Linear Logic for Meaning Assembly

Mary Dalrymple[1], John Lamping[1], Fernando Pereira[2], and Vijay Saraswat[2]

[1] Xerox PARC, Palo Alto CA 94304 USA; {dalrymple,lamping}@parc.xerox.com
[2] AT&T Laboratories, Murray Hill NJ 07974 USA; {pereira,vj}@research.att.com

Abstract. Semantic theories of natural language associate meanings with utterances by providing meanings for lexical items, together with *composition* rules for computing the meanings of larger units from the meanings of their parts. Traditionally, this *semantic compositionality* has been expressed in terms of functional abstraction, functional application and derived operators such as function composition. This form of semantic composition relies on tree-structured syntactic analyses to identify function-argument relations and specify the order of function application. However, a variety of linguistic arguments, including cross-linguistic evidence, support the view that unordered attribute-value structures, in particular the *functional structures* of Lexical-Functional Grammar, are more suitable for representing the main meaning-determining structural relationships in sentences. But such analyses do not provide the tree-structured scaffolding demanded by traditional compositionality. To solve this problem, we propose a deductive approach to meaning assembly in which the meanings associated with attribute-value structures are specified by constraints stated in a resource logic, *linear logic*. Linear logic is the right language to specify requirements on the composition of the semantic contribution of words without relying on particular phrase structure relations to guide the composition. The resource sensitivity of linear logic also enables a unified treatment of the LFG requirements of completeness and coherence as well as of modification and quantification.

1 The Framework

This paper is 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, Hinrichs, Lamping, and Saraswat 1993; Kehler, Dalrymple, Lamping, and Saraswat 1995; Dalrymple, Lamping, Pereira, and Saraswat 1995, 1996a, 1996b). We use *linear logic* (Girard 1987) to represent the connection between two dissimilar linguistic levels: LFG f-structures and their contributions to sentence meaning.

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 is not a good match with traditional notions of semantic compositionality, which are based on functional abstraction and application and so depend on a rigid order of semantic composition. We are thus led to a more relaxed form of compositionality. As in more traditional compositionality, the semantic contribution of each lexical entry in a sentence is used exactly once in interpretation in a way constrained by syntactic relations. In contrast with that tradition, however, constituent structure will play no role in managing the combination of semantic contributions. Instead, we rely on constraints on formula use which are built into linear logic and on the shape of the linear-logic formulas describing semantic contributions. Approaches to semantic interpretation that encode semantic representations in attribute-value structures (Fenstad et al. 1987, Pollard and Sag 1994) also provide a relaxation of compositionality, but are unable to properly represent essential constraints between binding and scope (Pereira 1990, Dalrymple, Lamping, Pereira, and Saraswat 1996b).

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 but does not force a particular hierarchical order of composition except as required by the properties of particular lexical entries.

1.1 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 then a convenient formalism for assembling the semantics along that scaffolding (Montague 1974). The reason for this is that the derivation of the meaning of a phrase often mirrors the surface constituent structure of the English phrase. For example, the sentence 'Bill appointed Hillary' has the surface constituent structure indicated by the bracketing in (1):

(1) $[_{\mathrm{S}}\ [_{\mathrm{NP}}$ Bill$]\ [_{\mathrm{VP}}$ appointed $[_{\mathrm{NP}}$ 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. In a VSO language such as Irish (McCloskey 1979), for example, the verb does not appear adjacent to the object. In a free word order language such as Warlpiri (Simpson 1991), surface constituent structure does not always give rise to units that are semantically coherent or useful: an argument of a verb may not even appear as a single unit at surface constituent structure, and arguments of a verb may appear in various different places in the string:

(2) Kurdu-jarra-rlu wita-jarra-rlu ka-pala maliki wajilipi-nyi.
child-ERG small-ERG Pres dog-ABS chase
'The small children are chasing the dog.'

The word glossed ‘Pres’ is the present tense auxiliary, which must always appear in second position in the clause. ERG and ABS casemarkers appear on the subject and object of the sentence, respectively. As Simpson notes, any permutation of the words in this sentence is acceptable and conveys the same meaning, as long as the Pres auxiliary remains in second position; in particular, the words of the subject noun phrase need not appear adjacent to each other:

(3)	Kurdu-jarra-rlu	ka-pala	maliki	wajilipi-nyi	wita-jarra-rlu.
	child-ERG	Pres	dog-ABS	chase	small-ERG

‘The two small children are chasing the dog.’

Simpson (1991) argues convincingly for a flat phrase structure for Warlpiri sentences like (2) and (3). It is difficult to see how to provide an account of semantic composition based on a flat constituent structure, where distinctions between subject and object are not made, and where arguments like the subject and object need not form a cohesive constituent.

The observation that surface constituent structure does not always provide the optimal set of constituents or hierarchical structure to guide semantic interpretation has motivated 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 underlying, abstract syntactic structure for example (2) and all permissible permutations of the words in it is represented by the f-structure in (4):

(4)
$$\left[\begin{array}{ll} \text{PRED} & \text{‘CHASE’} \\ \text{SUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{‘CHILDREN’} \\ \text{MODS} & \left\{\left[\begin{array}{ll} \text{PRED} & \text{‘SMALL’} \end{array}\right]\right\} \end{array}\right] \\ \text{OBJ} & \left[\begin{array}{ll} \text{PRED} & \text{‘KANGAROO’} \end{array}\right] \end{array}\right]$$

As illustrated, an f-structure represents the more abstract syntactic function-argument structure of the sentence, encoding relations such as SUBJect and OBJect. Those relations can realized in different constituent structure forms in different languages and within a single language like Warlpiri, 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. For perspicuity, here and in the following we omit details of syntactic structure in the f-structures we present, providing only skeletal f-structure representations.

We will not provide a detailed account of the syntactic assumptions of LFG, the nature of constituent structure and f-structure, and the relation between them; for such an account, see Bresnan (1982), Levin, Rappaport, and Zaenen (1983), Dalrymple et al. (1995), 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.

1.2 Semantics

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, which we will call *meaning constructors*, to a logical deduction. A meaning constructor is a linear logic formula that can be understood as 'instructions' for obtaining the meaning of the f-structure headed by the entry by combining the meanings of the lexical entry and its syntactic arguments. In effect, then, our approach uses linear logic as the 'glue' with which semantic representations are tied together. Once all these constraints are gathered, deduction in the logic is used to obtain a meaning for the entire structure. Throughout this process we maintain a clear distinction between assertions about the meaning and the meaning itself.

We will assume that the f-structure for a sentence like 'Bill appointed Hillary' is as given in (5):

(5)
$$\begin{bmatrix} \textsc{pred} & \text{`}\textsc{appoint}\text{'} \\ \textsc{subj} & \begin{bmatrix} \textsc{pred} & \text{`}\textsc{Bill}\text{'} \end{bmatrix} \\ \textsc{obj} & \begin{bmatrix} \textsc{pred} & \text{`}\textsc{Hillary}\text{'} \end{bmatrix} \end{bmatrix}$$

Following work by Kaplan (1987) and Halvorsen and Kaplan (1988), we assume a level of linguistic representation which we will call a *semantic* or *σ-structure*, and we make use of a *semantic* or *σ-projection* function σ to map f-structures to the semantic structures that correspond to them. A distinction between f-structures and their semantic projections is made because semantic projections can carry more information than just the association to the meaning for the corresponding f-structure; see Dalrymple, Lamping, Pereira, and Saraswat (1996b) for more discussion. We also assume that semantic structures stand in a direct relation to meanings; this relation will be represented by the symbol $\rightsquigarrow$. The configuration we assume is illustrated by the following:

(6)
$$h : \begin{bmatrix} \textsc{pred} & \text{`}\textsc{Bill}\text{'} \end{bmatrix} \qquad h_\sigma : [\;] \rightsquigarrow \mathit{Bill}$$

We will use labels such as h to refer to particular f-structures: here, h is the f-structure for a particular use of the proper noun 'Bill'. The semantic projection function σ, relating f-structures to semantic structures, is represented by an arrow from the f-structure h to the corresponding semantic structure h_σ.

The association between h_σ and the constant *Bill* representing its meaning will be written as:

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

More generally, the association between a semantic structure h_σ and a meaning P will be represented by the meaning constructor $h_\sigma \rightsquigarrow P$, 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 τ of the intended second argument, although for simplicity we will omit the type subscript whenever it is determinable from context.

In the following, we will use simple predicate calculus expressions to represent meanings, ignoring many issues of semantic detail, including (among other things) the representation of tense and aspect. It would of course be possible to use a different, more expressive language for expressing meanings. Dalrymple, Lamping, Pereira, and Saraswat (1996b) provide a brief discussion of the use of Discourse Representation Structures, and van Genabith and Crouch (1997) explore the use of a dynamic meaning representation language for DRT (Muskens 1994) within the current framework.

The meaning constructor for a verb is a formula consisting of instructions on how to assemble the meaning of a sentence, given the meanings of its arguments. For example, the verb 'appointed' contributes the following f-structure and meaning constructor:

(7)
$$f : \begin{bmatrix} \textsc{pred} & \text{`APPOINT'} \\ \textsc{subj} & [\,] \\ \textsc{obj} & [\,] \end{bmatrix} \curvearrowright f_\sigma : [\,]$$

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

The meaning constructor, given in the last line of (7), 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 $appoint(X, Y)$.

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 their classical counterparts conjunction $\wedge$ and implication $\rightarrow$. We will soon see, however, 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:

(8)
$$f{:} \begin{bmatrix} \textsc{pred} & \text{`APPOINT'} \\ \textsc{subj} & g{:}\begin{bmatrix} \textsc{pred} & \text{`Bill'} \end{bmatrix} \\ \textsc{obj} & h{:}\begin{bmatrix} \textsc{pred} & \text{`Hillary'} \end{bmatrix} \end{bmatrix}$$

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

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

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

bill-appointed: $\forall Y.\ h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow appoint(Bill, Y)$

Then the following derivation is possible in linear logic ($\vdash$ stands for the linear-logic entailment relation, and $\mapsto$ specifies the variable substitutions in implicit universal instantiation steps):

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

Another derivation with the same result is also possible. Assume that the formula **appointed-hillary** is defined as:

appointed-hillary: $\forall X.\ g_\sigma \rightsquigarrow X \multimap f_\sigma \rightsquigarrow appoint(X, Hillary)$

Then we have the following derivation:

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

These two deductions are essentially the same, in that premises are satisfied in the same way, and thus they both yield the same conclusion. In the following, we will see cases of essentially different derivations yielding different conclusions.

The process of semantic derivation may be taken to be simultaneous with the process of syntactic analysis. As the feature structure of the utterance is elaborated, the formulas associated with the semantic projections may be assembled. For instance, once $(f\ \textsc{subj})$ has been determined to exist in the functional analysis, the contribution $(f\ \textsc{subj})_\sigma \rightsquigarrow Bill$ may be made available to the semantic analysis. Note, then, that the interface between the unification-based processing of LFG and the linear-logic based deduction of semantic assembly is very narrow: the machinery of LFG is used merely to identify the f-structure; the semantic deductions are carried out parallel to that through the projection of the f-structures into the σ-structures. In particular, there is no feedback from the process of semantic assembly to the process of f-structure construction. The syntactic analysis of LFG proceeds entirely unaffected by semantic assembly.[3]

[3] It may be technically possible to carry out the entire LFG syntactic analysis in the context of deduction in linear logic; see Oehrle (1995) and Johnson (1996) for proposals along these lines. However, this is not necessary for the validity of the current approach to semantic assembly via linear logic.

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.

2 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 logical truths that may be called upon in the derivation of the meaning of the sentence either once, frequently, or not at all, depending on the course of the derivation. Instead, the semantic contributions are *occurrences* of information that are generated and used exactly once. For example, the formula $g_\sigma \rightsquigarrow Bill$ 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 (9)) 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).

A direct consequence of these properties of linear logic is that the constraints of functional *completeness* and *coherence* hold without further stipulation. Kaplan and Bresnan (1982, pages 211–212) define completeness and coherence as follows:

> 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.

Incomplete f-structures are missing some necessary arguments:

(11) *John devoured. [incomplete]

Incoherent sentences contain additional arguments not required by the predicate:

(12) *John arrived Bill. [incoherent]

Dalrymple, Lamping, and Saraswat (1993) note that completeness and coherence can be defined as a condition on deductions. The feature structure f for any utterance is associated with the ($\otimes$) conjunction ϕ of all the formulas associated with the lexical items in the utterance. A derivation for f with premises ϕ is said to be *complete* and *coherent* iff

$$Th, \phi \vdash f_\sigma \rightsquigarrow t \text{ (for some term } t)$$

where Th is a background theory of general linguistic principles, general linguistic rules that can be used as often or as seldom as is necessary. Each possible t is to be thought of as a valid meaning for the sentence.

Consider, for example, the f-structure and meaning constructors for a sentence like (11):

(13)
$$f : \begin{bmatrix} \text{PRED} & \text{`DEVOUR'} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{`JOHN'} \end{bmatrix} \end{bmatrix}$$

devour: $\forall X, Y.\ (f\ \text{SUBJ})_\sigma \rightsquigarrow X \otimes (f\ \text{OBJ})_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow devour(X, Y)$
John: $(f\ \text{SUBJ})_\sigma \rightsquigarrow John$

Taking the meaning constructors in (13) as premises, we can perform the following deduction:

(14)
$$\begin{array}{ll} & \textbf{devour} \otimes \textbf{John} \\ \vdash & \forall Y.\ (f\ \text{OBJ})_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow devour(John, Y) \end{array}$$

This is not a complete and coherent deduction, since the result of the deduction is not a term of the form $f_\sigma \rightsquigarrow t$.

Now consider the f-structure and meaning constructors for a sentence like (12):

(15)
$$f : \begin{bmatrix} \text{PRED} & \text{'ARRIVE'} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'JOHN'} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{'BILL'} \end{bmatrix} \end{bmatrix}$$

arrive: $\forall X.\ (f\ \text{SUBJ})_\sigma \rightsquigarrow X \multimap f_\sigma \rightsquigarrow arrive(X)$
John: $(f\ \text{SUBJ})_\sigma \rightsquigarrow John$
Bill: $(f\ \text{OBJ})_\sigma \rightsquigarrow Bill$

Taking these meaning constructors as premises, we can perform the following deduction:

(16) $\quad$ **arrive** $\otimes$ **John** $\otimes$ **Bill**
$\vdash \quad f_\sigma \rightsquigarrow arrive(John) \otimes (f\ \text{OBJ})_\sigma \rightsquigarrow Bill$

Here too, the deduction is not complete and coherent, since the result of the deduction is a conjunction of terms, rather than a single term of the form $f_\sigma \rightsquigarrow t$. Thus, the completeness and coherence condition guarantees that the entries are used exactly once in building up the meaning 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). In this setting, meaning assembly is a process of an underlying collection of 'agents' interacting with each other according to their explicit internal structure. Computationally, roles act as *channels* on which data (the associated meaning) is produced and consumed by several simultaneously active agents. A linear implication $\forall X.r \rightsquigarrow X \multimap B$ acts like an agent (an 'ask' agent) that is suspended on the appearance of some meaning m on the channel r; it then reduces to B (with X substituted with m). An agent $r \rightsquigarrow m$ acts as an agent posting the meaning m on the channel r; this meaning may be consumed by any agent suspended on r. Using linear logic rather than intuitionistic logic (for which such a concurrent interpretation is also valid) allows multiplicity of messages sent on these channels to be tracked, accounting for completeness and coherence. Saraswat and Lincoln (1992) provide an explicit formulation for the higher-order version of the linear concurrent constraint programming scheme. Scedrov (1993) and Troelstra (1993) give 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.

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 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, Lamping, Pereira, and Saraswat (1996b).

3 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). Intuitively, the function of modifiers is to consume the meaning of the phrase they modify and produce a new, modified meaning of the same semantic shape, which can play the same semantic role as the unmodified phrase can play. Sentential modification provides a simple case of this generalization.

Consider the following sentence, containing the sentential modifier 'obviously', and its f-structure:

(17) Bill obviously appointed Hillary.

(18)
$$f\colon\begin{bmatrix} \textsc{pred} & \text{`}\textsc{kiss}\text{'} \\ \textsc{subj} & g\colon\big[\,\textsc{pred} \quad \text{`}\textsc{Bill}\text{'}\,\big] \\ \textsc{obj} & h\colon\big[\,\textsc{pred} \quad \text{`}\textsc{Hillary}\text{'}\,\big] \\ \textsc{mods} & \Big\{\big[\,\textsc{pred} \quad \text{`}\textsc{obviously}\text{'}\,\big]\Big\} \end{bmatrix}$$

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

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

Clearly, 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. 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.

Our analysis appeals to the fact that linear logic enables a coherent notion of *consumption* and *production* of meanings. Note that the meaning of the modified structure f in the meaning constructor contributed by 'obviously' appears on *both* sides of $\multimap$:

$$\begin{array}{ll} \textbf{bill}: & g_\sigma \rightsquigarrow \mathit{Bill} \\ \textbf{hillary}: & h_\sigma \rightsquigarrow \mathit{Hillary} \\ \textbf{appointed}: & \forall X, Y.\ g_\sigma \rightsquigarrow X \otimes h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow appoint(X, Y) \\ \textbf{obviously}: & (\forall P.\ f_\sigma \rightsquigarrow P \ \multimap\ f_\sigma \rightsquigarrow obviously(P)) \end{array}$$

$$\begin{array}{lll} & \textbf{bill} \otimes \textbf{hillary} \otimes \textbf{appointed} \otimes \textbf{obviously} & \text{(Premises.)} \\ \vdash & \textbf{bill-appointed} \otimes \textbf{hillary} \otimes \textbf{obviously} & X \mapsto \mathit{Bill} \\ \vdash & f_\sigma \rightsquigarrow appoint(\mathit{Bill}, \mathit{Hillary}) \otimes \textbf{obviously} & Y \mapsto \mathit{Hillary} \\ \vdash & f_\sigma \rightsquigarrow obviously(appoint(\mathit{Bill}, \mathit{Hillary})) & P \mapsto appoint(\mathit{Bill}, \mathit{Hillary}) \end{array}$$

The first part of the derivation of the meaning of example (17) 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. 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’ is still left. The final step provides the only complete and coherent meaning derivable for the utterance.

4 Quantification

Our treatment of quantification (Dalrymple, Lamping, Pereira, and Saraswat 1995, 1996a, 1996b), 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:

(20) Bill convinced everyone.

The f-structure for (20) is:

(21)
$$f{:}\begin{bmatrix} \textsc{pred} & \text{‘}\textsc{convince}\text{’} \\ \textsc{subj} & g{:}\begin{bmatrix}\textsc{pred} & \text{‘}\textsc{Bill}\text{’}\end{bmatrix} \\ \textsc{obj} & h{:}\begin{bmatrix}\textsc{pred} & \text{‘}\textsc{everyone}\text{’}\end{bmatrix} \end{bmatrix}$$

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, Lamping, Pereira, and Saraswat (1996b). The quantifier ‘everyone’ can be seen as making a semantic contribution along the following lines:

(22) $\forall S.\ (\forall x.\ f_\sigma \rightsquigarrow x \multimap scope \rightsquigarrow S(x)) \multimap scope \rightsquigarrow every(person, S)$

Informally, the meaning constructor for ‘everyone’ can be read as follows: if by giving the arbitrary meaning x to the semantic projection of f, the f-structure for ‘everyone’, we can derive the meaning $S(x)$ for the scope of quantification $scope$, then S can be the property that the quantifier requires as its scope, yielding the meaning $every(person, S)$ for $scope$. 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 (22) 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 by using *inside-out functional uncertainty* to nondeterministically choose a scope structure for quantified noun phrases. Thus, the scope of a quantified NP is required to correspond to 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, without having to state that requirement independently. 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. For further discussion of constraints on quantifier scope, see Dalrymple, Lamping, Pereira, and Saraswat (1996b).

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

(23) **everyone**: $\forall H, S.\ (\forall x.\ f_\sigma \rightsquigarrow x \multimap H \rightsquigarrow S(x)) \multimap H \rightsquigarrow every(person, S)$

The premises for the derivation of the meaning of example (20) are the meaning constructors for 'Bill', 'convinced', and 'everyone':

$$\begin{array}{ll}
\textbf{bill}: & g_\sigma \rightsquigarrow Bill \\
\textbf{convinced}: & \forall X, Y.\ g_\sigma \rightsquigarrow X \otimes h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow convince(X, Y) \\
\textbf{everyone}: & \forall H, S.\ (\forall x.\ h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow_t S(x)) \\
& \quad \multimap H \rightsquigarrow_t every(person, S)
\end{array}$$

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 (22).

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

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

we have the following derivation:

$$\begin{array}{lll}
& \textbf{bill} \otimes \textbf{convinced} \otimes \textbf{everyone} & \text{(Premises.)} \\
\vdash & \textbf{bill-convinced} \otimes \textbf{everyone} & X \mapsto Bill \\
\vdash & f_\sigma \rightsquigarrow every(person, \lambda z.convince(Bill, z)) & H \mapsto f_\sigma, Y \mapsto x, \\
& & S \mapsto \lambda z.convince(Bill, z)
\end{array}$$

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 scope of suitable type for the quantifier.

5 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:[4]

(24) Every candidate appointed a manager.

The f-structure for sentence (24) is:

(25)
$$f\!:\!\begin{bmatrix} \text{PRED} & \text{`APPOINT'} \\ \text{SUBJ} & g\!:\!\begin{bmatrix} \text{SPEC} & \text{`EVERY'} \\ \text{PRED} & \text{`CANDIDATE'} \end{bmatrix} \\ \text{OBJ} & h\!:\!\begin{bmatrix} \text{SPEC} & \text{`A'} \\ \text{PRED} & \text{`MANAGER'} \end{bmatrix} \end{bmatrix}$$

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':

$$\begin{array}{ll} \textbf{every-candidate}: & \forall G, R.\ (\forall x.\ g_\sigma \rightsquigarrow x \multimap G \rightsquigarrow R(x)) \\ & \quad \multimap G \rightsquigarrow every(candidate, R) \\ \textbf{a-manager}: & \forall H, S.\ (\forall y.\ h_\sigma \rightsquigarrow y \multimap H \rightsquigarrow S(y)) \\ & \quad \multimap H \rightsquigarrow a(manager, S) \\ \textbf{appointed}: & \forall X, Y.\ g_\sigma \rightsquigarrow X \otimes h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow appoint(X, Y) \end{array}$$

We have not yet 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**:

$$\begin{array}{l} \textbf{appointed}_1: \forall X.\ g_\sigma \rightsquigarrow X \multimap (\forall Y.\ h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow appoint(X, Y)) \\ \textbf{appointed}_2: \forall Y.\ h_\sigma \rightsquigarrow Y \multimap (\forall X.\ g_\sigma \rightsquigarrow X \multimap f_\sigma \rightsquigarrow appoint(X, Y)) \end{array}$$

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

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

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

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

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

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

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

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

This gives the derivation

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

of the $\forall\exists$ reading of (24), where the last step uses the substitutions

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

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

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

This gives the derivation

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

for the $\exists\forall$ reading. As required, these are the only two possible outcomes of the derivation of a meaning for (24); see Dalrymple, Lamping, Pereira, and Saraswat (1996b) for further discussion.

6 Conclusion

Our approach exploits the f-structure of LFG for syntactic information needed to guide semantic composition, and also exploits the resource-sensitivity 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 and their interactions with intensionality (Dalrymple, Lamping, Pereira, and Saraswat 1995, 1996a, 1996b), offers a clean and natural treatment of complex predicate formation (Dalrymple, Hinrichs, Lamping, and Saraswat 1993), and also extends readily to an analysis of the semantics of shared structures in coordination (Kehler, Dalrymple, Lamping, and Saraswat 1995).

7 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, Dick Oehrle, Paula Newman, Chris Manning, Henriette de Swart, Johan van Benthem, David Israel, Michael Moortgat, Glyn Morrill and Bob Carpenter 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.

Crouch, Richard and Josef van Genabith. 1997. How to glue a donkey to an f-structure, or porting a dynamic meaning representation into LFG's linear logic based glue-language semantics. Presented at the Second International Workshop on Computational Semantics, Tilburg, The Netherlands, January 1997.

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, Ronald M. Kaplan, John T. Maxwell, III, and Annie Zaenen, editors. 1995. *Formal Issues in Lexical-Functional Grammar.* Center for the Study of Language and Information, Stanford University.

Dalrymple, Mary, John Lamping, Fernando C. N. Pereira, and Vijay Saraswat. 1995a. 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, Fernando C. N. Pereira, and Vijay Saraswat. 1996a. Intensional verbs without type-raising or lexical ambiguity. In Jerry Seligman and Dag Westerståhl, editors, *Logic, Language, and Computation,volume 1.* Center for the Study of Language and Information, Stanford University. Also in *Proceedings of the Conference on Information-Oriented Approaches to Logic, Language and Computation/Fourth Conference on Situation Theory and its Applications*, Saint Mary's College of California, Moraga, California. June 1994.

Dalrymple, Mary, John Lamping, Fernando C. N. Pereira, and Vijay Saraswat. 1996b. Quantifiers, anaphora, and intensionality. *Journal of Logic, Language, and Information.* To appear.

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*, 50:1–102.

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 (FGCS-88)*, pages 1116–1122, Tokyo, Japan, November. Reprinted in Dalrymple, Kaplan, Maxwell, and Zaenen, eds., *Formal Issues in Lexical-Functional Grammar*, 279–292. Stanford: Center for the Study of Language and Information. 1995.

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

Johnson, Mark. 1996. Resource-sensitivity in Lexical-Functional Grammar. In *Proceedings of the 1996 Roma Workshop.*

Kaplan, Ronald M. 1987. Three seductions of computational psycholinguistics. In Peter Whitelock, Mary McGee Wood, Harold L. Somers, Rod Johnson, and Paul Bennett, editors, *Linguistic Theory and Computer Applications.* Academic Press, London, pages 149–188. Also: CCL/UMIST Report No. 86.2: Alvey/ICL Workshop on Linguistic Theory and Computer Applications: Transcripts of Presentations and Discussions. Center for Computational Linguistics, University of Manchester, Institute of Science and Technology, Manchester. Reprinted in Dalrymple, Kaplan,Maxwell, and Zaenen, eds., *Formal Issues in Lexical-Functional Grammar*, 337–367. Stanford: Center for the Study of Language and Information. 1995.

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. Reprinted in Dalrymple, Kaplan, Maxwell, and Zaenen, eds., *Formal Issues in Lexical-Functional Grammar*, 29–130. Stanford: Center for the Study of Language and Information. 1995.

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. Reprinted in Dalrymple, Kaplan, Maxwell, and Zaenen, eds., *Formal Issues in Lexical-Functional Grammar*, 311–329. Stanford: Center for the Study of Language and Information. 1995.

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.

Kehler, Andrew, Mary Dalrymple, 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.

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.
Muskens, Reinhard. 1994. A compositional discourse representation theory. In P. Dekker and M. Stokhof, editors, *Proceedings of the 9th Amsterdam Colloquium*, pages 78–96, Amsterdam, Holland.
Neale, Stephen. 1990. *Descriptions*. The MIT Press, Cambridge, MA.
Oehrle, Richard T. 1995. LFG as labeled deduction. Presented at Mathematics of Language 4, University of Pennsylvania, October 1995.
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. 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.
Troelstra, A. S. 1993. A tutorial on linear logic. In Peter Schroeder-Heister and Kosta Došen, editors, *Studies in Logic and Computation 2*. Oxford University Press, Oxford, pages 327–355.

This article was processed using the LaTeX macro package with LLNCS style