A Deductive Account of Quantification in LFG*

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

January 14, 2015

The relationship between Lexical-Functional Grammar (LFG) *functional structures* (f-structures) for sentences and their semantic interpretations can be expressed directly in a fragment of linear logic in a way that explains correctly the constrained interactions between quantifier scope ambiguity and bound anaphora.

The use of a deductive framework to account for the compositional properties of quantifying expressions in natural language obviates the need for additional mechanisms, such as Cooper storage, to represent the different scopes that a quantifier might take. Instead, the semantic contribution of a quantifier is recorded as an ordinary logical formula, one whose use in a proof will establish the scope of the quantifier. The properties of linear logic ensure that each quantifier is scoped exactly once.

Our analysis of quantifier scope can be seen as a recasting of Pereira's analysis (Pereira, 1991), which was expressed in higher-order intuitionistic logic. But our use of LFG and linear logic provides a much more direct and computationally more flexible interpretation mechanism for at least the same range of phenomena. We have developed a preliminary Prolog implementation of the linear deductions described in this work.

*Xerox Palo Alto Research Center Technical Report ISTL-NLTT-1993-06-01. To appear in *Quantifiers, Deduction, and Context*, ed. Makoto Kanazawa, Christopher J. Piñón, and Henriette de Swart. Stanford, California: Center for the Study of Language and Information, 1994.

†Xerox PARC, Palo Alto, California.

‡AT&T Bell Laboratories, Murray Hill, New Jersey.

1 Introduction

This paper describes part of our ongoing investigation on the use of formal deduction in linear logic to explicate the relationship between syntactic analyses in Lexical-Functional Grammar (LFG) and semantic interpretations. The use of formal deduction in semantic interpretation was implicit in deductive systems for categorial syntax (Lambek, 1958), and has been made explicit through applications of the Curry-Howard parallelism between proofs and terms in more recent work on categorial semantics (van Benthem, 1988; van Benthem, 1991), labeled deductive systems (Moortgat, 1992b) and flexible categorial systems (Hendriks, 1993). Accounts of the syntax-semantics interface in the categorial tradition require that syntactic and semantic analyses be formalized in parallel algebraic structures of similar signatures, based on generalized application and abstraction (or residuation) operators, and structure-preserving relations between them. Those accounts therefore force the adoption of categorial syntactic analyses, with their strong dependence on phrase structure and linear order.

In contrast, our approach uses linear logic (Girard, 1987) to represent the connection between two dissimilar levels of representation, LFG f-structures and their semantic interpretations. F-structures provide a crosslinguistically uniform representation of syntactic information relevant to semantic interpretation that abstracts away from the details of phrase structure and linear order in particular languages. This generality is in part achieved by using grammatical functions rather than functor-argument relations to represent syntactic predicate-argument relationships. As Halvorsen (1988) notes, however, the flatter, unordered, grammatical function structure of LFG does not fit well with traditional semantic compositionality, based on functional abstraction and application, which mandates a rigid order of semantic composition. We are thus forced to use a more relaxed form of compositionality, in which, as in more traditional ones, the semantics of each lexical entry in a sentence is used exactly once in interpretation, but without imposing a rigid order of composition. It turns out that linear logic offers exactly what is required for a calculus of semantic composition for LFG, in that it can represent directly the constraints on the creation and use of semantic units in sentence interpretation without forcing a particular hierarchical order of composition except as required by the properties of particular lexical entries.

We have shown previously that the linear-logic formalization of the syntax-semantics interface for LFG provides simple and general analyses of modification, functional completeness and coherence, and complex predicate formation (Dalrymple, Lamping, and Saraswat, 1993; Dalrymple, Hinrichs, Lamping, and Saraswat, 1993). In the present

paper, the analysis is extended to the interpretation of quantified noun phrases. After an overview of the approach, we present our analysis of the compositional properties of quantifiers, and we conclude by showing that the analysis correctly accounts for scope ambiguity and its interactions with bound anaphora.

2 LFG and Linear Logic

Syntactic Framework LFG assumes two syntactic levels of representation: constituent structure (*c-structure*) represents phrasal dominance and precedence relations, while functional structure (*f-structure*) represents syntactic predicate-argument structure. For example, the f-structure for sentence (1) is given in (2):

(1) Bill appointed Hillary.

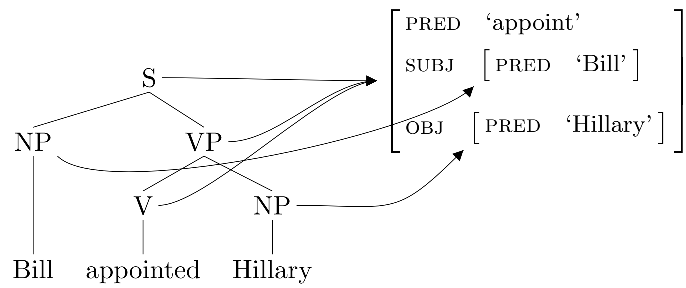
(2) $$\begin{bmatrix} \textsc{pred} & \text{`appoint'} \\ \textsc{subj} & \begin{bmatrix} \textsc{pred} & \text{`Bill'} \end{bmatrix} \\ \textsc{obj} & \begin{bmatrix} \textsc{pred} & \text{`Hillary'} \end{bmatrix} \end{bmatrix}$$

As illustrated, a functional structure consists of a collection of attributes, such as PRED, SUBJ, and OBJ, whose values can, in turn, be other functional structures. The following annotated phrase-structure rules can generate the f-structure in (2):

(3) $$\begin{array}{lccc} \text{S} & \longrightarrow & \text{NP} & \text{VP} \\ & & (\uparrow \textsc{subj}) = \downarrow & \uparrow = \downarrow \\ \text{VP} & \longrightarrow & \text{V} & \text{NP} \\ & & \uparrow = \downarrow & (\uparrow \textsc{obj}) = \downarrow \end{array}$$

These two phrase structure rules do not encode semantic information; they specify only how grammatical functions such as SUBJ are expressed in English. The f-structure metavariables $\uparrow$ and $\downarrow$ refer, respectively, to the f-structure of the mother of the current node and to the f-structure of the current node (Kaplan and Bresnan, 1982). The annotations on the S rule indicate, then, that the f-structure for the S has a SUBJ attribute whose value is the f-structure for the NP daughter, and that the f-structure for the S is the same as the one for the VP daughter. The relation between the nodes of the c-structure and the f-structure for the sentence (1) is expressed by means of arrows in (4):

(4)



Lexically-Specified Semantics Unlike phrase structure rules, lexical entries specify semantic as well as syntactic information. Here are the lexical entries for the words in the sentence:

(5) *Bill* NP $(\uparrow \textsc{pred}) =$ 'Bill'
$\uparrow_\sigma \rightsquigarrow \mathit{Bill}$

appointed V $(\uparrow \textsc{pred})=$ 'appoint'
$\forall X, Y.\ (\uparrow \textsc{subj})_\sigma \rightsquigarrow X \otimes (\uparrow \textsc{obj})_\sigma \rightsquigarrow Y \multimap \uparrow_\sigma \rightsquigarrow \mathit{appoint}(X, Y)$

Hillary NP $(\uparrow \textsc{pred}) =$ 'Hillary'
$\uparrow_\sigma \rightsquigarrow \mathit{Hillary}$

Just like phrase structure rules, lexical entries are instantiated for a particular utterance. The metavariable $\uparrow$ in a lexical entry represents the f-structure of the c-structure mother of (an instance of) the entry in a c-structure. The syntactic information given in lexical entries consists of equality statements about the f-structure, while the semantic information consists of assertions about how the meaning of the f-structure participates in various semantic relations.

The semantic information in a lexical entry, which we will call the *semantic contribution* of the entry, is a linear-logic formula that constrains the association between *semantic structures* projected from the f-structures mentioned in the lexical entry (Kaplan, 1987; Halvorsen and Kaplan, 1988) and their semantic interpretations. The semantic projection function σ maps an f-structure to a semantic structure encoding information about its meaning, in the same way as the functional projection function ϕ maps c-structure nodes to the associated f-structures. The association between f_σ and a meaning P is represented by the atomic formula $f_\sigma \rightsquigarrow P$, 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 of the intended second argument, although for simplicity we will omit the type subscript whenever it is determinable from context.) We will often informally say that P is f's meaning without referring to the role of the semantic structure

f_σ in $f_\sigma \rightsquigarrow P$. We will see, however, that f-structures and their semantic projections must be distinguished, because in general semantic projections carry more information than just the association to the meaning for the corresponding f-structure.

We can now explain the semantic contributions in (5). If a particular occurrence of 'Bill' in a sentence is associated with f-structure f, the syntactic constraint in the lexical entry *Bill* will be instantiated as $(f\ \text{PRED}) = \text{'Bill'}$ and the semantic constraint will be instantiated as $f_\sigma \rightsquigarrow Bill$, representing the association between f_σ and the constant *Bill* representing its meaning.

The semantic contribution of the *appointed* entry is more complex, as it relates the meanings of the subject and object of a clause to the clause's meaning. Specifically, if f is the f-structure for a clause with predicate (PRED) 'appoint', the semantic contribution asserts that if f's subject $(f\ \text{SUBJ})$ has meaning X and (linear conjunction $\otimes$) f's object $(f\ \text{OBJ})$ has meaning Y, then (linear implication $\multimap$) f has meaning $appoint(X, Y)$.[1]

Logical Representation of Semantic Compositionality In the semantic contribution for *appointed* in (5), the linear-logic connectives of multiplicative conjunction $\otimes$ and linear implication $\multimap$ are used to specify how the meaning of a clause headed by the verb is composed from the meanings of the arguments of the verb. For the moment, we can think of the linear connectives as playing the same role as the analogous classical connectives conjunction and implication, but we will soon see that the specific properties of the linear connectives are essential to guarantee that lexical entries bring into the interpretation process all and only the information provided by the corresponding words. The semantic contribution of *appointed* asserts that if the subject of a clause with main verb *appointed* means X and its object means Y, then the whole clause means $appoint(X, Y)$. The semantic contribution can thus

[1] In fact, we believe that the correct treatment of the relation between a verb and its arguments requires the use of *mapping principles* specifying the relation between the array of semantic arguments required by a verb and their possible syntactic realizations (Bresnan and Kanerva, 1989; Alsina, 1993; Butt, 1993). A verb like *appoint*, for example, might specify that one of its arguments is an agent and the other is a theme. Mapping principles would then specify that agents can be realized as subjects and themes as objects.

Here we make the simplifying assumption that the arguments of verbs have already been linked to syntactic functions and that this linking is represented in the lexicon, since for the examples we will discuss this assumption is innocuous. However, in the case of *complex predicates* this assumption produces incorrect results, as shown by Butt (1993). Mapping principles are very naturally incorporated into the framework discussed here; see Dalrymple, Lamping, and Saraswat (1993) and Dalrymple, Hinrichs, Lamping, and Saraswat (1993) for discussion and illustration.

be thought of as a linear definite clause, with the variables X and Y playing the same role as Prolog variables.

It is worth noting that the form of the semantic contribution of *appointed* parallels the type $e \times e \rightarrow t$ which, in its curried form $e \rightarrow e \rightarrow t$, is the standard type for a transitive verb in a compositional semantics setting (Gamut, 1991). In general, the propositional structure of the semantic contributions of lexical entries will parallel the types assigned to the meanings of the same words in compositional analyses.

Given the semantic contributions in (5), we can derive deductively the meaning for example (1). Let the constants f, g and h name the following f-structures:

(6)
$$f\!:\!\begin{bmatrix} \textsc{pred} & \text{`appoint'} \\ \textsc{subj} & g\!:\!\begin{bmatrix} \textsc{pred} & \text{`Bill'} \end{bmatrix} \\ \textsc{obj} & h\!:\!\begin{bmatrix} \textsc{pred} & \text{`Hillary'} \end{bmatrix} \end{bmatrix}$$

Instantiating the lexical entries for *Bill*, *Hillary*, and *appointed* appropriately, we obtain the following semantic contributions, abbreviated as **bill**, **hillary**, and **appointed**:

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

These formulas show how the generic semantic contributions in the lexical entries are instantiated to reflect their participation in this particular f-structure. For example, since the entry *Bill* is used for f-structure g, the semantic contribution for *Bill* provides a meaning for g_σ. More interestingly, the verb *appointed* requires two pieces of information, the meanings of its subject and object, in no particular order, to produce a meaning for the clause. As instantiated, the f-structures corresponding to the subject and object of the verb are g and h, respectively, and f is the f-structure for the entire clause. Thus, the instantiated entry for *appointed* shows how to combine a meaning for g_σ (its subject) and h_σ (its object) to generate a meaning for f_σ (the entire clause).

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

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

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

$$
\begin{array}{lll}
(7) & & \mathbf{bill} \otimes \mathbf{hillary} \otimes \mathbf{appointed} \quad (Premises.) \\
& \vdash & \mathbf{bill\text{-}appointed} \otimes \mathbf{hillary} \\
& \vdash & f_\sigma \rightsquigarrow appoint(Bill, Hillary)
\end{array}
$$

At each step, universal instantiation and modus ponens are used.

In summary, each word in a sentence contributes a linear-logic formula relating the semantic projections of specific f-structures in the LFG analysis to representations of their meanings. From those 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 may yield different such conclusions, corresponding to semantic interpretation ambiguities.

Meaning and glue Our approach shares the order-independence of representations of semantic information by attribute-value matrices (Pollard and Sag, 1987; Fenstad et al., 1987; Pollard and Sag, 1993), while still allowing a well-defined treatment of variable binding and scope. We do this by distinguishing (1) a *language of meanings* and (2) a language for assembling meanings or *glue language*.

The language of meanings could be that of any appropriate logic, for instance Montague's intensional logic (Montague, 1974). The glue language, described below, is a fragment of linear logic. The semantic contribution of each lexical entry is represented by a linear-logic formula that can be understood as instructions in the glue language for combining the meanings of the lexical entry's syntactic arguments into the meaning of the f-structure headed by the entry. Glue formulas may also be contributed by some syntactic constructions, when properties of a construction as a whole and not just of its lexical elements are responsible for the interpretation of the construction; these cases include the semantics of relative clauses. We will not discuss construction-specific interpretation rules in this paper.

Appendix A gives further details on the syntax of the meaning and glue languages used in this paper.

Linear logic As we have just outlined, we use deduction in linear logic to assign meanings to sentences, starting from information about their functional structure and about the semantics of the words they contain. An approach based on linear logic, which crucially allows premises to commute, appears to be more compatible with the shallow and relatively free-form functional structure than are compositional approaches, which rely on deeply nested binary-branching immediate dominance relationships. As noted above, the use of linear logic as the system for assembling meanings permits a uniform treatment of a range

of natural language phenomena described by Dalrymple, Lamping, and Saraswat (1993), including modification, completeness and coherence,[2] and complex predicate formation.

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

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

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

It is this "resource-sensitivity" of natural language semantics—an expression is used exactly once in a semantic derivation—that linear logic can model. The basic insight underlying linear logic is that logical formulas are *resources* that are produced and consumed in the deduction process. This gives rise to a resource-sensitive notion of implication, the *linear implication* $\multimap$: the formula $A \multimap B$ can be thought of as an action that can *consume* (one copy of) A to produce (one copy of) B. Thus, the formula $A \otimes (A \multimap B)$ linearlyentails 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

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

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

To handle quantification, 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 (with at most one level of nesting of implication in antecedents). In fact, all but the determiner lexical entries are in the first-order subset of this fragment. This fragment 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 (1993); see also Saraswat (1993).

Relationship with Categorial Syntax and Semantics As suggested above, there are interesting connections between our approach and various systems of categorial syntax and semantics. The Lambek calculus (Lambek, 1958), introduced as a logic of syntactic combination, turns out to be a fragment of noncommutative multiplicative linear logic. If permutation is added to Lambek's system, its left- and right-implication connectives (\ and /) collapse into a single implication connective with behavior identical to $\multimap$. This undirected version of the Lambek calculus was developed by van Benthem (1988; 1991) to account for the semantic combination possibilities of phrase meanings.

Those systems and related ones (Moortgat, 1988; Hepple, 1990; Morrill, 1990) were developed as calculi of syntactic/semantic types, with propositional formulas representing syntactic categories or semantic types. Given the types for the lexical items in a sentence as assumptions, the sentence is syntactically well-formed in the Lambek calculus if the type of the sentence can be derived from the assumptions arranged as an ordered list. Furthermore, the Curry-Howard isomorphism between proofs and terms (Howard, 1980) allows the extraction of a term representing the meaning of the sentence from the proof that the sentence is well-formed (van Benthem, 1986). However, the Lambek calculus and its variants carry with them a particular view of syntactic structure that is not obviously compatible with the flatter f-structures proposed by LFG.

On the other hand, categorial semantics in the undirected Lambek calculus and other related commutative calculi provides an analysis of the possibilities of meaning combination independently of the syntactic realizations of those meanings, but does not provide a mechanism for relating semantic combination possibilities to the corresponding syntactic combination possibilities.

In more recent work, multidimensional and labeled deductive systems (Moortgat, 1992b; Morrill, 1993) have been proposed as refinements of the Lambek systems that are able to represent synchronized derivations involving multiple levels of representation, for instance a level of head-dependent representations and a level of syntactic functor-argument representations. However, these systems do not yet seem able to represent the connection between a flat syntactic representation in terms of grammatical functions and a function-argument semantic representation. As far as we can see, the problem in those systems is that at the type level it is not possible to express the link between particular syntactic structures (f-structures in our case) and particular contributions to meaning. The extraction of meanings from derivations following the Curry-Howard isomorphism that is standard in categorial systems demands that the order of syntactic combination coincide with the order of semantic combination so that functor-argument relations at the syntactic and semantic level are properly aligned.

Thus, while the "propositional skeleton" of an analysis in our system can be seen as a close relative of the corresponding categorial semantics derivation in the undirected Lambek calculus, the first-order part of our analysis (notably the f, g, and h in the example above) explicitly carries the connection between f-structures and their contributions to meaning. In this way, we can take advantage of the principled description of potential meaning combinations of categorial semantics without losing track of the constraints imposed by syntax on the possible combinations of those meanings.

3 Quantification

Our treatment of quantification, and in particular of quantifier scope ambiguity and of the interactions between scope and bound anaphora, follows the approach of Pereira (1990; 1991). It turns out, however, that the linear-logic formulation is simpler and easier to justify than the earlier analysis, which used an intuitionistic type assignment logic.

The basic idea for the analysis can be seen as a logical counterpart at the glue level of the standard type assignment for generalized quantifiers (Barwise and Cooper, 1981). The generalized quantifier meaning of a natural language determiner has the following type, a function from

two properties, the quantifier's restriction and scope, to a proposition:

(8) $(e \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow t$

At the semantic glue level, we can understand that type as follows. For any determiner, if for arbitrary x we can construct a meaning Rx for the quantifier's restriction, and again for arbitrary x we can construct a meaning Sx for the quantifier's scope, where R and S are properties (functions from entities to propositions), then we can construct the meaning QRS for the whole sentence containing the determiner, where Q is the meaning of the determiner. In the following we will notate QRS meaning more perspicuously as $Q(z, Rz, Sz)$.

Assume that we have determined the following semantic structures: *restr* for the restriction (a common noun phrase), *restr-arg* for its implicit argument, *scope* for the scope of quantification, and *scope-arg* for the grammatical function filled by the quantified noun phrase. Then the foregoing analysis can be represented in linear logic by the following schematic formula:

(9) $$\begin{aligned} \forall R, S.\ & (\forall x.\ \mathit{restr\text{-}arg} \rightsquigarrow x \multimap \mathit{restr} \rightsquigarrow Rx) \\ & \otimes (\forall x.\ \mathit{scope\text{-}arg} \rightsquigarrow x \multimap \mathit{scope} \rightsquigarrow Sx) \\ & \multimap \mathit{scope} \rightsquigarrow Q(z, Rz, Sz) \end{aligned}$$

Given the equivalence between $A \otimes B \multimap C$ and $A \multimap (B \multimap C)$, the propositional part of (9) parallels the generalized quantifier type (8).

In addition to providing a semantic type assignment for determiners, (9) uses glue language quantification to express how the meanings of the restriction and scope of quantification are determined and combined into the meaning of the quantified clause. The condition $\forall x.\ \mathit{restr\text{-}arg} \rightsquigarrow x \multimap \mathit{restr} \rightsquigarrow Rx$ specifies that, if for arbitrary x *restr-arg* has meaning x,[3] then *restr* has meaning Rx, that is, it gives the dependency of the meaning of a common noun phrase on its implicit argument. Property R is the representation of that dependency as a function. Similarly, the subformula $\forall x.\ \mathit{scope\text{-}arg} \rightsquigarrow x \multimap \mathit{scope} \rightsquigarrow Sx$ specifies the dependency of the meaning Sx of a semantic structure

[3] We use lower-case letters for *essentially universal* variables, that is, variables that stand for new local constants in a proof. We use capital letters for *essentially existential* variables, that is, Prolog-like variables that become instantiated to particular terms in a proof. In other words, essentially existential variables stand for specific but as yet unspecified terms, while essentially universal variables stand for arbitrary constants, that is, constants that could be replaced by *any* term while still maintaining the validity of the derivation. In the linear-logic fragment we use here, essentially existential variables arise from universal quantification with outermost scope, while essentially universal variables arise from universal quantification whose scope is a conjunct in the antecedent of an outermost implication.

scope on the meaning x of one of its arguments *scope-arg*. If both dependencies hold, then R and S are an appropriate restriction and scope for the determiner meaning Q.

Computationally, the nested universal quantifiers substitute unique new constants (eigenvariables) for the quantified variable x, and the nested implications try to prove their consequent with the antecedent added to the current set of assumptions. Higher-order unification (in this case, a fairly restricted version thereof (Miller, 1990)) is used to solve for the values of R and S that satisfy the nested implication, which cannot contain occurrences of the eigenvariables.

To complete our specification of the semantic contribution of a determiner, we need to see how it relates to the f-structure it contributes to. The f-structure for a quantified noun phrase has the general form

(10) $$f{:}\begin{bmatrix} \text{SPEC} & q \\ \text{PRED} & g{:}\cdots \end{bmatrix}$$

where SPEC is the determiner and PRED is the noun.

Since the meaning of a noun is a property (type $e \to t$), its semantic contribution has the form of an implication, just like a verb.[4] However, while for a verb the arguments and result of the verb meaning can be associated to projections of appropriate f-structures in the syntactic analysis of the verb's clause, there are no appropriate f-structures in the analysis of a noun phrase that can be associated with the argument and result of the noun's meaning. Instead, we take the semantic projection f_σ of the noun phrase to be structured with two attributes (f_σ VAR) and (f_σ RESTR), performing a comparable role to the attributes CM and PM in the semantic structure in Halvorsen's (1983) treatment of quantifiers. Using those attributes, the semantic contribution of a noun can be expressed in the form

$$\forall x.\ (\uparrow_\sigma \text{VAR}) \rightsquigarrow x \multimap (\uparrow_\sigma \text{RESTR}) \rightsquigarrow Px$$

where P is the meaning of the noun.

We can now describe how the semantic structures *restr-arg*, *restr*, *scope-arg* and *scope* in (9) relate to the f-structure. The contribution of determiner q is expressed in terms of f's semantic projection f_σ. To connect the restriction of the determiner with the noun, we take *restr-arg* = (f_σ VAR) and *restr* = (f_σ RESTR). Since f fills an appropriate argument position, *scope-arg* = f_σ. As for *scope*, the scope of a determiner is not explicitly given, so we can only say that it can be any

[4]For a discussion of relational nouns, whose meanings are relations rather than properties, see Section 3.4.

semantic structure,[5] subject to the constraint that the meaning associated to the semantic structure have proposition type t, that is, the semantic contribution should quantify universally over possible scopes. Therefore, the contribution of a determiner is[6]:

$$
\begin{array}{lll}
(11) & \forall H, R, S. & (\forall x. (\uparrow_\sigma \text{VAR}) \rightsquigarrow x \multimap (\uparrow_\sigma \text{RESTR}) \rightsquigarrow Rx) \\
& & \otimes (\forall x. \uparrow_\sigma \rightsquigarrow x \multimap H \rightsquigarrow_t Sx) \\
& & \multimap H \rightsquigarrow_t Q(z, Rz, Sz)
\end{array}
$$

where H ranges over semantic structures.

The VAR and RESTR components of the semantic projection for a quantified noun phrase in our analysis play a similar role to the $/\!/$ category constructor in PTQ (Montague, 1974), that of distinguishing syntactic configurations with identical semantic types but different contributions to the interpretation. The two PTQ syntactic categories t/e for intransitive verb phrases and $t/\!/e$ for common noun phrases correspond to the single semantic type $e \rightarrow t$; similarly, the two conjuncts in the antecedent of (11) correspond to the same semantic type, encoded with a linear implication, but to two different syntactic contexts, one relating the predication of a noun phrase to its implicit argument and one relating a clause to an embedded argument.

3.1 Quantified noun phrase meanings

We first demonstrate how the semantic contribution of a quantified noun phrase such as *every voter* is derived. The following annotated

[5] Because of this indeterminacy as to choice of scope, Halvorsen and Kaplan (1988) used *inside-out functional uncertainty* to nondeterministically choose a scope f-structure for a quantifier. Their approach requires the scope of a quantifier to be an f-structure which contains the quantifier f-structure. In contrast, our approach places no syntactic constraints whatever on the choice of quantifier scope, since the propositional structure of the formulas involved in the derivation will preclude all but the appropriate choices.

[6] There is an alternative formulation of quantifier meaning that doesn't use nested implications:

$$
\begin{array}{l}
\exists x. (\uparrow_\sigma \text{VAR}) \rightsquigarrow x \\
\quad \otimes \forall H, R. (\uparrow_\sigma \text{RESTR}) \rightsquigarrow Rx \multimap (\uparrow_\sigma \rightsquigarrow x \otimes \forall S. (H \rightsquigarrow Sx) \multimap H \rightsquigarrow Q(z, Rz, Sz))
\end{array}
$$

This formulation just asserts that there is a generic entity, x, which stands for the meaning of the quantified phrase, and also serves as the argument of the restriction. The derivations of the restriction and scope are then expected to consume this information. By avoiding nested implications, this formulation may be easier to work with computationally.

However, the logical structure of this formulation is not as restrictive as that of (11), as it can allow additional derivations where information intended for the restriction can be used by the scope. In fact, though, for many analyses (including all those that we have investigated) such interactions are already precluded by the quantificational structure of the formula, and in such cases the formulation above is equivalent to (11).

phrase structure rule is necessary:

(12) $\text{NP} \longrightarrow \underset{\uparrow = \downarrow}{\text{Det}} \quad \underset{\uparrow = \downarrow}{\text{N}}$

This rule states that the determiner Det and noun N each contribute to the f-structure for the NP. Lexical specifications ensure that the noun contributes the PRED attribute and its value, and the determiner contributes the SPEC attribute and its value. The f-structure for the noun phrase *every voter* is:

(13) $$h{:}\begin{bmatrix} \text{SPEC} & \text{`every'} \\ \text{PRED} & \text{`voter'} \end{bmatrix}$$

The lexical entries used in this f-structure are:

(14) *every* Det $(\uparrow \text{SPEC}) = \text{`every'}$
$$\begin{aligned} &\forall H, R, S.\ (\forall x.\ (\uparrow_\sigma \text{VAR}) \rightsquigarrow x \multimap (\uparrow_\sigma \text{RESTR}) \rightsquigarrow Rx) \\ &\qquad \otimes (\forall x.\ \uparrow_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Sx) \\ &\qquad \multimap H \rightsquigarrow \mathit{every}(z, Rz, Sz) \end{aligned}$$

(15) *voter* N $(\uparrow \text{PRED}) = \text{`voter'}$
$$\forall X.\ (\uparrow_\sigma \text{VAR}) \rightsquigarrow X \multimap (\uparrow_\sigma \text{RESTR}) \rightsquigarrow \mathit{voter}(X)$$

The semantic contributions of common nouns and determiners were described in the previous section.

Given those entries, the semantic contributions of *every* and *voter* in (13) are

$$\begin{aligned} \textbf{every}: \quad &\forall H, R, S.\ (\forall x.\ (h_\sigma \text{VAR}) \rightsquigarrow x \multimap (h_\sigma \text{RESTR}) \rightsquigarrow Rx) \\ &\qquad \otimes (\forall x.\ h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Sx) \\ &\qquad \multimap H \rightsquigarrow \mathit{every}(z, Rz, Sz) \\ \textbf{voter}: \quad &\forall X.\ (h_\sigma \text{VAR}) \rightsquigarrow X \multimap (h_\sigma \text{RESTR}) \rightsquigarrow \mathit{voter}(X) \end{aligned}$$

From these two premises, the semantic contribution for *every voter* follows:

$$\begin{aligned} \textbf{every-voter}: \quad &\forall H, S.\ (\forall x.\ h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Sx) \\ &\qquad \multimap H \rightsquigarrow \mathit{every}(z, \mathit{voter}(z), Sz) \end{aligned}$$

The propositional part of this contribution corresponds to the standard type for noun phrase meanings, $(e \rightarrow t) \rightarrow t$. Informally, the whole contribution can be read as follows: if by giving the arbitrary meaning x of type e to the argument position filled by the noun phrase we can derive the meaning Sx of type t for the semantic structure scope of quantification H, then S can be the property that the noun phrase meaning

requires as its scope, yielding the meaning $every(z, voter(z), Sz)$ for H. The quantified noun phrase can thus be seen as providing two contributions to an interpretation: locally, a *referential import* x, which must be discharged when the scope of quantification is established; and globally, a *quantificational import* of type $(e \rightarrow t) \rightarrow t$, which is applied to the meaning of the scope of quantification to obtain a quantified proposition.

3.2 Simple example of quantification

Before we look at quantifier scope ambiguity and interactions between scope and bound anaphora, we demonstrate the basic operation of our proposed representation of the semantic contribution of a determiner. We use the following sentence with a single quantifier and no scope ambiguities:

(16) Bill convinced every voter.

To carry out the analysis, we need a lexical entry for *convinced*:

(17) *convinced* V $(\uparrow \textsc{pred}) = \text{`convince'}$
$(\uparrow \textsc{tense}) = \textsc{past}$
$\forall X, Y.\ (\uparrow \textsc{subj})_\sigma \rightsquigarrow X \otimes (\uparrow \textsc{obj})_\sigma \rightsquigarrow Y \multimap \uparrow_\sigma \rightsquigarrow convince(X, Y)$

The f-structure for (16) is:

(18)
$$f{:}\begin{bmatrix} \textsc{pred} & \text{`convince'} \\ \textsc{tense} & \textsc{past} \\ \textsc{subj} & g{:}\begin{bmatrix} \textsc{pred} & \text{`Bill'} \end{bmatrix} \\ \textsc{obj} & h{:}\begin{bmatrix} \textsc{spec} & \text{`every'} \\ \textsc{pred} & \text{`voter'} \end{bmatrix} \end{bmatrix}$$

The premises for the derivation are the semantic contributions for *Bill* and *convinced* together with the contribution derived above for the quantified noun phrase *every voter*:

bill: $g_\sigma \rightsquigarrow Bill$

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

every-voter: $\forall H, S.\ (\forall x.\ h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Sx)$
$\multimap\ H \rightsquigarrow every(z, voter(z), Sz)$

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

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

we have the derivation:

$$\begin{array}{ll} & \textbf{bill} \otimes \textbf{convinced} \otimes \textbf{every-voter} \quad \text{(Premises.)} \\ \vdash & \textbf{bill-convinced} \otimes \textbf{every-voter} \\ \vdash & f_\sigma \rightsquigarrow every(z, voter(z), convince(Bill, z)) \end{array}$$

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

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

could at first sight be considered another possible, but erroneous, scope. However, 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, while this formula represents the dependency of an individual on an individual (itself). Therefore, it does not provide a valid scope for the quantifier.

3.3 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 the sentence[7]

(19) Every candidate appointed a manager

for which we need the additional lexical entries

(20) *a* Det $(\uparrow \textsc{spec}) = \text{`a'}$
$$\begin{array}{l} \forall H, R, S.\ (\forall x.\ (\uparrow_\sigma \textsc{var}) \rightsquigarrow x \multimap (\uparrow_\sigma \textsc{restr}) \rightsquigarrow Rx) \\ \qquad \otimes (\forall x.\ \uparrow_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Sx) \\ \qquad \multimap H \rightsquigarrow a(z, Rz, Sz) \end{array}$$

(21) *candidate* N $(\uparrow \textsc{pred}) = \text{`candidate'}$
$$\forall X.\ (\uparrow_\sigma \textsc{var}) \rightsquigarrow X \multimap (\uparrow_\sigma \textsc{restr}) \rightsquigarrow candidate(X)$$

(22) *manager* N $(\uparrow \textsc{pred}) = \text{`manager'}$
$$\forall X.\ (\uparrow_\sigma \textsc{var}) \rightsquigarrow X \multimap (\uparrow_\sigma \textsc{restr}) \rightsquigarrow manager(X)$$

The f-structure for sentence (19) is

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

(23)

$$f\!:\!\begin{bmatrix} \textsc{pred} & \text{`appoint'} \\ \textsc{tense} & \textsc{past} \\ \textsc{subj} & g\!:\!\begin{bmatrix} \textsc{spec} & \text{`every'} \\ \textsc{pred} & \text{`candidate'} \end{bmatrix} \\ \textsc{obj} & h\!:\!\begin{bmatrix} \textsc{spec} & \text{`a'} \\ \textsc{pred} & \text{`manager'} \end{bmatrix} \end{bmatrix}$$

We can derive semantic contributions for *every candidate* and *a manager* in the way shown in Section 3.1. Further derivations proceed from those contributions together with the contribution of *appointed*:

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

As of yet, we have not made any commitment about the scopes of the quantifiers; the $\forall S$'s have not been instantiated. Scope ambiguities are manifested in two different ways in our system: through the choice of different semantic structures H, corresponding to different syntactic choices for where to scope the quantifier, or through different relative orders of 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**:

$$\begin{array}{l} \forall X.\ g_\sigma \rightsquigarrow X \multimap (\forall Y.\ h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow \mathit{appoint}(X, Y)) \\ \forall Y.\ h_\sigma \rightsquigarrow Y \multimap (\forall X.\ g_\sigma \rightsquigarrow X \multimap f_\sigma \rightsquigarrow \mathit{appoint}(X, Y)) \end{array}$$

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:

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

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

We select *a manager* to take narrower scope by using universal instantiation and transitivity of implication to combine the first form with **a-manager** to yield

$$\begin{array}{ll} \textbf{appointed-a-manager}: & \forall X.\ g_\sigma \rightsquigarrow X \\ & \quad \multimap f_\sigma \rightsquigarrow_t a(z, \mathit{manager}(z), \mathit{appoint}(X, z)) \end{array}$$

We have thus the following derivation

$$\begin{array}{ll} & \textbf{every-candidate} \otimes \textbf{appointed} \otimes \textbf{a-manager} \\ \vdash & \textbf{every-candidate} \otimes \textbf{appointed-a-manager} \\ \vdash & f_\sigma \rightsquigarrow_t \mathit{every}(w, \mathit{candidate}(w), a(z, \mathit{manager}(z), \mathit{appoint}(w, z))) \end{array}$$

of the $\forall\exists$ reading of (19).

Alternatively, we could have chosen *every candidate* to take narrow scope, by combining the second equivalent form of **appointed** with **every-candidate** to produce:

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

This gives the derivation

$$\begin{array}{ll} & \textbf{every-candidate} \otimes \textbf{appointed} \otimes \textbf{a-manager} \\ \vdash & \textbf{every-candidate-appointed} \otimes \textbf{a-manager} \\ \vdash & f_\sigma \rightsquigarrow_t a(z, \mathit{manager}(z), \mathit{every}(w, \mathit{candidate}(w), \mathit{appoint}(w, z))) \end{array}$$

for the $\exists\forall$ reading. These are the only two possible outcomes of the derivation of a meaning for (19), as required. We have used our implementation to verify that no other outcomes are possible, since manual verification would be rather laborious.

3.4 Constraints on quantifier scoping

Sentence (24) contains two quantifiers and therefore might be expected to show a two-way ambiguity analogous to the one described in the previous section:

(24) Every candidate appointed an admirer of his.

However, no such ambiguity is found if the pronoun *his* is taken to corefer with the subject *every candidate*. In this case, only one reading is available, in which *an admirer of his* takes narrow scope. Intuitively, this noun phrase may not take wider scope than the quantifier *every candidate*, on which its restriction depends.

As we will soon see, the lack of a wide scope *a* reading follows automatically from our formulation of the semantic contributions of quantifiers without further stipulation. In Pereira's earlier work on deductive interpretation (Pereira, 1990; Pereira, 1991), the same result was achieved through constraints on the relative scopes of glue-level universal quantifiers representing the dependencies between meanings

of clauses and the meanings of their arguments. Here, although universal quantifiers are used to support the extraction of properties representing the meanings of the restriction and scope (the variables R and S in the determiner lexical entries), the blocking of the unwanted reading follows from the propositional structure of the glue formulas, specifically the nested linear implications. This is more satisfactory, since it does not reduce the problem of proper quantifier scoping in the object language to the same problem in the metalanguage.

The lexical entry for *admirer* is:

(25) *admirer* N $(\uparrow \textsc{pred}) = \text{`admirer'}$
$$\forall X, Y.\ (\uparrow_\sigma \textsc{var}) \rightsquigarrow X \otimes (\uparrow \textsc{obl}_{\textsc{of}})_\sigma \rightsquigarrow Y \multimap (\uparrow_\sigma \textsc{restr}) \rightsquigarrow \mathit{admirer}(X, Y)$$

Here, *admirer* is a relational noun taking as its oblique argument a phrase with prepositional marker *of*, as indicated in the f-structure by the attribute $\textsc{obl}_{\textsc{of}}$. The semantic contribution for a relational noun has, as expected, the same propositional form as the binary relation type $e \times e \to t$: one argument is the admirer, and the other argument is the admiree.

We assume that the semantic projection for the antecedent of the pronoun *his* has been determined by some separate mechanism and recorded as the ANT attribute of the pronoun's semantic projection.[8] The semantic contribution of the pronoun is, then, a formula that consumes the meaning of its antecedent and then reintroduces that meaning, simultaneously assigning it to its own semantic projection:

(26) *his* N $(\uparrow \textsc{pred}) = \text{`pro'}$
$$\forall X.\ (\uparrow_\sigma \textsc{ant}) \rightsquigarrow X \multimap (\uparrow_\sigma \textsc{ant}) \rightsquigarrow X \otimes \uparrow_\sigma \rightsquigarrow X$$

In other words, the semantic contribution of a pronoun copies the meaning X of its antecedent as the meaning of the pronoun itself. Since the left-hand side of the linear implication "consumes" the antecedent meaning, it must be reinstated in the consequent of the implication. The f-structure for example (24) is, then:

[8] The determination of appropriate values for ANT requires a more more detailed analysis of other linguistic constraints on anaphora resolution, which would need further projections to give information about, for example, discourse relations and salience. Dalrymple (1993) discusses in detail LFG analyses of anaphoric binding.

(27)
$$f\!:\left[\begin{array}{ll}
\textsc{pred} & \text{`appointed'}\\
\textsc{tense} & \textsc{past}\\
\textsc{subj} & g\!:\left[\begin{array}{ll}\textsc{spec} & \text{`every'}\\ \textsc{pred} & \text{`candidate'}\end{array}\right]\\
\textsc{obj} & h\!:\left[\begin{array}{ll}\textsc{spec} & \text{`a'}\\ \textsc{pred} & \text{`admirer'}\\ \textsc{obl}_{\textsc{of}} & i\!:\left[\begin{array}{ll}\textsc{pred} & \text{`pro'}\end{array}\right]\end{array}\right]
\end{array}\right]$$

with $(i_\sigma\ \textsc{ant}) = g_\sigma$.

We will begin by illustrating the derivation of the meaning of *an admirer of his*, starting from the following premises:

a: $\forall H, R, S.\ (\forall x.\ (h_\sigma\ \textsc{var}) \rightsquigarrow x \multimap (h_\sigma\ \textsc{restr}) \rightsquigarrow Rx)$
$\otimes (\forall x.\ h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Sx)$
$\multimap H \rightsquigarrow a(z, Rz, Sz)$

admirer: $\forall Z, X.\ (h_\sigma\ \textsc{var}) \rightsquigarrow Z \otimes i_\sigma \rightsquigarrow X$
$\multimap (h_\sigma\ \textsc{restr}) \rightsquigarrow \mathit{admirer}(Z, X)$

his: $\forall X.\ g_\sigma \rightsquigarrow X \multimap g_\sigma \rightsquigarrow X \otimes i_\sigma \rightsquigarrow X$

First, we rewrite **admirer** into the equivalent form

$$\forall X.\ i_\sigma \rightsquigarrow X \multimap (\forall Z.\ (h_\sigma\ \textsc{var}) \rightsquigarrow Z \multimap (h_\sigma\ \textsc{restr}) \rightsquigarrow \mathit{admirer}(Z, X))$$

We can use this formula to rewrite the the second conjunct in the consequent of **his**, yielding

admirer-of-his:
$\forall X.\ g_\sigma \rightsquigarrow X \multimap$
$g_\sigma \rightsquigarrow X \otimes$
$(\forall Z.\ (h_\sigma\ \textsc{var}) \rightsquigarrow Z \multimap (h_\sigma\ \textsc{restr}) \rightsquigarrow \mathit{admirer}(Z, X))$

In turn, the second conjunct in the consequent of **admirer-of-his** matches the first conjunct in the antecedent of **a** given appropriate variable substitutions, allowing us to derive

an-admirer-of-his:
$\forall X.\ g_\sigma \rightsquigarrow X \multimap$
$g_\sigma \rightsquigarrow X \otimes (\forall H, S.\ (\forall x.\ h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Sx) \multimap$
$H \rightsquigarrow a(z, \mathit{admirer}(z, X), Sz))$

At this point the other formulas available are:

every-candidate:
$\forall H, S.\ (\forall x.\ g_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Sx)$
$\multimap H \rightsquigarrow \mathit{every}(z, \mathit{candidate}(z), Sz)$

appointed:
$\forall Z, Y.\ g_\sigma \rightsquigarrow Z \otimes h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow \mathit{appoint}(Z, Y)$

We have thus the meanings of the two quantified noun phrases. The antecedent implication of **every-candidate** has an atomic conclusion and hence cannot be satisfied by **an-admirer-of-his**, which has a conjunctive conclusion. Therefore, the only possible move is to combine **appointed** and **an-admirer-of-his**. We do this by first putting **appointed** in the equivalent form

$$\forall Z.\ g_\sigma \rightsquigarrow Z \multimap (\forall Y.\ h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow \mathit{appoint}(Z, Y))$$

After universal instantiation of Z with X, this can be used to rewrite the first conjunct in the consequent of **an-admirer-of-his** to derive

$$\begin{array}{l} \forall X.\ g_\sigma \rightsquigarrow X \multimap \\ \qquad (\forall Y.\ h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow \mathit{appoint}(X, Y)) \otimes \\ \qquad (\forall H, S.\ (\forall x.\ h_\sigma \rightsquigarrow x \multimap H \rightsquigarrow Sx) \multimap H \rightsquigarrow a(z, \mathit{admirer}(z, X), Sz)) \end{array}$$

Universal instantiation of H and S together with modus ponens with the two conjuncts in the consequent as premises yield

$$\forall X.\ g_\sigma \rightsquigarrow X \multimap f_\sigma \rightsquigarrow_t a(z, \mathit{admirer}(z, X), \mathit{appoint}(X, z))$$

Finally, this formula can be combined with **every-candidate** to give the meaning of the whole sentence:

$$f_\sigma \rightsquigarrow_t \mathit{every}(w, \mathit{candidate}(w), a(z, \mathit{admirer}(z, w), \mathit{appoint}(w, z)))$$

In fact, this is the only derivable conclusion, showing that our analysis blocks those putative scopings in which variables occur outside the scope of their binders.

4 Conclusion

Our approach exploits the f-structure of LFG for syntactic information needed to guide semantic composition, and also exploits the resource-sensitive properties of linear logic to express the semantic composition requirements of natural language. The use of linear logic as the glue language in a deductive semantic framework allows a natural treatment of quantification which automatically gives the right results for quantifier scope ambiguities and interactions with bound anaphora.

The analyses discussed here show that our linear-logic encoding of semantic compositionality captures the interpretation constraints between quantified noun phrases, their scopes and bound anaphora. The same basic facts are also accounted for in other recent treatments of compositionality, in particular categorial analyses with discontinuous constituency connectives (Moortgat, 1992a). However, we show elsewhere (Dalrymple, Lamping, Pereira and Saraswat, 1994) that our

approach has advantages over those accounts, in that certain available readings of sentences with intensional verbs and quantified noun phrases that current categorial analyses cannot derive are readily produced in our analysis.

Recently, Oehrle (1993) independently proposed a multidimensional categorial system with types indexed so as to keep track of the syntax-semantic connections that we represent with $\rightsquigarrow$. Using proof net techniques due to Moortgat (1992b) and Roorda (1991), he maps categorial formulas to first-order clauses similar to our semantic contributions, except that the formulas arising from determiners lack the embedded implication. Oehrle's system models quantifier scope ambiguities in a way similar to ours, but it is not clear that it can account correctly for the interactions with anaphora, given the lack of implication embedding in the clausal representation used. A full comparison of the two systems is left for future work.

Acknowledgments

We thank Johan van Benthem, Bob Carpenter, Jan van Eijck, Angie Hinrichs, David Israel, Ron Kaplan, John Maxwell, Michael Moortgat, John Nerbonne, Stanley Peters, Henriette de Swart and an anonymous reviewer for discussions and comments. They are not responsible for any remaining errors, and we doubt that they will endorse all our analyses and conclusions, but we are sure that the end result is much improved for their help.

A Syntax of the Meaning and Glue Languages

The meaning language is based on Montague's intensional higher-order logic. In fact, in the present paper we just use an extensional fragment with the following syntax:

$$
\begin{array}{llcll}
\text{(M-terms)} & M & ::= & c & \text{(Constants)} \\
 & & \mid & x & \text{(Lambda-variables)} \\
 & & \mid & \lambda x\, M & \text{(Abstraction)} \\
 & & \mid & M\, M & \text{(Application)} \\
 & & \mid & X & \text{(Glue-language variables)}
\end{array}
$$

Terms are typed in the usual way; logical connectives such as *every* and *a* are represented by constants of appropriate type. For readability, we will often "uncurry" $M N_1 \cdots N_m$ as $M(N_1, \ldots, N_m)$. Note that we allow variables in the glue language to range over meaning terms.

The glue language refers to three kinds of terms: meaning terms, f-structures, and semantic or σ-structures. f- and σ-structures are

feature structures in correspondence (through projections) with constituent structure. Conceptually, feature structures are just functions which, when applied to attributes (a set of constants), return constants or other feature structures. In the following we let A range over some pre-specified set of attributes.

$$
\begin{array}{llcll}
\text{(F-terms)} & F & ::= & \uparrow & \text{(Indexical reference)} \\
 & & \mid & f \mid g \mid h \mid \cdots & \text{(F-structure constants)} \\
 & & \mid & (F A) & \text{(Attribute selection)}
\end{array}
$$

$$
\begin{array}{llcll}
(\sigma\text{-terms}) & S & ::= & F_\sigma & \text{(Semantic projection)} \\
 & & \mid & (S A) & \text{(Attribute selection)} \\
 & & \mid & H & \text{(Glue-language variable)}
\end{array}
$$

Glue-language formulas are built up using linear connectives from atomic formulas of the form $S \rightsquigarrow_\tau M$, whose intended interpretation is that the meaning associated with σ-structure S is denoted by term M of type τ. We omit the type subscript τ when it can be determined from context.

$$
\begin{array}{llcll}
\text{(Glue formulas)} & G & ::= & S \rightsquigarrow_\tau M & \text{(Basic assertion)} \\
 & & \mid & G \otimes G & \text{(Linear conjunction)} \\
 & & \mid & G \multimap G & \text{(Linear implication)} \\
 & & \mid & \forall X. G & \text{(Quantification over M-terms)} \\
 & & \mid & \forall H. G & (\text{Quantification over } \sigma\text{-terms})
\end{array}
$$

References

Alsina, Alex. 1993. *Predicate Composition: A Theory of Syntactic Function Alternations*. Ph.D. thesis, Stanford University.

Barwise, Jon and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219.

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

Butt, Miriam. 1993. *The Structure of Complex Predicates*. Ph.D. thesis, Stanford University.

Dalrymple, Mary. 1993. *The Syntax of Anaphoric Binding*. Number 36 in CSLI Lecture Notes. Center for the Study of Language and Information.

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.

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, John Lamping, Fernando C. N. Pereira, and Vijay Saraswat. 1994. Intensional verbs without type-raising or lexical ambiguity. In *Conference on Information-Oriented Approaches to Logic, Language and Computation*, Moraga, California. Saint Mary's College. To appear.

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

Gamut, L. T. F. 1991. *Logic, Language, and Meaning*, volume 2: Intensional Logic and Logical Grammar. The University of Chicago Press, Chicago.

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

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

Halvorsen, Per-Kristian. 1988. Situation Semantics and semantic interpretation in constraint–based grammars. In *Proceedings of the International Conference on Fifth Generation Computer Systems, FGCS-88*, pages 471–478, Tokyo, Japan, November. Also published as CSLI Technical Report CSLI-TR-101, Stanford University, 1987.

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

Hendriks, Herman. 1993. *Studied Flexibility: Categories and Types in Syntax and Semantics*. ILLC dissertation series 1993—5, University of Amsterdam, Amsterdam, Holland.

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

Howard, W.A. 1980. The formulae-as-types notion of construction. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, London, England, pages 479–490.

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

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

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.

Miller, Dale A. 1990. A logic programming language with lambda abstraction, function variables and simple unification. In Peter Schroeder-Heister, editor, *Extensions of Logic Programming*, Lecture Notes in Artificial Intelligence. Springer-Verlag.

Montague, Richard. 1974. The proper treatment of quantification in ordinary English. In Richmond H. Thomason, editor, *Formal Philosophy*. Yale University Press, New Haven, Connecticut.

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

Moortgat, Michael. 1992a. Generalized quantifiers and discontinuous type constructors. In W. Sijtsma and H. van Horck, editors, *Discontinuous Constituency*. Mouton de Gruyter, Berlin, Germany. To appear.

Moortgat, Michael. 1992b. Labelled deductive systems for categorial theorem proving. In P. Dekker and M. Stokhof, editors, *Proceedings of the Eighth Amsterdam Colloquium*, pages 403–423, Amsterdam. Institute for Logic, Language and Computation.

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

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

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

Oehrle, Richard T. 1993. String-based categorial type systems. Workshop "Structure of Linguistic Inference: Categorial and Unification-Based Approaches," European Summer School in Logic, Language and Information, Lisbon, Portugal.

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

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

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

Pollard, Carl and Ivan A. Sag. 1993. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago.

Roorda, Dirk. 1991. *Resource Logics: Proof-theoretical Investigations*. Ph.D. thesis, University of Amsterdam.

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*, pages 377–394. World Scientific Publishing Co.

van Benthem, Johan. 1986. Categorial grammar and lambda calculus. In D. Skordev, editor, *Mathematical Logic and its Application*. Plenum Press, New York, New York, pages 39–60.

van Benthem, Johan. 1988. The Lambek calculus. In Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler, editors, *Categorial Grammars and Natural Language Structures*. D. Reidel, Dordrecht, pages 35–68.

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