LFG Semantics via Constraints

Mary Dalrymple **John Lamping** **Vijay Saraswat**
{dalrymple, lamping, saraswat}@parc.xerox.com
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304 USA

Abstract

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. Traditionally, meanings are combined via function composition, which works well when constituent structure trees are used to guide semantic composition. More recently, the *functional structure* of LFG has been 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 composition. 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 also allows for a coherent treatment of modification as well as of the LFG requirements of completeness and coherence.

1 Introduction

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 kissed Hillary* has the surface constituent structure indicated by the bracketing in 1:

(1) $[_{\mathrm{S}}\ [_{\mathrm{NP}}$ Bill$]\ [_{\mathrm{VP}}$ kissed $[_{\mathrm{NP}}$ 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 that of 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, 1983; 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, cross-linguistically uniform 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. However, as noted by Halvorsen [1983] and Reyle [1988], 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 not assumed to be ordered, 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.

Related approaches. In an effort to solve the problem of the order-dependence imposed by standard versions of the λ-calculus, Reyle [1988] proposes to extend the λ-calculus to reduce its sequential bias, assembling meanings by an enhanced application mechanism. However, it is not clear how Reyle's system can be extended to treat modification or complex predicates, phenomena which our use of linear logic allows us to handle.

Another means of overcoming the problem of the order-dependence of the λ-calculus is to adopt semantic terms whose structure resembles f-structures [Fenstad *et al.*, 1985; Pollard and Sag, 1987; Halvorsen and Kaplan, 1988]. On these approaches, attribute-value matrices are used to encode semantic information, allowing the syntactic and semantic representations to be built up simultaneously and in the same order-independent manner. However, when expressions of the λ-calculus are replaced with attribute-value matrices, other problems arise: in particular, it is not clear how to view such attribute-value matrices as *formulas*, since issues such as the representation of variable binding and scope are not treated precisely.

These problems have been noted, and remedies have been proposed. Sometimes, for example, an algorithm is given which globally examines a semantic attribute-value matrix representation to construct a sentence in a well-defined logic; for instance, Halvorsen [1983] presents an approach in which attribute-value matrices are translated into formulas of intensional logic. However, the computation involved is concerned with manipulating these *representations* in procedural ways: it is hard to see how these procedural mechanisms translate to *meaning preserving* manipulations on the formulas that the matrices represent. In sum, such approaches tend to sacrifice the semantic precision and declarative simplicity of logical approaches (e.g. λ-calculus based approaches), and seem difficult to extend generally or motivate convincingly.

Our approach. Our approach shares the order-independent features of approaches that represent semantic information using attribute-value matrices, while still allowing a well-defined treatment of variable binding and scope. We do this by identifying (1) a *language of meanings* and (2) a *language for assembling meanings*.

In principle, (1) can be any logic (e.g., Montague's higher-order logic); for the purposes of this paper all we need is the language of first-order terms. Because we assemble the meaning out of semantically precise components, our approach shares the precision of the λ-calculus based approaches. For example, the assembled meaning has precise variable binding and scoping.

We take (2) to be a fragment of first-order (linear) logic carefully chosen for its computational properties, as discussed below. In contrast to using the λ-calculus to combine fragments of meaning via ordered applications, we combine fragments of meaning through unordered conjunction, and implication. Rather than using λ-reduction to simplify meanings, we rely on deduction, as advocated by Pereira [1990; 1991].

The elements of the f-structure provide an unordered set of constraints, expressed in the logic, governing how the semantics can fit together. Constraints for combining lexically-provided meanings can be encoded in lexical items, as instructions for combining several arguments into a result.[1]

In effect, then, our approach uses first order 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.

To better capture some linguistic properties, we make use of first order *linear logic* as the glue with which meanings are assembled [Girard, 1987].[2] One

[1] Constraints may also be provided as rules governing particular configurations. Such rules are applicable when properties not of individual lexical items in the construction but of the construction as a whole are responsible for its interpretation; these cases include the semantics of relative clauses. We will not discuss examples of configurationally-defined rules in this paper.

[2] Specifically, we make use only of the *tensor fragment* of linear logic. The fragment is closed under conjunction, universal quantification and implication (with atomic an-

way of thinking about linear logic is that it introduces accounting of premises and conclusions, so that deductions consume their premises to generate their conclusions. It turns out that this property of linear logic nicely captures the LFG requirements of *coherence* and *consistency*, and additionally provides a natural way to handle modifiers: a modifier consumes the unmodified meaning of the structure it modifies and produces from it a new, modified meaning.

In the following, we first illustrate our approach by discussing a simple example, and then present more complex examples showing how modifiers and valence changing operations are handled.

2 Theoretical preliminaries

In the following, we describe two linguistic assumptions that underlie this work. First, we assume that various aspects of linguistic structure (phonological, syntactic, semantic, and other aspects) are formally represented as *projections* and are related to one another by means of functional correspondences. We also assume that the relation between the thematic roles of a verb and the grammatical functions that realize them are specified by means of *mapping principles* which apply postlexically.

Projections. We adopt the *projection architecture* proposed by Kaplan [1987] and Halvorsen and Kaplan [1988] to relate f-structures to representations of their meaning: f-structures are put in functional correspondence with semantic representations, similar to the correspondence between nodes of the constituent structure tree and f-structures. The *semantic projection* of an f-structure, written with a subscript σ, is a representation of the meaning of that f-structure.

Thus, the notation '$\uparrow_\sigma$' in the lexical entries given in Figure 1 stands for the semantic projection of the f-structure '$\uparrow$'; similarly, '$(\uparrow \textsc{subj})_\sigma$' is the semantic projection of $(\uparrow \textsc{subj})$. The equation $\uparrow_\sigma = \mathit{Bill}$ indicates that the semantic projection of $\uparrow$, the f-structure introduced by the NP *Bill*, is *Bill*. The lexical entry for *Hillary* is analogous. When a lexical entry is used, the metavariable '$\uparrow$' is instantiated and replaced with an actual variable corresponding to an f-structure f_n [Kaplan and Bresnan, 1982, page 183]. Similarly, the metavariable '$\uparrow_\sigma$' is instantiated to a logic variable corresponding to the meaning of the f-structure. In other words, the equation $\uparrow_\sigma = \mathit{Bill}$ is instantiated as $f_{n\sigma} = \mathit{Bill}$ for some logic variable $f_{n\sigma}$.

tecedents). It arises from transferring to linear logic the ideas underlying the concurrent constraint programming scheme of Saraswat [1989] — an explicit formulation for the higher-order version of the linear concurrent constraint programming scheme is given in Saraswat and Lincoln [1992]. A nice tutorial introduction to linear logic itself may be found in Scedrov [1990].

We have used the multiplicative conjunction $\otimes$ and linear implication $\multimap$ connectives of linear logic, rather than the analogous conjunction $\wedge$ and implication $\rightarrow$ of classical logic. For the present, we can think of the linear and classical connectives as being identical. Similarly, the *of course* connective '!' of linear logic can be ignored for now. Below, we will discuss respects in which the linear logic connectives have properties that are crucially different from their counterparts in classical logics.

Mapping principles. We follow Bresnan and Kanerva [1989], Alsina [1993], Butt [1993] and others in assuming that verbs specify an association between each of their arguments and a particular thematic role, and that *mapping principles* associate these thematic roles with surface grammatical functions; this assumption, while not necessary for the treatment of simple examples such as the one discussed in Section 3, is linguistically well-motivated and enables us to provide a nice treatment of *complex predicates*, to be discussed in Section 5.

The lexical entry for *kiss* specifies the denotation of $(\uparrow \textsc{pred})$: it requires two arguments which we will label *agent* and *theme*. Mapping principles ensure that each of these arguments is associated with some grammatical function: here, the SUBJ of *kiss* (*Bill*) is interpreted as the agent, and the OBJ of *kiss* (*Hillary*) is interpreted as the theme. The specific mapping principles that we assume are given in Figure 2.

The function of the mapping principles is to specify the set of possible associations between grammatical functions and thematic roles. This is done by means of implication. Grammatical functions always appear on the left side of a mapping principle implication, and the thematic roles with which those grammatical functions are associated appear on the right side. Mapping principle (1), for example, relates the thematic roles of agent and theme designated by a two-argument verb like *kiss* to the grammatical functions that realize these arguments: it states that if a SUBJ and an OBJ are present, this permits the deduction that the thematic role of agent is associated with the SUBJ and the thematic role of theme is associated with the OBJ. (Other associations are encoded by means of other mapping principles; the mapping principles given in Figure 2 encodes only two of the possibilities.)

We make implicit appeal to an independently-given, fully-worked-out theory of argument mapping, from which mapping principles such as those given in Figure 2 can be shown to follow. It is important to note that we do not intend any claims about the correctness of the specific details of the mapping principles given in Figure 2; rather, our claim is that mapping principles should be of the general form illustrated there, specifying possible relations between thematic roles and grammatical functions. In particular, no theoretical significance should be

Bill	NP	$(\uparrow \text{PRED}) = \text{'BILL'}$ $\uparrow_\sigma = \mathit{Bill}$
kissed	V	$(\uparrow \text{PRED}) = \text{'KISS'}$ $\forall X, Y.\ \mathit{agent}((\uparrow \text{PRED})_\sigma, X) \otimes \mathit{theme}((\uparrow \text{PRED})_\sigma, Y) \multimap\ \uparrow_\sigma = \mathit{kiss}(X, Y)$
Hillary	NP	$(\uparrow \text{PRED}) = \text{'HILLARY'}$ $\uparrow_\sigma = \mathit{Hillary}$

Figure 1: Lexical entries for *Bill*, *kissed*, *Hillary*

(1) $!(\forall f, X, Y.\ ((f\ \text{SUBJ})_\sigma = X) \otimes ((f\ \text{OBJ})_\sigma = Y) \multimap \mathit{agent}((f\ \text{PRED})_\sigma, X) \otimes \mathit{theme}((f\ \text{PRED})_\sigma, Y))$

(2) $!(\forall f, X, Y, Z.\ ((f\ \text{SUBJ})_\sigma = X) \otimes ((f\ \text{OBJ})_\sigma = Y) \otimes ((f\ \text{OBJ2})_\sigma = Z) \multimap$
$\mathit{permitter}((f\ \text{PRED})_\sigma, X) \otimes \mathit{agent}((f\ \text{PRED})_\sigma, Z) \otimes \mathit{theme}((f\ \text{PRED})_\sigma, Y))$

Figure 2: Argument mapping principles

bill: $(f_{2\sigma} = \mathit{Bill})$
hillary : $(f_{3\sigma} = \mathit{Hillary})$
kissed : $(\forall X, Y.\ \mathit{agent}(f_{1\sigma}, X) \otimes \mathit{theme}(f_{1\sigma}, Y) \multimap f_{4\sigma} = \mathit{kiss}(X, Y))$
mapping1 : $(\forall X, Y.\ (f_{2\sigma} = X) \otimes (f_{3\sigma} = Y) \multimap \mathit{agent}(f_{1\sigma}, X) \otimes \mathit{theme}(f_{1\sigma}, Y)))$

$(\mathbf{bill} \otimes \mathbf{hillary} \otimes \mathbf{kissed} \otimes \mathbf{mapping1})$ (Premises.)

$\multimap \mathit{agent}(f_{1\sigma}, \mathit{Bill}) \otimes \mathit{theme}(f_{1\sigma}, \mathit{Hillary}) \otimes \mathbf{kissed}$ (UI, Modus Ponens.)

$\multimap f_{4\sigma} = \mathit{kiss}(\mathit{Bill}, \mathit{Hillary})$ (UI, Modus Ponens.)

Figure 3: Derivation of *Bill kissed Hillary*

attached to the choice of thematic role labels used here; for the verb *kiss*, for example, labels such as 'kisser' and 'kissed' would do as well. We require only that the thematic roles designated in the lexical entries of individual verbs are specified in enough detail for mapping principles such as those illustrated in Figure 2 to apply successfully.

3 A simple example of semantic composition

Consider sentence 2 and the lexical entries given in Figure 1:

(2) Bill kissed Hillary.

The f-structure for (2) is:

(3)
$$f_4: \begin{bmatrix} \text{PRED} & f_1\text{:'KISS'} \\ \text{SUBJ} & f_2\text{:}\begin{bmatrix} \text{PRED} & \text{'BILL'} \end{bmatrix} \\ \text{OBJ} & f_3\text{:}\begin{bmatrix} \text{PRED} & \text{'HILLARY'} \end{bmatrix} \end{bmatrix}$$

The meaning associated with the f-structure may be derived by logical deduction, as shown in Figure 3.[3]

The first three lines contain the information contributed by the lexical entries for *Bill*, *Hillary*, and *kissed*, abbreviated as **bill**, **hillary**, and **kissed**. The verb *kissed* requires two pieces of information, an agent and a theme, in no particular order, to produce a meaning for the sentence, $f_{4\sigma}$. The mapping principle needed for associating the syntactic arguments of transitive verbs with the agent/theme argument structure is given on the fourth line and abbreviated as **mapping1**. Mapping principles are assumed to be a part of the background theory, rather than being introduced by particular lexical items. Each mapping principle can, then, be used as many or as few times as necessary.

The premises—i.e., the lexical entries and mapping principle—are restated as the first step of the derivation, labeled 'Premises'. The second step is derived from the premises by Universal Instantiation and Modus Ponens. The last step is then derived from this result by Universal Instantiation and Modus Ponens.

To summarize: a variable is introduced for the meaning corresponding to each f-structure in the

[3] An alternative derivation, not using mapping principles, is also possible. In that case, the lexical entry for *kissed* would require a SUBJ and an OBJ rather than an agent and a theme, and the derivation would proceed in this way:
$((f_{2\sigma} = \mathit{Bill})$
$\otimes (f_{3\sigma} = \mathit{Hillary})$
$\otimes (\forall X, Y. f_{2\sigma} = X \otimes f_{3\sigma} = Y \multimap f_{4\sigma} = \mathit{kiss}(X, Y)))$
$\multimap f_{4\sigma} = \mathit{kiss}(\mathit{Bill}, \mathit{Hillary})$

syntactic representation. These variables form the scaffolding that guides the assembly of the meaning. Further information is then introduced: information associated with each lexical entry is made available, as are all the mapping rules. Once all this information is present, we look for a logical deduction of a meaning of the sentence from that information.

The use of linear logic provides certain advantages, since it allows us to capture the intuition that lexical items and phrases contribute uniquely 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.

Similar observations underlie the work of Lambek [1958] on categorial grammars and the recent work of van Benthem [1991] and others on dynamic logics.

It is this 'resource-conscious' property of natural language semantics – a meaning is used once and once only in a semantic derivation – that linear logic allows us to capture. The basic insight underlying linear logic is to treat logical formulas as finite *resources*, which are consumed in the process of deduction. This gives rise to a notion of *linear implication* $\multimap$ which is resource-conscious: 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 implies B — but not $A \otimes B$ (because the deduction consumes A), and not $(A \multimap B) \otimes B$ (because the linear implication is also consumed in doing the deduction). The resource consciousness not only disallows arbitrary duplication of formulas, but also arbitrary deletion of formulas. This causes the notion of conjunction we use ($\otimes$) to be 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 $A \otimes A \otimes (A \multimap B)$ does linearly imply $A \otimes B$ (there is still one A left over) — but does not linearly imply B (there must still be one A present). Thus, linear logic checks that a formula is used once and only once in a deduction, reflecting the resource-consciousness of natural language semantics. Finally, linear logic has an *of course* connective ! which turns off accounting for its formula. That is, $!A$ linearly implies an arbitrary number copies of A, including none. We use this connective on the background theory of mapping principles to indicate that they are not subject to accounting; they can be used as often or seldom as necessary.

A primary advantage of the use of linear logic is that it enables a clean semantic definition of *completeness* and *coherence*.[4] 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 $Th \vdash \phi \multimap f_\sigma = t$ (for some term t), where Th is the background theory containing, e.g., the mapping principles. Each 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.

4 Modification

Another 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. Consider the following sentence, containing the sentential modifier *obviously*:

(4) Bill obviously kissed Hillary.

We make the standard assumption that the verb *kissed* is the main syntactic predicate of this sentence. The following is the f-structure for example 4:

(5)
$$f_4\!:\!\left[\begin{array}{ll} \textsc{pred} & f_1\!:\text{`}\textsc{kiss}\text{'} \\ \textsc{subj} & f_2\!:\!\left[\begin{array}{ll}\textsc{pred} & \text{`}\textsc{Bill}\text{'}\end{array}\right] \\ \textsc{obj} & f_3\!:\!\left[\begin{array}{ll}\textsc{pred} & \text{`}\textsc{Hillary}\text{'}\end{array}\right] \\ \textsc{mods} & \left\{ f_5\!:\!\left[\begin{array}{ll}\textsc{pred} & \text{`}\textsc{obviously}\text{'}\end{array}\right] \right\} \end{array}\right]$$

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

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

It is clear that there is a 'mismatch' of sorts 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.[5]

Consider now the lexical entry for *obviously* given in Figure 4. The semantic equation associated with

[4] '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]

[5] 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	NP	$(\uparrow \text{PRED}) = \text{'BILL'}$ $\uparrow_\sigma = \mathit{Bill}$
obviously	ADV	$(\uparrow \text{PRED}) = \text{'OBVIOUSLY'}$ $\forall P.\ (\text{MODS}\ \uparrow)_\sigma = P \multimap (\text{MODS}\ \uparrow)_\sigma = \mathit{obviously}(P)$
kissed	V	$(\uparrow \text{PRED}) = \text{'KISS'}$ $\forall X, Y.\ \mathit{agent}((\uparrow \text{PRED})_\sigma, X) \otimes \mathit{theme}((\uparrow \text{PRED})_\sigma, Y) \multimap \uparrow_\sigma = \mathit{kiss}(X, Y)$
Hillary	NP	$(\uparrow \text{PRED}) = \text{'HILLARY'}$ $\uparrow_\sigma = \mathit{Hillary}$

Figure 4: Lexical entries for *Bill, obviously, kissed, Hillary*

$$
\begin{array}{ll}
\mathbf{bill} : (f_{2\sigma} = \mathit{Bill}) & \\
\mathbf{hillary} : (f_{3\sigma} = \mathit{Hillary}) & \\
\mathbf{kissed} : (\forall X, Y.\ \mathit{agent}(f_{1\sigma}, X) \otimes \mathit{theme}(f_{1\sigma}, Y) \multimap f_{4\sigma} = \mathit{kiss}(X, Y)) & \\
\mathbf{obviously} : (\forall P.\ f_{4\sigma} = P \multimap f_{4\sigma} = \mathit{obviously}(P)) & \\
\mathbf{mapping1} : (\forall X, Y.\ (f_{2\sigma} = X) \otimes (f_{3\sigma} = Y) \multimap \mathit{agent}(f_{1\sigma}, X) \otimes \mathit{theme}(f_{1\sigma}, Y))) & \\
(\mathbf{bill} \otimes \mathbf{hillary} \otimes \mathbf{kissed} \otimes \mathbf{obviously} \otimes \mathbf{mapping1}) & \text{(Premises.)} \\
\multimap \mathit{agent}(f_{1\sigma}, \mathit{Bill}) \otimes \mathit{theme}(f_{1\sigma}, \mathit{Hillary}) \otimes \mathbf{kissed} \otimes \mathbf{obviously} & \text{(UI, Modus Ponens.)} \\
\multimap f_{4\sigma} = \mathit{kiss}(\mathit{Bill}, \mathit{Hillary}) \otimes \mathbf{obviously} & \text{(UI, Modus Ponens.)} \\
\multimap f_{4\sigma} = \mathit{obviously}(\mathit{kiss}(\mathit{Bill}, \mathit{Hillary})) & \text{(UI, Modus Ponens.)}
\end{array}
$$

Figure 5: Derivation of *Bill obviously kissed Hillary*

obviously makes use of 'inside-out functional uncertainty' [Halvorsen and Kaplan, 1988]. The expression (MODS $\uparrow$) denotes an f-structure through which there is a path MODS leading to $\uparrow$. For example, if $\uparrow$ is the f-structure labeled f_5 above, then (MODS $\uparrow$) is the f-structure labeled f_4, and $(\text{MODS}\ \uparrow)_\sigma$ is the semantic projection of f_4. Thus, the lexical entry for *obviously* specifies the semantic representation of the f-structure that it modifies, an f-structure in which it is properly contained.

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 in particular that the meaning of the modified structure, $(\text{MODS}\ \uparrow)_\sigma$, appears on *both* sides of $\multimap$; the unmodified meaning is consumed, and the modified meaning is produced.

The derivation of the meaning of example 4 is shown in Figure 5. The first part of the derivation is the same as the derivation shown in Figure 3 for the sentence *Bill kissed Hillary*. The crucial difference is the presence of information introduced by *obviously*, shown in the fourth line and abbreviated as **obviously**. In the last step in the derivation, the linear implication introduced by *obviously* consumes the previous value for $f_{4\sigma}$ 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 $f_{4\sigma}$ 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 Valence-changing operations

We have seen that modifiers can be treated as 'consuming' the meaning of the structure that they modify, producing a new, modified meaning. A similar, although syntactically more complex, case arises with *complex predicates*, as Butt [1990; 1993] shows.

Butt discusses the 'permissive construction' in Urdu, illustrated in 7:

(7) Hillary-ne diyaa [$_{\text{VP}}$ Bill-ko xat likhne]
Hillary-ERG let Bill-DAT letter-NOM write-PART
'Hillary let Bill write a letter.'

She shows that although the permissive construction is seemingly biclausal, it actually involves a *complex predicate*: a syntactically monoclausal predicate formed in the presence of the verb *diyaa* 'let'. In the case at hand, the presence of *diyaa* requires an

Hillary	NP	$(\uparrow \text{PRED}) = \text{`HILLARY'}$ $\uparrow_\sigma = Hillary$
Bill	NP	$(\uparrow \text{PRED}) = \text{`BILL'}$ $\uparrow_\sigma = Bill$
xat	N	$(\uparrow \text{PRED}) = \text{`LETTER'}$ $\uparrow_\sigma = letter$
likhne	V	$(\uparrow \text{PRED}) = \text{`WRITE'}$ $\forall X, Y.\ agent((\uparrow \text{PRED})_\sigma, X) \otimes theme((\uparrow \text{PRED})_\sigma, Y) \multimap\ \uparrow_\sigma = write(X, Y)$
diyaa	V	$\forall X, P.\ permitter((\uparrow \text{PRED})_\sigma, X) \otimes \uparrow_\sigma = P \multimap\ \uparrow_\sigma = let(X, P)$

Figure 6: Lexical entries for *Hillary*, *Bill*, *xat*, *likhne*, *diyaa*

$$
\begin{array}{ll}
\textbf{hillary} : (f_{2\sigma} = Hillary) & \\
\textbf{bill} : (f_{3\sigma} = Bill) & \\
\textbf{letter} : (f_{4\sigma} = letter) & \\
\textbf{write} : (\forall X, Y.\ agent(f_{1\sigma}, X) \otimes theme(f_{1\sigma}, Y) \multimap f_{5\sigma} = write(X, Y)) & \\
\textbf{let} : (\forall X, P.\ permitter(f_{5\sigma}, X) \otimes f_{5\sigma} = P \multimap\ = f_{5\sigma} = let(X, P) & \\
\textbf{mapping2} : (\forall X, Y, Z.\ (f_{2\sigma} = X) \otimes (f_{3\sigma} = Y) \otimes (f_{4\sigma} = Z) \multimap & \\
\quad permitter(f_{1\sigma}, X) \otimes agent(f_{1\sigma}, Y) \otimes theme(f_{1\sigma}, Z)) & \\
(\textbf{bill} \otimes \textbf{hillary} \otimes \textbf{letter} \otimes \textbf{write} \otimes \textbf{let} \otimes \textbf{mapping2}) & \text{(Premises.)} \\
\multimap permitter(f_{1\sigma}, Hillary) \otimes agent(f_{1\sigma}, Bill) \otimes theme(f_{1\sigma}, letter) \otimes \textbf{write} \otimes \textbf{let} & \text{(UI, Modus Ponens.)} \\
\multimap permitter(f_{1\sigma}, Hillary) \otimes \textbf{let} \otimes (f_{5\sigma} = write(Bill, letter)) & \text{(UI, Modus Ponens.)} \\
\multimap f_{5\sigma} = let(Hillary, (write(Bill, letter)) & \text{(UI, Modus Ponens.)}
\end{array}
$$

Figure 7: Derivation of *Hillary let Bill write a letter*

additional argument which we will label 'permitter', in addition to the arguments required by the verb *likhne* 'write'. In general, the verb *diyaa* 'let' modifies the argument structure of the verb with which it combines, requiring in addition to the original inventory of arguments the presence of a permitter. The f-structure for example 7 is:

(8)
$$
f_5 : \begin{bmatrix}
\text{PRED} & f_1 : \text{`LET}\langle\text{WRITE}\rangle\text{'} \\
\text{SUBJ} & f_2 : \begin{bmatrix} \text{PRED} & \text{`HILLARY'} \end{bmatrix} \\
\text{OBJ2} & f_3 : \begin{bmatrix} \text{PRED} & \text{`BILL'} \end{bmatrix} \\
\text{OBJ} & f_4 : \begin{bmatrix} \text{PRED} & \text{`LETTER'} \end{bmatrix}
\end{bmatrix}
$$

As Butt points out, the verbs participating in the formation of the permissive construction need not form a syntactic constituent; in example 7, the verbs *likhne* and *diyaa* are not even next to each other. This shows that complex predicate formation cannot be analyzed as taking place in the lexicon; a method of dynamically creating a complex predicate in the syntax is needed. That is, sentences such as 7 have, in essence, two syntactic heads, which dynamically combine to produce a single syntactic argument structure.

We claim that the function of a verb such as permissive *diyaa* is somewhat analogous to that of a modifier: *diyaa* consumes the meaning of the original verb and its arguments, producing a new permissive meaning and requiring an additional argument, the permitter. Mapping principles apply to this new, augmented argument structure to associate the new thematic argument structure with the appropriate set of syntactic roles. We illustrate the derivation of the meaning of example 7 in Figure 7.

The lexical entries necessary for example 7 can be found in Figure 6. The instantiated information from these lexical entries appears in the first five lines of Figure 7. Mapping principle (2) in Figure 2, abbreviated as **mapping2**, links the permitter, agent, and theme of the (derived) argument structure to the syntactic arguments of a permissive construction; the mapping principle is given in the sixth line of Figure 7.[6]

The premises of the derivation are, as above, information given by lexical entries and the mapping principle. By means of mapping principle **mapping2**, information about the possible array of thematic roles required by the complex predicate *let-write* can be

[6] Recall that in our framework, all the mapping principles are present to be used as needed. In the derivation of the meaning of example 7, shown in Figure 7, we have omnisciently provided the one that will be needed.

derived; this step uses Universal Instantiation and Modus Ponens.

Next, a (preliminary) meaning for f-structure f_5, $write(Bill, letter)$, is derived by Universal Instantiation and Modus Ponens. At this point, the requirements imposed by *diyaa* 'let', labeled **let**, are met: a permitter ($Hillary$) is present, and a complete meaning for f-structure f_5 has been produced. These meanings can be consumed, and a new meaning produced, as represented in the final line of the derivation. Again, this meaning is the only one available, since completeness and coherence obtains only when all requirements are fulfilled and no extra information remains. As with the case of modifiers, the final step provides the only complete and coherent meaning derivable for the utterance.

Notice that the meaning of the complex predicate is not derived by composition of verb meanings: the permissive verb *diyaa* does not combine with the verb *likhne* 'write' to form a new verb meaning. Instead, permissive *diyaa* requires a (preliminary) sentence meaning, $write(Bill, letter)$ in the example above, in addition to the presence of a permitter argument.

More generally, this approach treats linguistic phenomena such as modification and complex predicate formation function by operating on semantic entities that have combined with all of their arguments, producing a modified meaning and (in the case of complex predicate formation) introducing further arguments. While it would be possible to extend our approach to operate on semantic entities that have not combined with all their arguments, we have not yet encountered a compelling reason to do so. Our current restriction is not so confining as it might appear; most operations that can be performed on semantic entities that have not combined with all their arguments have analogues that operate on fully combined entities. In further research, we plan to explore this characteristic of our analysis more fully.

6 Conclusion

Our approach results in a somewhat different view of semantic composition, compared to λ-calculus based approaches. First of all, notice that both in λ-calculus based approaches and in our approach, there is not only a *semantic level* of meanings of utterances and phrases, but also a *glue level* or *composition level* responsible for assembling semantic level meanings of constituents to get a meaning for an entire utterance.

In λ-calculus based approaches, the semantic level is higher order intensional logic. The composition level is the rules, often not stated in any formal system, that say what pattern of applications to do to assemble the constituent meanings. The composition level relies on function application in the semantic level to assemble meanings. This forces some conflation of the levels, because it is using a semantic level operation, application, to carry out a composition level task. It requires functions at the semantic level whose primary purpose is to allow the composition level to combine meanings via application. For example, in order for the composition level to work right, the semantic level meaning of a transitive verb must be a function of two arguments, rather than a relation. This rather artificial requirement is a symptom of some of the work of the composition level being done at the semantic level.

Our approach, on the other hand, better segregates the two levels of meaning, because the composition level uses its own mechanism (substitution) to assemble semantic level meanings, rather than relying on semantic level operations. Thus, the linear logic operations of the composition level don't appear at the semantic level and the classical operations of the semantic level don't appear at the composition level.[7]

Our system also expresses the composition level rules in a formal system, first order linear logic. The composition rules are expressed by relations in the lexical entries and the mapping rules. There is no separate process of deciding how the meanings of lexical entries will be combined; the relations they establish, together with some background facts, just imply the high level meaning. All the necessary connections between phrases are made at the composition level when lexical entries are instantiated, through the shared variables of the sigma projections. From then on, logical inference at the composition level assembles the semantic level meaning.

These examples illustrate the capability of our framework to handle the combination of predicates with their arguments, modification, and arity-affecting operations. The use of linear logic provides a simple treatment of the requirements of completeness and consistency and of complex predicates. Further, our deduction framework enables us to use linear logic to state such operations in a formally well-defined and tractable manner.

In future work, we plan to explore more fully the semantics of modification, and to pursue the addition of a type system to the logic to treat quantifiers analogously to Pereira [1990; 1991].

7 Acknowledgments

We are grateful to Ron Kaplan, Stanley Peters, John Maxwell, Joan Bresnan, and Stuart Shieber for helpful comments on earlier versions of this paper. We would particularly like to thank Fernando Pereira for extensive and very helpful discussion of the issues presented here.

[7] This separation is not a necessary consequence of using deduction to assemble meanings; the composition logic could call for semantic level operations. But we have so far been able to maintain the separation, and the question of whether the separation can be maintained seems to be linguistically interesting and worthy of further pursuit.

References

[Alsina, 1993] Alex Alsina. *Predicate Composition: A Theory of Syntactic Function Alternations.* PhD thesis, Stanford University, 1993.

[Bresnan and Kanerva, 1989] Joan Bresnan and Jonni M. Kanerva. Locative inversion in Chicheŵa: A case study of factorization in grammar. *Linguistic Inquiry*, 20(1):1–50, 1989. Also in E. Wehrli and T. Stowell, eds., Syntax and Semantics 26: Syntax and the Lexicon. New York: Academic Press.

[Butt *et al.*, 1990] Miriam Butt, Michio Isoda, and Peter Sells. Complex predicates in LFG. MS, Stanford University, 1990.

[Butt, 1993] Miriam Butt. *The Structure of Complex Predicates.* PhD thesis, Stanford University, 1993. In preparation.

[Fenstad *et al.*, 1985] Jens Erik Fenstad, Per-Kristian Halvorsen, Tore Langholm, and Johan van Benthem. Equations, schemata and situations: A framework for linguistic semantics. Technical Report 29, Center for the Study of Language and Information, Stanford University, 1985.

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

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

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

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

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

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

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

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

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

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

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

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

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

[Pollard and Sag, 1987] Carl Pollard and Ivan A. Sag. *Information-Based Syntax and Semantics, Volume I.* Number 13 in CSLI Lecture Notes. CSLI/The University of Chicago Press, Stanford University, 1987.

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

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

[Saraswat, 1989] Vijay A. Saraswat. *Concurrent Constraint Programming Languages.* PhD thesis, Carnegie-Mellon University, 1989. To appear, Doctoral Dissertation Award and Logic Programming Series, MIT Press, 1993.

[Scedrov, 1990] A. Scedrov. A brief guide to linear logic. *Bulletin of the European Assoc. for Theoretical Computer Science*, 41:154–165, June 1990.

[Simpson, 1983] Jane Simpson. *Aspects of Warlpiri Morphology and Syntax.* PhD thesis, MIT, 1983.

[Simpson, 1991] Jane Simpson. *Warlpiri Morpho-Syntax.* Kluwer Academic Publishers, Dordrecht, 1991.

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