Linear Logic for Meaning Assembly*

Mary Dalrymple
John Lamping
Fernando Pereira
Vijay Saraswat

March 20, 1995

1 Introduction

Semantic theories of natural language associate meanings with utterances by providing meanings for lexical items and rules for determining the meaning of larger units given the meanings of their parts. Meanings are often assumed to combine via function application, which works well when constituent structure trees are used to guide semantic composition. However, we believe that the *functional structure* of Lexical-Functional Grammar is best used to provide the syntactic information necessary for constraining derivations of meaning in a cross-linguistically uniform format. It has been difficult, however, to reconcile this approach with the combination of meanings by function application.

In contrast to compositional approaches, we present a deductive approach to assembling meanings, based on reasoning with constraints, which meshes well with the unordered nature of information in the functional structure. Our use of *linear logic* as a 'glue' for assembling meanings allows for a coherent treatment of the LFG requirements of completeness and coherence as well as of modification and quantification.

2 The Framework

This paper provides a brief overview of our ongoing investigation in the use of formal deduction to explicate the relationship between syntactic analyses in Lexical-Functional Grammar (LFG) and semantic interpretations (Dalrymple, Lamping, and Saraswat, 1993; Dalrymple et al., 1993; Dalrymple et al., 1994a; Dalrymple et al., 1994b; Dalrymple et al., 1995; Dalrymple et

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

al., 1995). We use *linear logic* (Girard, 1987) to represent the connection between two dissimilar linguistic levels: LFG f-structures and their semantic interpretations.

F-structures provide a uniform representation of syntactic information relevant to semantic interpretation that abstracts away from the varying details of phrase structure and linear order in particular languages. But as noted by Halvorsen (1983) and Reyle (1988), the flatter, unordered functional structure of LFG does not fit well with traditional semantic compositionality, based on functional abstraction and application, which mandates a rigid order of semantic composition. We are thus led to a more relaxed form of compositionality, in which, as in more traditional ones, the semantics of each lexical entry in a sentence is used exactly once in interpretation, but without imposing a rigid order of composition. Approaches to semantic interpretation that encode semantic representations in attribute-value structures (Pollard and Sag, 1987; Fenstad et al., 1987; Pollard and Sag, 1994) offer such a relaxation of compositionality, but are unable to properly represent constraints on variable binding and scope (Pereira, 1990).

The present approach, in which linear logic is used to specify the relation between f-structures and their meanings, provides exactly what is required for a calculus of semantic composition for LFG. It can directly represent the constraints on the creation and use of semantic units in sentence interpretation, including those pertaining to variable binding and scope, without forcing a particular hierarchical order of composition, except as required by the properties of particular lexical entries.

2.1 Syntax

In languages like English, the substantial scaffolding provided by surface constituent structure trees is often a useful guide for semantic composition, and the λ-calculus is a convenient formalism for assembling the semantics along that scaffolding (Montague, 1974). This is because the derivation of the meaning of a phrase can often be viewed as mirroring the surface constituent structure of the English phrase. The sentence 'Bill appointed Hillary' has the surface constituent structure indicated by the bracketing in 1:

(1) $[_{S}\ [_{NP}$ Bill$]\ [_{VP}$ appointed $[_{NP}$ Hillary$]]]$

The verb is viewed as bearing a close syntactic relation to the object and forming a constituent with it; this constituent then combines with the subject of the sentence. Similarly, the meaning of the verb can be viewed as a two-place function which is applied first to the object, then to the subject, producing the meaning of the sentence.

However, this approach is not as natural for languages whose surface structure does not resemble English. For instance, a problem is presented

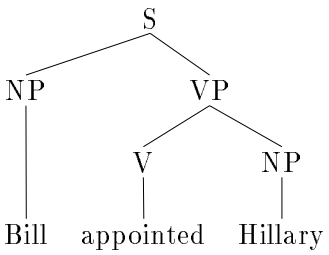
by VSO languages such as Irish (McCloskey, 1979). To preserve the hypothesis that surface constituent structure provides the proper scaffolding for semantic interpretation in VSO languages, one of two assumptions must be made. One must assume either that semantic composition is nonuniform across languages (leading to loss of explanatory power), or that semantic composition proceeds not with reference to surface syntactic structure, but instead with reference to a more abstract (English-like) constituent structure representation. This second hypothesis seems to us to render vacuous the claim that surface constituent structure is useful in semantic composition.

Further problems are encountered in the semantic analysis of a free word order language such as Warlpiri (Simpson, 1991), where surface constituent structure does not always give rise to units that are semantically coherent or useful. Here, an argument of a verb may not even appear as a single unit at surface constituent structure; further, arguments of a verb may appear in various different places in the string. In such cases, the appeal to an order of composition different from that of English is particularly unattractive, since different orders of composition would be needed for each possible word order sequence.

The observation that surface constituent structure does not always provide the optimal set of constituents or hierarchical structure to guide semantic interpretation has led to efforts to use a more abstract structure to guide semantic composition. As originally proposed by Kaplan and Bresnan (1982) and Halvorsen (1983), the *functional structure* or *f-structure* of LFG is a representation of such a structure. The c-structure and f-structure for sentence (2) are given in (3):

(2) Bill appointed Hillary.

(3) C-structure: F-structure:



$$\begin{bmatrix} \textsc{pred} & \text{`APPOINT'} \\ \textsc{subj} & [\,\textsc{pred} \quad \text{`BILL'}\,] \\ \textsc{obj} & [\,\textsc{pred} \quad \text{`HILLARY'}\,] \end{bmatrix}$$

As illustrated, an f-structure represents the more abstract syntactic function-argument structure of the sentence, encoding relations such as SUBJ and OBJ. Those relations are realized in different c-structure forms in different languages, but are represented directly and uniformly in the f-structure. Formally, an f-structure consists of a collection of attributes, such as PRED, SUBJ, and OBJ, whose values can, in turn, be other f-structures.

We will not provide a detailed account of the relation between c-structure and f-structure; for such an account, see Bresnan (1982), Levin, Rappaport, and Zaenen (1983), and the references cited there. Here, we will begin with the f-structures for the examples we discuss, and concentrate on an exposition of how the f-structure provides information necessary to carry out a semantic deduction.

2.2 Semantic Projections

Following work by Kaplan (1987) and Halvorsen and Kaplan (1988), we make use of a *semantic* or *σ-projection* function σ to map f-structures to *semantic* or *σ-structures* encoding information about f-structure meaning. For a particular use of 'Bill', the resulting configuration is:

(4) $h:\left[\textsc{pred} \quad \text{'}\textsc{Bill}\text{'}\right] \qquad h_\sigma:[\,] \rightsquigarrow \mathit{Bill}$

The semantic projection function σ is represented by an arrow. We use labels such as h to refer to particular f-structures. The association between the semantic structure h_σ and a meaning P can be represented by the atomic formula $h_\sigma \rightsquigarrow P$, which we will refer to as a *meaning constructor*, where $\rightsquigarrow$ is an otherwise uninterpreted binary predicate symbol. (In fact, we use not one but a family of relations $\rightsquigarrow_\tau$ indexed by the semantic type of the intended second argument, although for simplicity we will omit the type subscript whenever it is determinable from context.) For the case at hand, if a particular occurrence of 'Bill' in a sentence is associated with f-structure h, the semantic constraint will be instantiated as:

$$h_\sigma \rightsquigarrow \mathit{Bill}$$

representing the association between h_σ and the constant *Bill* representing its meaning.

We will often informally say that P is h's meaning without referring to the role of the semantic structure h_σ in $h_\sigma \rightsquigarrow P$. Actually, however, f-structures and their semantic projections must be distinguished, because semantic projections can carry more information than just the association to the meaning for the corresponding f-structure; see Dalrymple et al. (1994b) for more discussion.

As noted above, the λ-calculus is not a very natural tool for combining meanings of f-structure constituents. The problem is that the subconstituents of an f-structure are unordered, and so the fixed order of combination of a functor with its arguments imposed by the λ-calculus is no longer an advantage; in fact, it becomes a disadvantage, since an artificial ordering must be imposed on the composition of meanings. Furthermore, the components of the f-structure may be not only complements but also modifiers, which contribute to the final semantics in a very different way.

Instead, we assume that lexical entries contribute premises – meaning constructors – to a logical deduction. The meaning constructor is a linear logic formula that can be understood as 'instructions' for combining the meanings of the lexical entry's syntactic arguments to obtain the meaning of the f-structure headed by the entry. In effect, then, our approach uses linear logic as the 'glue' with which semantic representations are assembled. Once all the constraints are assembled, deduction in the logic is used to infer the meaning of the entire structure. Throughout this process we maintain a sharp distinction between assertions about the meaning (the glue) and the meaning itself.

In the case of the verb 'appointed', the meaning constructor is a formula consisting of instructions on how to assemble the meaning of a sentence with main verb 'appointed', given the meanings of its subject and object. The verb 'appointed' contributes the following f-structure and meaning constructor:

(5)
$$f:\begin{bmatrix}\textsc{pred} & \text{`APPOINT'}\\ \textsc{subj} & [\,]\\ \textsc{obj} & [\,]\end{bmatrix} \qquad f_\sigma : [\,]$$

$$\forall X, Y.\ (f\ \textsc{subj})_\sigma \rightsquigarrow X \otimes (f\ \textsc{obj})_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow \mathit{appoint}(X, Y)$$

The meaning constructor, given in the last line of (5), asserts that:

- if f's subject (f SUBJ) has meaning X
- and ($\otimes$) f's object (f OBJ) has meaning Y
- then ($\multimap$) f has meaning $\mathit{appoint}(X, Y)$.

The linear-logic connectives of multiplicative conjunction $\otimes$ and linear implication $\multimap$ are used to specify how the meaning of a clause headed by the verb is composed from the meanings of the arguments of the verb. For the moment, we can think of the linear connectives as playing the same role as the analogous classical connectives conjunction $\wedge$ and implication $\rightarrow$, but we will see that the specific properties of the linear connectives are essential to guarantee that lexical entries bring into the interpretation process all and only the information provided by the corresponding words.

We will now show how meanings are assembled by linear-logic deduction. For readability, we will present derivations informally. As a first example, consider the following f-structures:

(6)
$$f:\begin{bmatrix}\textsc{pred} & \text{`APPOINT'}\\ \textsc{subj} & g:[\,\textsc{pred} \quad \text{`BILL'}\,]\\ \textsc{obj} & h:[\,\textsc{pred} \quad \text{`HILLARY'}\,]\end{bmatrix}$$

From the lexical entries for 'Bill', 'Hillary', and 'appointed', we obtain the following meaning constructors, abbreviated as **bill**, **hillary**, and **appointed**:[1]

bill:	$g_\sigma \rightsquigarrow \mathit{Bill}$
hillary:	$h_\sigma \rightsquigarrow \mathit{Hillary}$
appointed:	$\forall X, Y.\ g_\sigma \rightsquigarrow X \otimes h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow \mathit{appoint}(X, Y)$

In the following, assume that the formula **bill-appointed** is defined thus:

$$\textbf{bill-appointed}: \quad \forall Y.\ h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow \mathit{appoint}(\mathit{Bill}, Y)$$

Then the following derivation is possible in linear logic ($\vdash$ stands for the linear-logic entailment relation):

(7)		**bill** $\otimes$ **hillary** $\otimes$ **appointed**	$(\mathit{Premises.})$
	$\vdash$	**bill-appointed** $\otimes$ **hillary**	$X \mapsto \mathit{Bill}$
	$\vdash$	$f_\sigma \rightsquigarrow \mathit{appoint}(\mathit{Bill}, \mathit{Hillary})$	$Y \mapsto \mathit{Hillary}$

Of course, another derivation is also possible. Assume that the formula **appointed-hillary** is defined as:

$$\textbf{appointed-hillary}: \quad \forall X.\ g_\sigma \rightsquigarrow X \multimap f_\sigma \rightsquigarrow \mathit{appoint}(X, \mathit{Hillary})$$

Then we have the following derivation:

(8)		**bill** $\otimes$ **hillary** $\otimes$ **appointed**	$(\mathit{Premises.})$
	$\vdash$	**bill** $\otimes$ **appointed-hillary**	$Y \mapsto \mathit{Hillary}$
	$\vdash$	$f_\sigma \rightsquigarrow \mathit{appoint}(\mathit{Bill}, \mathit{Hillary})$	$X \mapsto \mathit{Bill}$

yielding the same conclusion.

In summary, each word in a sentence contributes a linear-logic formula, its meaning constructor, relating the semantic projections of specific f-structures in the LFG analysis to representations of their meanings. From these formulas, the interpretation process attempts to deduce an atomic formula relating the semantic projection of the whole sentence to a representation of the sentence's meaning. Alternative derivations can sometimes yield different such conclusions, corresponding to ambiguities of semantic interpretation.

In LFG, syntactic predicate-argument structure is projected from lexical entries. Therefore, its effect on semantic composition will for the most part – in fact, in all the cases considered in this paper – be determined by lexical entries, not by phrase-structure rules. In particular, the phrase-structure rules for S and VP in the examples discussed above need not encode semantic information, but only specify how grammatical functions such as SUBJ are

[1] For the sake of illustration, we will provide only the simplest semantics for the lexical entries we discuss, ignoring (among other issues) the representation of tense and aspect.

expressed in English. In some cases, the constituent structure of a syntactic construction may make a direct semantic contribution, as when properties of the construction as a whole and not just of its lexical elements are responsible for the interpretation of the construction. Such cases include, for instance, relative clauses with no complementizer ('the man Bill met'). We will not discuss construction-specific interpretation rules in this paper.

3 Linear logic

An important motivation for using linear logic is that it allows us to directly capture the intuition that lexical items and phrases each contribute exactly once to the meaning of a sentence. As noted by Klein and Sag (1985, page 172):

> Translation rules in Montague semantics have the property that the translation of each component of a complex expression occurs exactly once in the translation of the whole. ...That is to say, we do not want the set S [of semantic representations of a phrase] to contain *all* meaningful expressions of IL which can be built up from the elements of S, but only those which use each element exactly once.

In our terms, the semantic contributions of the constituents of a sentence are not context-independent assertions that may be used or not in the derivation of the meaning of the sentence depending on the course of the derivation. Instead, the semantic contributions are *occurrences* of information which are generated and used exactly once. For example, the formula $g_\sigma \rightsquigarrow \mathit{Bill}$ can be thought of as providing one occurrence of the meaning *Bill* associated to the semantic projection g_σ. That meaning must be consumed exactly once (for example, by **appointed** in (7)) in the derivation of a meaning of the entire utterance.

It is this 'resource-sensitivity' of natural language semantics—an expression is used exactly once in a semantic derivation—that linear logic can model. The basic insight underlying linear logic is that logical formulas are *resources* that are produced and consumed in the deduction process. This gives rise to a resource-sensitive notion of implication, the *linear implication* $\multimap$: the formula $A \multimap B$ can be thought of as an action that can *consume* (one copy of) A to produce (one copy of) B. Thus, the formula $A \otimes (A \multimap B)$ linearly entails B. It does not entail $A \otimes B$ (because the deduction consumes A), and it does not entail $(A \multimap B) \otimes B$ (because the linear implication is also consumed in doing the deduction). This resource-sensitivity not only disallows arbitrary duplication of formulas, but also disallows arbitrary deletion of formulas. Thus the linear multiplicative conjunction $\otimes$ is sensitive to the multiplicity of formulas: $A \otimes A$ is not equivalent to A (the former has two copies of the formula A). For example, the formula $A \otimes A \otimes (A \multimap B)$

linearly entails $A \otimes B$ (there is still one A left over) but does not entail B (there must still be one A present). The following table provides a summary:

$$
\begin{array}{lrcl}
\text{INCORRECT:} & A & \multimap & (A \otimes A) \\
\text{INCORRECT:} & (A \otimes B) & \multimap & A \\
\text{CORRECT:} & (A \otimes (A \multimap B)) & \multimap & B \\
\text{INCORRECT:} & (A \otimes (A \multimap B)) & \multimap & (A \otimes B) \\
\text{INCORRECT:} & (A \otimes (A \multimap B)) & \multimap & (A \multimap B) \otimes B
\end{array}
$$

In this way, linear logic checks that a formula is used once and only once in a deduction, enforcing the requirement that each component of an utterance contributes exactly once to the assembly of the utterance's meaning.

A direct consequence of the above properties of linear logic is that the constraints of functional *completeness* and *coherence* hold without further stipulation[2] (Dalrymple, Lamping, and Saraswat, 1993). In the present setting, the feature structure f corresponding to the utterance is associated with the $(\otimes)$ conjunction ϕ of all the formulas associated with the lexical items in the utterance. The conjunction is said to be *complete* and *coherent* iff $Th \vdash \phi \multimap f_\sigma \rightsquigarrow t$ (for some term t), where Th is the background theory of general linguistic principles. Each possible t is to be thought of as a valid meaning for the sentence. This guarantees that the entries are used exactly once in building up the denotation of the utterance: no syntactic or semantic requirements may be left unfulfilled, and no meaning may remain unused.

Our glue language needs to be only a fragment of higher-order linear logic, the *tensor fragment*, that is closed under conjunction, universal quantification, and implication. This fragment arises from transferring to linear logic the ideas underlying the concurrent constraint programming scheme of Saraswat (1989).[3]

Our approach shares a number of commonalities with various systems of categorial syntax and semantics. In particular, the Lambek calculus (Lambek, 1958), introduced as a logic of syntactic combination, turns out to be

[2] 'An f-structure is *locally complete* if and only if it contains all the governable grammatical functions that its predicate governs. An f-structure is *complete* if and only if all its subsidiary f-structures are locally complete. An f-structure is *locally coherent* if and only if all the governable grammatical functions that it contains are governed by a local predicate. An f-structure is *coherent* if and only if all its subsidiary f-structures are locally coherent.' (Kaplan and Bresnan, 1982, pages 211–212)

To illustrate:

(a) *John devoured. [incomplete]

(b) *John arrived Bill the sink. [incoherent]

[3] Saraswat and Lincoln (1992) provide an explicit formulation for the higher-order version of the linear concurrent constraint programming scheme. Scedrov (1993) gives a tutorial introduction to linear logic itself; Saraswat (1993) supplies further background on computational aspects of linear logic relevant to the implementation of the present proposal.

a fragment of noncommutative multiplicative linear logic. For a discussion of how our approach compares to the approach of Lambek and related approaches (Moortgat, 1988; Hepple, 1990; Morrill, 1990), see Dalrymple et al. (1994b).

4 Modification

A primary advantage of the use of linear logic 'glue' in the derivation of meanings of sentences is that it enables a clear treatment of modification, as described in Dalrymple, Lamping, and Saraswat (1993). Consider the following sentence, containing the sentential modifier 'obviously':

(9) Bill obviously appointed Hillary.

We make the standard assumption that the verb 'appointed' is the main syntactic predicate of this sentence. The following is the f-structure for example (9):

(10)
$$f\!:\begin{bmatrix} \textsc{pred} & \text{'KISS'} \\ \textsc{subj} & g\!:\!\begin{bmatrix} \textsc{pred} & \text{'BILL'} \end{bmatrix} \\ \textsc{obj} & h\!:\!\begin{bmatrix} \textsc{pred} & \text{'HILLARY'} \end{bmatrix} \\ \textsc{mods} & \left\{\begin{bmatrix} \textsc{pred} & \text{'OBVIOUSLY'} \end{bmatrix}\right\} \end{bmatrix}$$

We also assume that the meaning of the sentence can be represented by the following formula:

(11) $obviously(kiss(Bill, Hillary))$

It is clear that there is a mismatch between the syntactic representation and the meaning of the sentence; syntactically, the verb is the main functor, while the main semantic functor is the adverb.[4]

Recall that linear logic enables a coherent notion of *consumption* and *production* of meanings. We claim that the semantic function of adverbs (and, indeed, of modifiers in general) is to consume the meaning of the structure they modify, producing a new, modified meaning. Note below that the meaning of the modified structure f in the meaning constructor contributed by 'obviously' appears on *both* sides of $\multimap$; the unmodified meaning is consumed, and the modified meaning is produced.

The derivation of the meaning of example (9) is:

bill:	$g_\sigma \rightsquigarrow Bill$
hillary:	$h_\sigma \rightsquigarrow Hillary$
appointed:	$\forall X, Y.\ g_\sigma \rightsquigarrow X \otimes h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow appoint(X, Y)$
obviously:	$(\forall P.\ f_\sigma = P \multimap f_\sigma = obviously(P))$

[4]The related phenomenon of *head switching*, discussed in connection with machine translation by Kaplan et al. (1989) and Kaplan and Wedekind (1993), is also amenable to treatment along the lines presented here.

	bill $\otimes$ **hillary** $\otimes$ **appointed** $\otimes$ **obviously**	(Premises.)
$\vdash$	**bill-appointed** $\otimes$ **hillary** $\otimes$ **obviously**	$X \mapsto Bill$
$\vdash$	$f_\sigma \rightsquigarrow appoint(Bill, Hillary) \otimes$ **obviously**	$Y \mapsto Hillary$
$\vdash$	$obviously(appoint(Bill, Hillary))$	$P \mapsto appoint(Bill, Hillary)$

The first part of the derivation is the same as the derivation for the sentence 'Bill appointed Hillary'. The crucial difference is the presence of information introduced by 'obviously'. In the last step in the derivation, the linear implication introduced by 'obviously' consumes the previous value for f_σ and produces the new and final value.

By using linear logic, each step of the derivation keeps track of what 'resources' have been consumed by linear implications. As mentioned above, the value for f_σ is a meaning for this sentence only if there is no other information left. Thus, the derivation could not stop at the next to last step, because the linear implication introduced by 'obviously' was still left. The final step provides the only complete and coherent meaning derivable for the utterance.

5 Quantification

Our treatment of quantification (Dalrymple et al., 1994a; Dalrymple et al., 1994b; Dalrymple et al., 1995), and in particular of quantifier scope ambiguities and of the interactions between scope and bound anaphora, follows the analysis of Pereira (1990; 1991), but offers in addition a formal account of the syntax-semantics interface, which was treated only informally in that earlier work.

To illustrate our analysis of quantification, we will consider the following sentence:

(12) Bill convinced everyone.

The f-structure for (12) is:

(13)
$$f\!:\begin{bmatrix} \textsc{pred} & \text{`CONVINCE'} \\ \textsc{subj} & g\!:\begin{bmatrix} \textsc{pred} & \text{`BILL'} \end{bmatrix} \\ \textsc{obj} & h\!:\begin{bmatrix} \textsc{pred} & \text{`EVERYONE'} \end{bmatrix} \end{bmatrix}$$

We assume that this example has a meaning representable as:

$$every(person, \lambda z.convince(Bill, z))$$

Here, we will work with the meaning of a quantifier like 'everyone'; for a full exposition of our analysis of quantification and of how a determiner like 'every' combines with a noun like 'person', see Dalrymple et al. (1994b). The quantifier 'everyone' can be seen as making a semantic contribution along the following lines:

(14) $\forall S.\,(\forall x.\,f_\sigma \rightsquigarrow x \multimap \mathit{scope} \rightsquigarrow S(x)) \multimap \mathit{scope} \rightsquigarrow \mathit{every}(\mathit{person}, S)$

Informally, the constructor for 'everyone' can be read as follows: if by giving the arbitrary meaning x of type e to f, the f-structure for 'everyone', we can derive the meaning $S(x)$ of type t for the scope of quantification *scope*, then S can be the property that the quantifier requires as its scope, yielding the meaning $\mathit{every}(\mathit{person}, S)$ for *scope*. The quantified NP can thus be seen as providing two contributions to an interpretation: locally, a *referential import* x, which must be discharged when the scope of quantification is established; and globally, a *quantificational import* of type $(e \to t) \to t$, which is applied to the meaning of the scope of quantification to obtain a quantified proposition. Notice that the assignment of a meaning to *scope* appears on both sides of the implication, as in the case of modifiers, and that the meaning is not the same in the two instances.

We use the place-holder *scope* in (14) to represent possible choices for the scope of the quantifier, but we did not specify how this scope was chosen. Previous work on scope determination in LFG (Halvorsen and Kaplan, 1988) defined possible scopes at the f-structure level, using *inside-out functional uncertainty* to nondeterministically choose a scope f-structure for quantified noun phrases. That approach requires the scope of a quantified NP to be an f-structure which contains the NP f-structure. In contrast, our approach depends only on the logical form of semantic constructors to yield exactly the appropriate scope choices. Within the constraints imposed by that logical form, the actual scope can be freely chosen; the linear implication guarantees that the scope will contain the quantified NP, since only scope meanings which are obtained by consuming the variable representing the quantified NP can be chosen.

Logically, this means that the semantic constructor for an NP quantifies universally over semantic projections of possible scopes, as follows:

(15) **everyone**: $\forall H, S.\,(\forall x.\,f_\sigma \rightsquigarrow x \multimap H \rightsquigarrow S(x)) \multimap H \rightsquigarrow \mathit{every}(\mathit{person}, S)$

The premises for the derivation of the meaning of example (12) are the meaning constructors for 'Bill', 'convinced', and 'everyone':

bill: $g_\sigma \rightsquigarrow \mathit{Bill}$

convinced: $\forall X, Y.\; g_\sigma \rightsquigarrow X \otimes h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow \mathit{convince}(X, Y)$

everyone: $\forall H, S.\;(\forall x.\,h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow_t S(x)) \multimap H \rightsquigarrow_t \mathit{every}(\mathit{person}, S)$

Notice that we have explicitly indicated that the scope of the quantifier must be of type t, by means of the subscript t on the 'means' relation $\rightsquigarrow_t$. This typing is implicit in the schematic formula for quantifiers given in (14).

Giving the name **bill-convinced** to the formula

bill-convinced: $\forall Y.\,h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow \mathit{convince}(\mathit{Bill}, Y)$

we have the following derivation:

	bill $\otimes$ **convinced** $\otimes$ **everyone**	(Premises.)
$\vdash$	**bill-convinced** $\otimes$ **everyone**	$X \mapsto Bill$
$\vdash$	$f_\sigma \rightsquigarrow every(person, \lambda z.convince(Bill, z))$	$H \mapsto f_\sigma, Y \mapsto x$ $S \mapsto \lambda z.convince(Bill, z)$

The formula **bill-convinced** represents the semantics of the scope of the determiner 'every'. No derivation of a different formula $f_\sigma \rightsquigarrow_t P$ is possible.

While the formula

$$\forall Y.\ h_\sigma \rightsquigarrow_e Y \multimap h_\sigma \rightsquigarrow_e Y$$

could at first sight be considered another possible scope, the type subscripting of the $\rightsquigarrow$ relation used in the determiner lexical entry requires the scope to represent a dependency of a proposition on an individual. But this formula represents the dependency of an individual on an individual (itself). Therefore, it does not provide a valid scope for the quantifier.

6 Quantifier scope ambiguities

When a sentence contains more than one quantifier, scope ambiguities are of course possible. In our system, those ambiguities will appear as alternative successful derivations. We will take as our example this sentence:[5]

(16) Every candidate appointed a manager.

The f-structure for sentence (16) is:

(17)
$$f\!:\begin{bmatrix} \textsc{pred} & \text{'APPOINT'} \\ \textsc{subj} & g\!:\begin{bmatrix} \textsc{spec} & \text{'EVERY'} \\ \textsc{pred} & \text{'CANDIDATE'} \end{bmatrix} \\ \textsc{obj} & h\!:\begin{bmatrix} \textsc{spec} & \text{'A'} \\ \textsc{pred} & \text{'MANAGER'} \end{bmatrix} \end{bmatrix}$$

The meaning constructors for 'every candidate' and 'a manager' are analogous to the one for 'everyone' in the previous section. The derivation proceeds from those contributions together with the contribution of 'appointed':

every-candidate:	$\forall G, R.\ (\forall x.\ g_\sigma \rightsquigarrow x \multimap G \rightsquigarrow R(x)) \multimap G \rightsquigarrow every(candidate, R)$
a-manager:	$\forall H, S.\ (\forall y.\ h_\sigma \rightsquigarrow y \multimap H \rightsquigarrow S(y)) \multimap H \rightsquigarrow a(manager, S)$
appointed:	$\forall X, Y.\ g_\sigma \rightsquigarrow X \otimes h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow appoint(X, Y)$

[5] To allow for apparent scope ambiguities, we adopt a scoping analysis of indefinites, as proposed, for example, by Neale (1990).

As of yet, we have not made any commitment about the scopes of the quantifiers; the scope and scope meaning variables in **every-candidate** and **a-manager** have not been instantiated. Scope ambiguities are manifested in two different ways in our system: through the choice of different semantic structures G and H, corresponding to different scopes for the quantified NPs, or through different relative orders of application for quantifiers that scope at the same point. For this example, the second case is relevant, and we must now make a choice to proceed. The two possible choices correspond to two equivalent rewritings of **appointed**:

$$\textbf{appointed}_1\colon\ \forall X.\, g_\sigma \rightsquigarrow X \multimap (\forall Y.\, h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow \mathit{appoint}(X,Y))$$
$$\textbf{appointed}_2\colon\ \forall Y.\, h_\sigma \rightsquigarrow Y \multimap (\forall X.\, g_\sigma \rightsquigarrow X \multimap f_\sigma \rightsquigarrow \mathit{appoint}(X,Y))$$

These two equivalent forms correspond to the two possible ways of 'currying' a two-argument function $f : \alpha \times \beta \to \gamma$ as one-argument functions:

$$\lambda u.\lambda v.f(u,v) : \alpha \to (\beta \to \gamma)$$
$$\lambda v.\lambda u.f(u,v) : \beta \to (\alpha \to \gamma)$$

We select 'a manager' to take narrower scope by using the variable instantiations

$$H \mapsto f_\sigma, Y \mapsto y, S \mapsto \lambda v.\mathit{appoint}(X,v)$$

and transitivity of implication to combine $\textbf{appointed}_1$ with **a-manager** into:

$$\textbf{appointed-a-manager}\colon\ \forall X.\ g_\sigma \rightsquigarrow X \multimap f_\sigma \rightsquigarrow_t a(\mathit{manager}, \lambda v.\mathit{appoint}(X,v))$$

This gives the derivation

$$\begin{aligned} & \textbf{every-candidate} \otimes \textbf{appointed}_1 \otimes \textbf{a-manager} \\ \vdash\ & \textbf{every-candidate} \otimes \textbf{appointed-a-manager} \\ \vdash\ & f_\sigma \rightsquigarrow_t \mathit{every}(\mathit{candidate}, \lambda u.a(\mathit{manager}, \lambda v.\mathit{appoint}(u,v))) \end{aligned}$$

of the $\forall\exists$ reading of (16), where the last step uses the substitutions

$$G \mapsto f_\sigma, X \mapsto x, R \mapsto \lambda u.a(\mathit{manager}, \lambda v.\mathit{appoint}(u,v))$$

Alternatively, we could have chosen 'every candidate' to take narrow scope, by combining $\textbf{appointed}_2$ with **every-candidate** to produce:

$$\textbf{every-candidate-appointed}\colon\ \forall Y.\ h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow_t \mathit{every}(\mathit{candidate}, \lambda u.\mathit{appoint}(u,Y))$$

This gives the derivation

$$\begin{aligned} & \textbf{every-candidate} \otimes \textbf{appointed}_2 \otimes \textbf{a-manager} \\ \vdash\ & \textbf{every-candidate-appointed} \otimes \textbf{a-manager} \\ \vdash\ & f_\sigma \rightsquigarrow_t a(\mathit{manager}, \lambda v.\mathit{every}(\mathit{candidate}, \lambda u.\mathit{appoint}(u,v))) \end{aligned}$$

for the $\exists\forall$ reading. These are the only two possible outcomes of the derivation of a meaning for (16), as required (Dalrymple et al., 1994b).

7 Conclusion

Our approach exploits the f-structure of LFG for syntactic information needed to guide semantic composition, and also exploits the resource-sensitive properties of linear logic to express the semantic composition requirements of natural language. The use of linear logic as the glue language in a deductive semantic framework allows a natural treatment of the syntax-semantics interface which automatically gives the right results for completeness and coherence constraints and for modification (Dalrymple, Lamping, and Saraswat, 1993), covers quantifier scoping and bound anaphora (Dalrymple et al., 1994b; Dalrymple et al., 1995) and their interactions with intensionality (Dalrymple et al., 1994a), offers a clean and natural treatment of complex predicate formation (Dalrymple et al., 1993), and extends nicely to an analysis of the semantics of structure sharing (Dalrymple et al., 1995).

8 Acknowledgements

Over the (rather lengthy) course of the development of the approach outlined here, we have greatly benefited from discussion and assistance from many of our colleagues; we are particularly grateful to Andy Kehler, Angie Hinrichs, Ron Kaplan, John Maxwell, Kris Halvorsen, Joan Bresnan, Stanley Peters, Paula Newman, Chris Manning, Henriette de Swart, Johan van Benthem, and David Israel for very helpful comments as well as for incisive criticism.

References

Bresnan, Joan, editor. 1982. *The Mental Representation of Grammatical Relations*. The MIT Press, Cambridge, MA.

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, Mary, Andrew Kehler, John Lamping, and Vijay Saraswat. 1995. The semantics of resource-sharing in Lexical-Functional Grammar. In *Proceedings of the Seventh Meeting of the European ACL*, University College Dublin, March. European Chapter of the Association for Computational Linguistics.

Dalrymple, Mary, John Lamping, Fernando C. N. Pereira, and Vijay Saraswat. 1994a. Intensional verbs without type-raising or lexical am-

biguity. In *Conference on Information-Oriented Approaches to Logic, Language and Computation*, Moraga, California. Saint Mary's College.

Dalrymple, Mary, John Lamping, Fernando C. N. Pereira, and Vijay Saraswat. 1994b. Quantifiers, anaphora, and intensionality. MS, Xerox PARC and AT&T Bell Laboratories.

Dalrymple, Mary, John Lamping, Fernando C. N. Pereira, and Vijay Saraswat. 1995. A deductive account of quantification in LFG. In Makoto Kanazawa, Christopher J. Piñón, and Henriette de Swart, editors, *Quantifiers, Deduction, and Context*. Center for the Study of Language and Information, Stanford, California.

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.

Fenstad, Jens Erik, Per-Kristian Halvorsen, Tore Langholm, and Johan van Benthem. 1987. *Situations, Language and Logic*. D. Reidel, Dordrecht.

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

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

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*, pages 1116–1122, Tokyo, Japan. Institute for New Generation Systems.

Hepple, Mark. 1990. *The Grammar and Processing of Order and Dependency: a Categorial Approach*. Ph.D. thesis, University of Edinburgh.

Kaplan, Ronald M. 1987. Three seductions of computational psycholinguistics. In Peter Whitelock, Harold Somers, Paul Bennett, Rod Johnson, and Mary McGee Wood, editors, *Linguistic Theory and Computer Applications*. Academic Press, London, pages 149–188.

Kaplan, Ronald M. and Joan Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*. The MIT Press, Cambridge, MA, pages 173–281.

Kaplan, Ronald M., Klaus Netter, Jürgen Wedekind, and Annie Zaenen. 1989. Translation by structural correspondences. In *Proceedings of the Fourth Meeting of the European ACL*, pages 272–281, University of Manchester, April. European Chapter of the Association for Computational Linguistics.

Kaplan, Ronald M. and Jürgen Wedekind. 1993. Restriction and correspondence-based translation. In *Proceedings of the Sixth Meeting of the European ACL*, University of Utrecht, April. European Chapter of the Association for Computational Linguistics.

Klein, Ewan and Ivan A. Sag. 1985. Type-driven translation. *Linguistics and Philosophy*, 8:163–201.

Lambek, Joachim. 1958. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170.

Levin, Lori S., Malka Rappaport, and Annie Zaenen, editors. 1983. *Papers in Lexical-Functional Grammar*. Indiana University Linguistics Club, Bloomington, IN.

McCloskey, James. 1979. *Transformational syntax and model theoretic semantics : a case study in modern Irish*. D. Reidel, Dordrecht.

Montague, Richard. 1974. *Formal Philosophy*. Yale University Press, New Haven. Richmond Thomason, editor.

Moortgat, Michael. 1988. *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus*. Ph.D. thesis, University of Amsterdam, Amsterdam, The Netherlands.

Morrill, Glyn V. 1990. Intensionality and boundedness. *Linguistics and Philosophy*, 13(6):699–726.

Neale, Stephen. 1990. *Descriptions*. The MIT Press, Cambridge, MA.

Pereira, Fernando C. N. 1990. Categorial semantics and scoping. *Computational Linguistics*, 16(1):1–10.

Pereira, Fernando C. N. 1991. Semantic interpretation as higher-order deduction. In Jan van Eijck, editor, *Logics in AI: European Workshop JELIA '90*, pages 78–96, Amsterdam, Holland. Springer-Verlag.

Pollard, Carl and Ivan A. Sag. 1987. *Information-Based Syntax and Semantics, Volume I*. Number 13 in CSLI Lecture Notes. Center for the Study of Language and Information, Stanford University.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago.

Reyle, Uwe. 1988. Compositional semantics for LFG. In Uwe Reyle and Christian Rohrer, editors, *Natural language parsing and linguistic theories*. D. Reidel, Dordrecht.

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

Saraswat, Vijay A. 1993. A brief introduction to linear concurrent constraint programming. Technical report, Xerox Palo Alto Research Center, April.

Saraswat, Vijay A. and Patrick Lincoln. 1992. Higher-order, linear concurrent constraint programming. Technical report, Xerox Palo Alto Research Center, August.

Scedrov, Andre. 1993. A brief guide to linear logic. In G. Rozenberg and A. Salomaa, editors, *Current Trends in Theoretical Computer Science.* 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.

Simpson, Jane. 1991. *Warlpiri Morpho-Syntax: A Lexicalist Approach.* Kluwer Academic Publishers, Dordrecht.