 1981] Barwise, Jon, and Robin Cooper. 1981. Generalized Quantifiers and Natural Language. *Linguistics and Philosophy* 4:159–219.

[Carpenter 1993] Carpenter, Bob. 1993. Quantification and Scoping: a Deductive Account. Submitted for publication.

[Dalrymple et al. 1993] Dalrymple, Mary, Angie Hinrichs, John Lamping, and Vijay Saraswat. 1993. The Resource Logic of Complex Predicate Interpretation. In *Proceedings of the 1993 Republic of China Computational Linguistics Conference (ROCLING)*. Hsitou National Park, Taiwan, September. Computational Linguistics Society of R.O.C.

[Dalrymple et al. 1994] Dalrymple, Mary, John Lamping, Fernando C. N. Pereira, and Vijay Saraswat. 1994. A Deductive Account of Quantification in LFG. In *Quantifiers, Deduction, and Context*, ed. Makoto Kanazawa, Christopher J. Piñón, and Henriette de Swart. Stanford, California: Center for the Study of Language and Information.

[Dalrymple et al. 1993] Dalrymple, Mary, John Lamping, and Vijay Saraswat. 1993. LFG Semantics Via Constraints. In *Proceedings of the Sixth Meeting of the European ACL*. University of Utrecht, April. European Chapter of the Association for Computational Linguistics.

[Girard 1987] Girard, J.-Y. 1987. Linear Logic. *Theoretical Computer Science* 45:1–102.

[Halvorsen 1983] Halvorsen, Per-Kristian. 1983. Semantics for Lexical-Functional Grammar. *Linguistic Inquiry* 14(4):567–615.

[Halvorsen and Kaplan 1988] Halvorsen, Per-Kristian, and Ronald M. Kaplan. 1988. Projections and Semantic Description in Lexical-Functional Grammar. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, 1116–1122. Tokyo, Japan. Institute for New Generation Systems.

[Kaplan 1987] Kaplan, Ronald M. 1987. Three Seductions of Computational Psycholinguistics. In *Linguistic Theory and Computer Applications*, ed. Peter Whitelock, Harold Somers, Paul Bennett, Rod Johnson, and Mary McGee Wood. 149–188. London: Academic Press.

[Kaplan and Bresnan 1982] Kaplan, Ronald M., and Joan Bresnan. 1982. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In *The Mental Representation of Grammatical Relations*, ed. Joan Bresnan. 173–281. Cambridge, MA: The MIT Press.

[Lambek 1958] Lambek, Joachim. 1958. The Mathematics of Sentence Structure. *American Mathematical Monthly* 65:154–170.

[Montague 1974] Montague, Richard. 1974. The Proper Treatment of Quantification in Ordinary English. In *Formal Philosophy*, ed. Richmond Thomason. New Haven: Yale University Press.

[Moortgat 1988] Moortgat, Michael. 1988. *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus*. Doctoral dissertation, University of Amsterdam, Amsterdam, The Netherlands.

[Moortgat 1992a] Moortgat, Michael. 1992a. Generalized Quantifiers and Discontinuous Type Constructors. In *Discontinuous Constituency*, ed. W. Sijtsma and H. van Horck. Berlin, Germany: Mouton de Gruyter. To appear.

[Moortgat 1992b] Moortgat, Michael. 1992b. Labelled Deductive Systems for Categorial Theorem Proving. In *Proceedings of the Eighth Amsterdam Colloquium*, ed. P. Dekker and M. Stokhof, 403–423. Amsterdam. Institute for Logic, Language and Computation.

[Morrill 1993] Morrill, Glyn V. 1993. *Type Logical Grammar: Categorial Logic of Signs*. Studies in Linguistics and Philosophy. Dordrecht, Holland: Kluwer Academic Publishers. To appear.

[Pereira 1990] Pereira, Fernando C. N. 1990. Categorial Semantics and Scoping. *Computational Linguistics* 16(1):1–10.

[Pereira 1991] Pereira, Fernando C. N. 1991. Semantic Interpretation as Higher-Order Deduction. In *Logics in AI: European Workshop JELIA'90*, ed. Jan van Eijck, 78–96. Amsterdam, Holland. Springer-Verlag.

[Saraswat 1989] Saraswat, Vijay A. 1989. *Concurrent Constraint Programming Languages*. Doctoral dissertation, Carnegie-Mellon University. Reprinted by MIT Press, Doctoral Dissertation Award and Logic Programming Series, 1993.

[Scedrov 1993] Scedrov, Andre. 1993. A Brief Guide to Linear Logic. In *Current Trends in Theoretical Computer Science*, ed. G. Rozenberg and A. Salomaa. World Scientific Publishing Co. Revised and expanded version of the article originally appearing in *Bulletin of the European Assoc. for Theoretical Computer Science* 41, 1990.

[van Benthem 1991] van Benthem, Johan. 1991. *Language in Action: Categories, Lambdas and Dynamic Logic*. Amsterdam: North-Holland.