New directions for proof theory in linguistics

Chris Barker and Anna Szabolcsi
New York University, Linguistics
chris.barker@nyu.edu, anna.szabolcsi@nyu.edu
homepages.nyu.edu/~cb125, homepages.nyu.edu/~as109

Proof theory has been fruitfully applied in several areas of linguistics, including theories of syntactic well-formedness, ellipsis resolution, and cross-linguistic variation. We concentrate in this course on new work by various authors using proof theory for characterizing aspects of the syntax/semantics interface, including proof-theoretic accounts of phrase sub-typing for scope interactions; negative polarity and other types of licensing; the differential behavior of quantificational elements; and more. The research question addressed is: What insights does the proof theoretic approach offer in this particular empirical domain and to what extent should linguistic competence be viewed as a system of logical inference? Prerequisites should be familiarity with formal approaches to natural language semantics, including some exposure to formal logic.

Reader contents:


de Groote, Philippe. 2001. Type raising, continuations, and classical logic. 13th Amsterdam Colloquium. [preprint]


Scope Dominance with Upward Monotone Quantifiers

To appear in Journal of Logic, Language and Information

Alon Altman, Ya’acov Peterzil and Yoad Winter

Abstract

We give a complete characterization of the class of upward monotone generalized quantifiers $Q_1$ and $Q_2$ over countable domains that satisfy the scheme $Q_1 x \rightarrow Q_2 y \rightarrow Q_2 y Q_1 x$. This generalizes the characterization of such quantifiers over finite domains, according to which the scheme holds iff $Q_1$ is 3 or $Q_2$ is $\forall$ (excluding trivial cases). Our result shows that in infinite domains, there are more general types of quantifiers that support these entailments.

1 Introduction

A type 1 generalized quantifier over a domain $E$ is a set $Q \subseteq \wp(E)$. We henceforth refer to such sets more briefly as quantifiers. For instance, over a domain $E$ and some $X \subseteq E$, the following are the quantifiers that are more traditionally written as $\exists x \in X$ and $\forall x \in X$, respectively:

- $\exists X \{ A \subseteq E : X \cap A \neq \emptyset \}$
- $\forall X \{ A \subseteq E : X \subseteq A \}$

We call such quantifiers $\exists X$ ("existential") and $\forall X$, respectively. The quantifiers $Q$ that are both $\exists X$ and $\forall X$ are of the form $\{ A \subseteq E : x \in A \}$ for some $x \in E$.

which are precisely the principal ultrafilters over $E$.

When $Q_1$ and $Q_2$ are quantifiers and $R$ a binary relation, the formula $Q_1 x Q_2 y R(x, y)$ is often written $Q_1 Q_2 R$, which is interpreted in $E$ as follows.

1. $\{ x \in E : R_\ell \in Q_1 \} \in Q_1$,

where $R_\ell = \{ y \in E : R(x, y) \}$. Henceforth we will also use the notation $R^{-\ell}$ for $\{ x \in E : R(x, y) \}$, considering the following equivalence:

2. $Q_1 Q_2 R^{-\ell} \Leftrightarrow \{ y \in E : R_\ell \in Q_1 \} \in Q_1$.

Previous studies of generalized quantifiers have characterized various scope commutativity properties of quantifiers in constructions with multiple quantification. Notably, Westerståhl (1996) characterizes the class of self-commuting quantifiers – those quantifiers $Q$ that satisfy the following equivalence:

3. For all $R \subseteq E^3$: $Q Q R \Leftrightarrow Q Q R^{-\ell}$.

Zimmermann (1993) characterizes the class of scopeless quantifiers – those quantifiers $Q$ that satisfy the following equivalence.

4. For all $Q_1 \subseteq \wp(E)$, for all $R \subseteq E^3$: $Q_1 Q_1 R \Leftrightarrow Q_1 Q_1 R^{-\ell}$.

He shows that the scopeless quantifiers over $E$ are precisely the ultrafilters over $E$.

Westerståhl (1986) studies the more general problem of characterizing the quantifiers $Q_1, Q_2$ that satisfy the following unidirectional entailment.

5. For all $R \subseteq E^3$: $Q_1 Q_2 R \Rightarrow Q_3 Q_1 R^{-\ell}$.

When this entailment holds, we say that $Q_1$ is (scopally) dominant over $Q_2$.

We denote the complement of a quantifier $Q$ over $E$ by $\overline{Q} \overset{\text{df}}{=} \wp(E) \backslash Q$. Keenan (1993) defines the postcomplement of a quantifier $Q$ over $E$ as the set $Q^{-} \overset{\text{df}}{=} \{ A \subseteq E : E \setminus A \subseteq Q \}$. The dual $Q^d$ (cf. Barwise and Cooper (1981)) of a quantifier $Q$ is the complement of $Q$’s postcomplement:

$$Q^d \overset{\text{df}}{=} (Q^{-})^c = \{ A \subseteq E : E \setminus A \notin Q \}.$$
Note that for any quantifier $Q$: $(Q^d)^d = Q$ and $Q$ is EXIST iff $Q^d$ is UNIV. Further, over a domain $E$ the two trivial quantifiers – $\emptyset$ and $\emptyset(E)$ – are each other’s duals. As the following simple fact shows, there is a close relation between quantifier duality and scope dominance.

**Fact 1** For all quantifiers $Q_1$ and $Q_2$: $Q_1$ is dominant over $Q_2$ iff $Q_1^d$ is dominant over $Q_2^d$.

This fact follows directly from the definition of scope dominance and duality, and the observation that for any $R \subseteq E^3$ we have:

$$Q_1 Q_2 R \Leftrightarrow \neg (Q_2^d \setminus Q_1^d \setminus (E^3 \setminus R)).$$

For the sake of completeness we give in section 2 a simple proof of Westerståhl’s characterization of dominance between quantifiers in finite domains. The main part of the paper is section 3, where this characterization is extended to countable domains. Section 4 concludes with some remarks about scope commutativity, finiteness and monotonicity in natural language semantics.

## 2 Finite domains

Westerståhl’s characterization is restricted to upward monotone quantifiers over finite domains. Standardly, by saying that a quantifier $Q$ over $E$ is upward monotone we mean that $Q$ is closed under supersets: $A \in Q$ and $A \subseteq B$ implies $B \in Q$. Note that $Q$ is upward monotone iff $Q^d$ is. Under upward monotonicity and finiteness of the domain, Westerståhl’s claim can be stated as follows.\(^2\)

**Fact 2** Let $Q_2$ and $Q_2$ be upward monotone quantifiers over a finite domain $E$. $Q_2$ is dominant over $Q_2$ iff these quantifiers fall under at least one of the following cases.

(i) $Q_2$ is EXIST or $Q_2$ is UNIV.

(ii) $Q_2 = \emptyset(E)$ and $Q_2 \neq \emptyset$, or $Q_2 = \emptyset$ and $Q_2 \neq \emptyset(E)$.

Proof The “if” direction of the proof is easy, and does not require finiteness of the domain. For the “only if” direction, assume that $Q_1$ is dominant over $Q_2$. First it is easy to see that if $Q_1 = \emptyset$ then $Q_2 = \emptyset$ and (dually) that if $Q_2 = \emptyset$ then $Q_1 \neq \emptyset(E)$. Assume for contradiction that neither (i) nor (ii) holds. Then by finiteness of $E$ there is a minimal set $A \in Q_2$ such that $|A| \geq 2$ (otherwise by upward monotonicity, $Q_1 = \emptyset(E)$ or $Q_1 = \text{EXIST}(\cup_{x \in Q_1}(x))$). By the dual consideration, there are $B_1, B_2 \in Q_2$ such that $B_1 \cap B_2 \notin Q_2$. Given the sets $A, B_1$ and $B_2$, and an arbitrary $a \in A$, it is easy to verify that the relation $(\{a\} \times B_1) \cup (\{a\} \times B_2)$ contradicts our assumption that $Q_1$ is dominant over $Q_2$. □

Westerståhl (1996) calls two quantifiers $Q_1, Q_2 \subseteq \emptyset(E)$ independent if they satisfy the following equivalence.

(6) For all $R \subseteq E^3$: $Q_1 Q_2 R \Leftrightarrow Q_2 Q_1 R^{-1}$.

Using Fact 2 it is easy to establish the following corollary.

**Corollary 3** Let $Q_1$ and $Q_2$ be upward monotone quantifiers over a finite domain $E$. Then $Q_1$ and $Q_2$ are independent iff $Q_1$ and $Q_2$ fall under at least one of the following cases.

(i) $Q_1$ and $Q_2$ are EXIST, or $Q_2$ and $Q_2$ are UNIV.

(ii) $Q_1$ or $Q_2$ are principal ultrafilters.

(iii) $Q_1$ or $Q_2$ are trivial, and $Q_2 \neq \emptyset(Q_1)$.

Recall that the trivial quantifiers over a domain $E$ are $\emptyset(E)$ and $\emptyset$.

**Examples:** For illustrating scope dominance in simple natural language sentences, consider first a well-known type of example.

(7) Some priest visited every city.

Let us assume that the nouns priest and city are denoted by the sets $P, C \subseteq E$ respectively, and that the verb visited is denoted by the binary relation $V \subseteq E^3$. Sentence (7) has two readings, depending on the order in which the quantifiers operate on the arguments of the relation $V$:

1. Some priest visited every city.
2. Every city was visited by some priest.
The statement in (8a) is called the object narrow scope (ONS) reading of sentence (7), whereas the t statement in (8b) is called the object wide scope (OWS) reading of the sentence. As a matter of first-order logic, (8a) entails (8b) but not vice versa. Thus, the quantifier \( \exists X \) is dominant over the quantifier \( \forall Y \) for any \( P, C \subseteq E \), but the opposite does not hold.

The situation is similar in cases where (exactly) one of the existential/universal quantifiers is replaced by another upward monotone quantifier, not necessarily first-order. The sentences in (9) below illustrate some cases like that, where the ONS reading entails the OWS reading. The corresponding quantifiers we assume are given in (10).

\[
\begin{align*}
(9) & \quad \text{a. At least half/ at least two/all but at most five of the priests visited every city.} \\
& \quad \text{b. Some priest visited at least half/ at least two/all but at most five of the cities.}
\end{align*}
\]

\[
\begin{align*}
(10) & \quad \text{\# at least}_X & = \{ A \subseteq E : |X \cap A| \geq |X \setminus A| \} \\
& \quad \text{\# at most}_X & = \{ A \subseteq E : |X \cap A| \geq n \} \\
& \quad \text{\# at least}_X & = \{ A \subseteq E : |A \setminus X| \leq n \}
\end{align*}
\]

Note that the quantifier \( \text{\# at least}_X \) is not first-order definable.

Westerstål’s result shows that over finite domains, the \( \exists X \) quantifiers (for \( Q_1 \)) and the \( \forall Y \) quantifiers (for \( Q_2 \)) are the only non-trivial upward monotone quantifiers that lead to entailments as in (5). Thus, the sentences in (7) and (9) are representative of the cases where upward monotone quantifiers lead to an entailment from the ONS reading to the OWS reading on finite domains.

\section{Countable domains}

As Westerstål observes, his characterization of scope dominance over finite domains in Fact 2 does not hold for infinite domains. Thus, over infinite domains there are non-trivial upward monotone quantifiers besides the \( \exists X \) and \( \forall Y \) quantifiers that give rise to scope dominance. Consider the following example (following Westerstål), where \( E \) is assumed to be countable.

\[
(11) \text{Infinitely many dots are contained in at least one of the three circles.}
\]

\[
\begin{align*}
Q_1 & = \{ A \subseteq E : |D \cap A| = \infty \} \\
Q_2 & = \{ A \subseteq E : C \cap A \neq \emptyset \}, \text{ where } |C| = 3
\end{align*}
\]

It is easy to verify that \( Q_2 \) is dominant over \( Q_1 \), but \( Q_2 \) and \( Q_1 \) are upward monotone and the conditions in Fact 2 do not hold. Incidentally, since \( Q_1 \) is \( \exists X \) and the conditions in Fact 2 do not hold. Incidentally, since \( Q_2 \) is \( \forall Y \), it is dominant over \( Q_2 \) and the intersection \( \bigcap A_1 \) is in \( Q_2 \) as well. For example, any \( \exists X \) quantifier satisfies (DCC). A quantifier \( \exists X \) satisfies (DCC) if and only if \( X \) is finite. Another quantifier that satisfies (DCC) is the following, where the domain \( E = \emptyset \) is the set of natural numbers:

\[
\{ A \subseteq \mathbb{N} : \forall n \in \mathbb{N} \forall 2n \in A \lor 2n + 1 \in A \}
\]

If every set in a quantifier \( Q \) contains a finite subset that is also in \( Q \), we say that \( Q \) satisfies (FIN). The following fact shows that for upward monotone quantifiers over countable domains, the (FIN) property is dual to (DCC).

\textbf{Fact 4} For any upward monotone quantifier \( Q \) over a countable domain \( E : Q \) satisfies (DCC) iff \( Q^d \) satisfies (FIN).

\textbf{Proof} Assume that \( Q \) satisfies (DCC) and assume for contradiction that there is \( A \in Q^d \) such that for all \( B \subseteq A : \text{if } B \in Q^d \text{ then } B \text{ is infinite. Let } B_1, A \text{ be a finite set.} \)
Hence \( B_i \notin Q^d \), and \( E \setminus B_i \in Q \). By countability of \( E \), we can denote \( A \setminus B_i = \{e_i\}_{i \in \mathbb{N}} \). Let \( B_{n+1} = B_1 \cup \{e_{n+1}\} \), for any \( i \geq 0 \). By our assumption on \( A \) we have \( B_i \notin Q^d \) for any \( i \geq 0 \), hence \( E \setminus B_i \in Q \) for any \( i \geq 0 \). But \( \bigcap (E \setminus B_i) = E \setminus A \notin Q \), in contradiction to \( Q \) satisfying (DCC).

Conversely, assume that \( Q^d \) satisfies (FIN). Let \( B_1 \supseteq B_2 \supseteq \ldots \) be a descending chain in \( Q \), so \( E \setminus B_i \notin Q^d \) for any \( i \geq 1 \). Assume leading to a contradiction that \( B = \bigcap B_i \notin Q \), thus \( E \setminus B = \bigcup (E \setminus B_i) \in Q^d \). By (FIN) there is a finite \( A' \subseteq Q^d \) s.t. \( A' \subseteq \bigcup (E \setminus B_i) \). Hence for some \( n, A' \subseteq E \setminus B_n \), and from the upward monotonicity of \( Q \), and hence of \( Q^d \), \( E \setminus B_n \in Q^d \), a contradiction. □

These two pairs of dual properties will be used in the proof of the following theorem, which is the main result of this paper.

**Theorem 5** Let \( Q_1 \) and \( Q_2 \) be upward monotone quantifiers over a countable domain \( E \). Then \( Q_1 \) is dominant over \( Q_2 \) if and only if all of the following requirements hold:

(i) \( Q_1^d \) or \( Q_2 \) are closed under finite intersections;

(ii) \( Q_1^d \) or \( Q_2 \) satisfy (DCC);

(iii) \( Q_2^d \) or \( Q_2 \) are not empty.

**Proof**

For the “if” direction, assume that requirements (i)-(iii) hold. Consider first the case where \( Q_1^d \) is closed under finite intersections and \( Q_2 \) satisfies (DCC), where both \( Q_1^d \) and \( Q_2 \) are non-trivial.

Assume that \( A \seteq \{ x \in E : R_x \in Q_1 \} \) is in \( Q_1 \), and let \( B \seteq \{ y \in E : R_y \in Q_2 \} \).

We need to show that \( B \in Q_2 \). Since \( E \) is countable and \( Q_2 \) satisfies (DCC) it is sufficient to prove that for every finite \( F \subseteq E \setminus B \), we have \( E \setminus F \in Q_2 \).

For every \( b \notin B \), \( R_b \notin Q_1 \). Since \( Q_2 \) has property (U) (by assumption about \( Q_1^d \)) and \( E \in Q_1 \) (by upward monotonicity and non-triviality), we have \( E \setminus R_b \in Q_2 \). Thus, by the definition of \( R_b \) and \( R_x \), the set \( A_b \equiv \{ x : b \notin R_x \} \) is in \( Q_1 \).

Now for any \( F = \{ b_1, \ldots, b_n \} \subseteq E \setminus B \), the sets \( A_{b_1}, \ldots, A_{b_n} \) are all in \( Q_1 \), and since \( Q_1 \) is closed under finite intersections, we have \( A \cap A_{b_1} \cap \ldots \cap A_{b_n} \in Q_2 \).

Since \( Q_1 \) is non-trivial, this last set is non-empty, and hence there is \( x \in E \) such that \( R_x \in Q_2 \) and also \( F \cap R_x \notin Q_2 \). By the upward monotonicity of \( Q_2 \) it follows that \( E \setminus F \in Q_2 \).

For the other cases in requirements (i)-(iii), dominance of \( Q_1 \) over \( Q_2 \) now follows directly from Fact 1 about duality, and from the observation that for any non-empty quantifier \( Q \) over a countable domain: if \( Q \) is closed under finite intersections and satisfies (DCC), then \( Q \) is UNIV.

For the “only if” direction, assume that \( Q_1 \) is dominant over \( Q_2 \).

To show that (i) holds, assume that \( Q_1^d \) is not closed under finite intersections, so \( Q_2 \) does not satisfy (U). Hence, by upward monotonicity of \( Q_1 \), it contains a set \( A = A_1 \cup A_2 \) where \( A_1 \) and \( A_2 \) are disjoint and neither of them is in \( Q_2 \). To show that \( Q_2 \) is closed under finite intersections, let us denote for any \( B_1, B_2 \in Q_2 \):

\[
R = (A_1 \times B_1) \cup (A_2 \times B_2).
\]

We have \( \{ x : R_x \in Q_1 \} \subseteq \{ y : R_y \in Q_2 \} \). But, since \( A_1, A_2 \notin Q_2 \), we have \( B = B_1 \cap B_2 \), hence \( B \in Q_2 \).

To show that (ii) holds, assume that \( Q_2^d \) does not satisfy (DCC), so from Fact 4 it follows that there is a set \( A \in Q_2 \) such that every subset of \( A \) that is also in \( Q_2 \) is infinite. To show that \( Q_2 \) satisfies (DCC), let \( B_1 \supseteq B_2 \supseteq \ldots \supseteq B_n \supseteq \ldots \) be a sequence of sets in \( Q_2 \), and let \( B \) be their intersection. We again assume that \( E \) is the set of natural numbers, and enumerate \( A = \{ a_1, a_2, \ldots \} \). Consider the relation

\[
R = \bigcup_{n=1}^{\infty} \{(a_n) \times B_n\}.
\]

Then the set \( \{ x : R_x \in Q_2 \} = A \in Q_2 \), or \( Q_1, Q_2, R \), and by our assumption it follows that \( Q_1 Q_2 R \) holds, or \( \{ y : R_y \in Q_1 \} \in Q_2 \). We claim that this last set equals \( B \), so \( Q_2 \) satisfies (DCC). Clearly, for every \( y \in B : R_y = A \in Q_2 \). However, if \( y \notin B \) then there is \( n \) such that \( y \notin B_n \) for all \( m \geq n \), and therefore \( R_y \subseteq A \) is finite. By our previous observation, such \( R_y \) is not in \( Q_2 \).

Clause (iii) is easily seen to hold. □
From this theorem it is easy to conclude the following, more direct, classification of the upward monotone quantifiers $Q_1$, $Q_2$ that support scope dominance over countable domains. These are precisely the pairs of quantifiers $Q_1$ and $Q_2$ that satisfy at least one of the following requirements.

(12) (i) $Q_1$ is $\text{EXIST}$ or $Q_2$ is $\text{UNIV}$.

(ii) $Q_1$ satisfies $(U)$, $Q_1 \neq \emptyset$ and $Q_1$ satisfies (DCC), or $Q_1$ is closed under finite intersections, $Q_1 \neq \wp(E)$ and $Q_1$ satisfies (FIN).

(iii) $Q_1 = \wp(E)$ and $Q_2 \neq \emptyset$, or $Q_1 = \emptyset$ and $Q_1 \neq \wp(E)$.

That Theorem 5 is a generalization of Fact 2 for countable domains is obvious from clauses (i) and (ii) in this statement of the theorem. Clause (12)(iii) becomes redundant over finite domains, since over such domains the upward monotone quantifiers that satisfy $(U)$ are exactly the $\text{EXIST}$ quantifiers and $\wp(E)$, and the (DCC) requirement for $Q_2$ is trivially satisfied. Dually, over finite domains the upward monotone quantifiers that are closed under finite intersections are the $\text{UNIV}$ quantifiers and $\emptyset$, and the “finiteness” requirement for $Q_1$ is trivially satisfied. However, as we shall exemplify below, over infinite domains (also countably infinite), there are non-trivial non-$\text{EXIST}$ upward monotone quantifiers that satisfy $(U)$, and (dually) there are non-trivial non-$\text{UNIV}$ upward monotone quantifiers that are closed under finite intersections. Thus, clause (12)(ii) is where Theorem 5 generalizes Fact 2.

As in the case of scope dominance over finite domains (cf. Corollary 3), Theorem 5 allows us to characterize the pairs of independent quantifiers. To do so, let us first prove the following two lemmas.

Lemma 6 An upward monotone quantifier $Q$ over a countable domain $E$ satisfies both $(U)$ and (DCC) iff $Q = \wp(E)$ or $Q = \text{EXIST}(X)$ for some finite $X \subseteq E$.

Proof The proof of the “if” direction is easy. For the “only if” direction, assume that $Q$ satisfies $(U)$ and $Q \neq \wp(E)$. We will show that there are no minimal sets in $Q$ other than singletons. Let $A$ be some arbitrary set in $Q$. If $A$ is finite then it must contain a singleton in $Q$. If $A$ is infinite, then either it contains a singleton in $Q$, or by $(U)$ and the countability of $E$, we can form a descending chain of subsets of $A$, all in $Q$, whose intersection is empty. From (DCC) it follows that $\emptyset \in Q$ and by upward monotonicity $Q = \wp(E)$, in contradiction to our assumption. Thus, every set in $Q$ contains a singleton in $Q$, and if $X = \{x \in E : \{x\} \in Q\}$ then by upward monotonicity $Q = \text{EXIST}(X)$. Suppose for contradiction that $X$ is infinite, then again by (DCC), we conclude that $\emptyset \in Q$, contradiction. □

Lemma 7 If a quantifier $Q$ satisfies (DCC) and (FIN) then there are finitely many minimal sets in $Q$, all of them finite.

Proof By (FIN) it follows that the minimal sets in $Q$ are all finite. Assume for contradiction that there are infinitely many (finite) minimal sets in $Q$, and denote this collection of sets by $X$. It follows that for any $A, B \in X$ such that $A \neq B$, $A \cap B$ is a proper subset of both $A$ and $B$. Let $F_i$ be in $X$. Because $F_i$ is finite and $X$ is infinite, there must be some $F_i \subset \subset F_j$ such that the collection of sets $X_i = \{ F \in X : F \cap F_i = F_i^j \}$ is infinite. We can continue this process by defining $F_i, F_i^j$ and $X_i$ for every $i \geq 1$ as follows:

$F_{i+1}$ is some set in $X_i$.

$F_i^j$ is some proper subset of $F_{i+1}$ such that $\{ F \in X_i : F \cap F_{i+1} = F_i^{j+1} \}$ is finite.

$X_{i+1} \overset{df}{=} \{ F \in X_i : F \cap F_{i+1} = F_i^{j+1} \}$

We obtain an infinite sequence $F_1, \ldots, F_i, \ldots$ of finite minimal sets in $Q$, together with $F_1^j, \ldots, F_i^j, \ldots$ such that for every $i$, $F_i^j \subsetneq F_i$ and for every $m \geq n$, $F_m \cap F_n = F_m^j$. We now let $A_m = \bigcup_{n \geq m} F_m$. This is a decreasing sequence of sets, all in $Q$, so by (DCC), $A = \bigcap_{n \geq 1} A_n$ is in $Q$. We claim that $A = \bigcup_{n \geq 1} F_m^j$. Indeed, note that $A$ consists of all elements which belong to infinitely many sets $F_m$. Let $x$ be some element in $\bigcup_{n \geq 1} F_m^j$, thus $x \in F_m^j$ for some $n$. For every $m > n$, $x \in F_m$ because $F_m \in X_m$. Thus $x$ is in infinitely many sets $F_m$, and therefore $x \in A$. For the opposite direction, assume that $x \in A$, thus belongs to infinitely many $F_m$. In particular it belongs to some $F_m$, $F_n$ for $m \geq n$. But then $x \in (F_m \cap F_n) = F_m^j$, thus belonging to $\bigcup_{n \geq 1} F_m^j$.

By our assumption on $Q$, the set $A$ contains a finite subset $B \subset Q$. The set $B$ is then contained in the union of finitely many sets $F_m^j$, which implies that for some $m$
Because both $F_m$ and $B$ are in $Q$, this contradicts the minimality of $F_m$. □

Using Theorem 5 and the two lemmas above, the proof of the following claim is by a simple enumeration of cases.

**Corollary 8** Let $Q_1$ and $Q_2$ be upward monotone quantifiers over a countable domain $E$. Then $Q_1$ and $Q_2$ are independent if these two quantifiers or their duals $Q_1^\perp$ and $Q_2^\perp$ constitute a pair $S_1, S_2$, not necessarily in this order, which falls under at least one of the following cases.

(i) $S_1 = \text{EXIST}(X)$ for some $X \neq \emptyset$, s.t. $X$ is finite and $S_1$ satisfies (U), or $X$ is infinite and $S_1$ is $\text{EXIST}$.

(ii) $S_1$ or $S_2$ are principal ultrafilters.

(iii) For some finite collection $X \subseteq \wp(E)$ of finite sets, $S_1 = \bigcup_{X \in E} \text{UNIV}(X)$, and $S_1$ is an ultrafilter.

(iv) $S_1 = \emptyset$ and $S_1 \neq \wp(E)$.

**Remark:** Since we assume here the Axiom of Choice, non-principal ultrafilters exist over $E$, so (iii) is not subsumed by (ii).

**Examples:** First let us note that in example (11) above, $Q_2 = \text{EXIST}(C)$ for a finite $C$ ($|C| = 3$). $Q_1$ satisfies (U), hence $Q_1$ and $Q_2$ fall under clause (i) in Corollary 8, and the ONS reading of the sentence is equivalent to the OWS reading. The following example illustrates the dual case covered by clause (i) in Corollary 8, where $Q_1 = \text{UNIV}(C)$ for a finite $C$, and $Q_2$ is closed under finite intersections.

(13) Each of the three circles contains all but finitely many dots.

$$Q_1 = \{ A \subseteq E : C \subseteq A \}, \text{ where } |C| = 3$$

$$Q_2 = \{ A \subseteq E : |D \setminus A| < \aleph_1 \}$$

To illustrate non-trivial usages of clause (iii) in Corollary 8, we would have to use non-principal ultrafilters, which we here omit.

As for dominance between quantifiers without independence, the quantifiers in (14) and (15) below satisfy clause (ii) of (12). Hence, in these cases the ONS reading entails the OWS reading, but not vice versa (assuming a finite $n > 0$).

(14) Infinitely many dots are contained in all but at most $n$ circles.

(15) At least $n$ circles contain all but finitely many dots.

Similarly, consider the following examples.

(16) Infinitely many dots are contained in circle 1 or [circles 2 and 3].

(17) Circle 1 and [circles 2 or 3] contain all but finitely many dots.

We assume that the object of sentence (16) and the subject of sentence (17) denote the following quantifiers respectively, for three different circles $c_1, c_2$ and $c_3$.

$$\{ A \subseteq E : c_1 \in A \vee (c_1 \in A \wedge c_2 \in A) \}$$

$$\{ A \subseteq E : c_1 \in A \wedge (c_1 \in A \vee c_2 \in A) \}$$

Also the quantifiers in these sentences satisfy clause (ii) of (12), hence the scope dominance, but the two quantifiers in each sentence are not independent.

**4 Concluding remarks**

In this paper we have characterized scope dominance and independence for upward monotone quantifiers over countable domains. This is a natural extension of the results by Westerståhl and Zimmermann about self-commuting and scopeless quantifiers. This characterization directly extends a previous result by Altman et al. (2001), which concentrated on a subclass of quantifiers on countable domains, called finitely based quantifiers. Our results are still partial in some obvious respects. First, we did not characterize scope dominance for uncountable domains. Theorem 5 does not hold for such domains, for a similar reason to the reason that Fact 2 about finite domains does not hold for countable domains. Consider for instance the following sentence and quantifiers, parallel to (11) above over countable domains.
Uncountably many dots are contained in at least one of the countably many circles.

\[ Q_1 = \{ A \subseteq E : |D \cap A| = r_1 \} \]

\[ Q_2 = \{ A \subseteq E : C \cap A \neq \emptyset \}, \text{ where } |C| = r_1 \]

The quantifier \( Q_1 \) is dominant over \( Q_2 \), but these quantifiers do not satisfy the conditions of Theorem 5. Thus, a further generalization of our result is called for.

It is also natural to look for a characterization of dominance with non-upward monotone quantifiers. One recent result in this area is the characterization in Ben-Avi and Winter (2004) of scope dominance with downward monotone quantifiers over finite domains. One can also add further requirements on the relation \( R \) in (5), and obtain more quantifiers \( Q_1 \) and \( Q_2 \) that exhibit scope dominance for this restricted class of relations. Such more refined characterizations are relevant for natural language, where there are often logical restrictions on the possible denotations of binary relations. For instance, in the sentence every priest is taller than some peasant, where the relation be taller than is transitive, the ONS reading and the OWS reading are equivalent over finite domains, in contrast to the case with general \( R \)’s.

Characterizations of scope independence are useful for reducing ambiguity in computational representations of natural language sentences. One system that goes in this direction, using the results that were obtained in the present paper, is described in Altman and Winter (2003). Another system with the same motivations, based on slightly different formal assumptions, is described in Chaves (2003).

Acknowledgements

The first and third authors were partly supported by grant no. 1999210 ("Extensions and Implementations of Natural Logic") from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel. The third author is grateful for the UU OTS of Utrecht University, where part of this research was conducted. We are indebted to two anonymous JLLI referees for their useful remarks on a previous draft.

References


Continuations in Natural Language

[Extended Abstract]

Chris Barker
University of California, San Diego
9500 Gillman Drive
La Jolla, CA, USA
barker@ucsd.edu

ABSTRACT

Computer scientists, logicians and functional programmers have studied continuations in laboratory settings for years. As a result of that work, continuations are now accepted as an indispensable tool for reasoning about control, order of evaluation, classical versus intuitionistic proof, and more. But all of the applications just mentioned concern artificial languages; what about natural languages, the languages spoken by humans in their daily life? Do natural languages get by without any of the marvelous control operators provided by continuations, or can we find continuations in the wild? This paper argues yes: that an adequate and complete analysis of natural language must recognize and rely on continuations. In support of this claim, I identify four independent linguistic phenomena for which a simple CPS-based description provides an insightful analysis.

1. INTRODUCTION

The applications of continuations to date are remarkably diverse. As representative examples, I will mention three strands of research here: first, continuations provide an order-independent way of describing evaluation order in formal languages. For instance, Plotkin [10] shows how various Continuation-Passing Style (CPS) transforms can model evaluation disciplines such as call-by-name or call-by-value. Second, Griffin [5], Murthy [9] and others show that continuations are intimately involved in characterizing the computational content of classical (as opposed to intuitionistic) proofs. Third, continuations provide a useful tool for programmers who want to use powerful control operators such as call/cc in Scheme (and its analogs in other functional programming languages); conversely, Queinnec [11] shows how to use continuations in order to write in a direct style and still be assured that programs will behave in a reasonable way when it is the program’s users who have access to powerful control operators (such as the ‘back’ button on a web browser).

As diverse as these applications are, they all involve the design and analysis of artificial languages (i.e., programming languages and logical languages), as opposed to natural languages (the sort of languages that humans characteristically use when communicating with each other). We might wonder, then, whether continuations are relevant for the study of natural languages. This is the master question addressed by this paper:

- [Relevance] Are there phenomena in natural language that can profitably be analyzed using continuations?

Obviously, I believe that the answer is “yes”! I will try to persuade you to believe likewise by presenting four case studies, each of which has an analysis in terms of continuations.

An affirmative answer to the master question of relevance leads to a number of subquestions, including the following.

- [Universality] Even if some natural languages make use of continuation, do all languages make use of continuations?

Languages differ in many ways. For instance, all languages make a distinction between nouns and verbs, but not all languages make a distinction between singular nouns and plural nouns. I have drawn the case studies below exclusively from English, since that simplifies the exposition, but we must remain alert to the possibility that English may be unusual or even unique in the relevant respects. In fact, however, all of the phenomena discussed below have close analogs in many other languages. Coordination in particular is widespread: as far as I know, every language provides some way of saying something equivalent to John saw Mary and Tom, where the coordinating conjunction and combines two noun phrases (Mary and Tom) into a single complex noun phrase Mary and Tom. If an adequate analysis of coordination depends on continuations, as I argue below, that would strongly suggest that every natural language makes essential use of continuations.

- [Distribution] Which specific control operators do natural languages make use of? Which control operators are more ‘natural’?

A variety of control operators that make use of continuations have been proposed over the years, some of which differ in fairly subtle ways: call/cc versus control (C), shift and reset versus fcontrol and run, and so on. Sometimes one...
operator can be expressed in terms of another; nevertheless, if one or the other version is more prevalent in natural language, that might suggest that some operators are more ‘natural’ than others. In the discussion below, I will argue that $C$ and $f\text{control}/run$ are remarkably well-suited to handling quantification and focus.

- [Delimitation] Does natural language prefer delimited or undelimited continuations?

Historically, the first continuation operators were undelimited (e.g., call/cc or $J$). Fellessen [4, 3] proposed delimited continuations (sometimes called ‘composable’ continuations) such as control (‘$C$’) and prompt (‘%’). Interestingly, the natural-language phenomena discussed in this paper all seem to require some form of delimitation.

2. PRELIMINARIES

The intrepid reader is welcome to skip directly to the case studies, backtracking to this section only if some assumption or technique seems puzzling.

Theoretical computer scientists think deeply about the nature of formal languages. Many of the tools and techniques that are relevant for the study and the design of formal languages apply also to the study of natural language, including lexical analysis, parsing, and denotational semantics. As a result, computer scientists have highly developed intuitions about languages and about good ways to analyze them. These intuitions will go a long way towards understanding the issues and phenomena discussed below; after all, one of my main goals in this paper is to emphasize similarities between formal languages and natural languages.

Nevertheless, there are significant differences between computer science approaches and linguistic approaches. I have made some effort to present a discussion of natural language in a way that makes sense to a computer scientist—but only up to a point. To the extent that the assumptions and techniques agree with standard practice in computer science, well and good; but when there are differences, I will depend on the reader to trust that the linguistic assumptions are coherent and well motivated, and to keep an open mind about what is the “right” way to model language.

2.1 Declarative sentences denote truth values

As a starting point, it is necessary to think of natural language utterances as expressing computations. Natural language is clearly capable of specifying algorithms: just think of a recipe in a cookbook. Natural language can also express imperatives (Shut the door!), queries, and can even impinge on the real world in a direct way that formal languages typically do not (I hereby declare you husband and wife).

In this paper, however, I will concentrate exclusively on relatively simple and mundane declarative sentences such as Everyone left. Declarative sentences will have a type equivalent to a boolean, and their values will be either true or false. To see how this makes sense, it may help to consider the role of everyone left in the conditional sentence If (everyone left), then shut the door.

A little thought will reveal that treating declarative sentences as denoting truth values is inadequate in general. After all, there are more than two possible meanings for sentences. Even if Everyone left and The room is empty are both true, they have different meanings. But the same operation applies for formal languages: ‘$x = x$’ means something very different from ‘$x = 3$’, yet it still makes sense to treat them both as boolean expressions.

In some more elaborate linguistic treatments, sentences denote functions from times and worlds to truth values, with an analogous shift for expressions of other types. In the parlance of linguistics, a treatment in terms of truth values is ‘extensional’, and a system with times and worlds is ‘intensional’. Shan [13] shows that intensionalization can be rendered as a monad. Intensionality is not crucial in any of the discussions below, and the types will be complex enough anyway, so I will use an extensional semantics on which sentences denote truth values.

2.2 Methodology: formal grammar fragments

It is standard in mainstream linguistics when analyzing a particular type of expression to provide a formal grammar approximating the syntax and the semantics of the expressions under study, and that is the approach that I will take in this paper. Although I intend the grammar fragment developed below to capture something genuine about the nature of the natural language expressions, it is important to bear in mind that any formal grammar is at best an approximation of natural language, i.e., a hypothesis, not a definition, and the degree to which the formal treatment accurately reflects the behavior of the natural language expression will always be an empirical issue.

In the fragment, each expression will be assigned membership in some syntactic category, and the name of each syntactic category will also serve to indicate the semantic type of the meanings of expressions in that category.

A concrete example will help make this clear. Here is a direct-style starting point that the fragment below will build on:

(1) Some direct-style lexical entries:

<table>
<thead>
<tr>
<th>Syntactic category</th>
<th>Expression (= type)</th>
<th>Semantic value</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>e</td>
<td>j</td>
</tr>
<tr>
<td>Mary</td>
<td>e</td>
<td>m</td>
</tr>
<tr>
<td>left</td>
<td>e→t</td>
<td>left</td>
</tr>
<tr>
<td>saw</td>
<td>e→(e→t)</td>
<td>saw</td>
</tr>
</tbody>
</table>

Intuitively, $e$ is the category of expressions that denote individuals. Since the semantic job of the proper name John is to refer to a particular individual (in this case, the individual $j$), the word John is a member of the category $e$. Expressions in a category with a complex label of the form $A\rightarrow B$ will denote a function from meanings of type $A$ to meanings of type $B$. Expressions in category $t$ denote truth values (booleans), so that an intransitive verb such as left has type $e→t$ and denotes a function from individuals to truth values.

Syntactic combination in this direct grammar will always correspond semantically to functional application.

(2) $f:x:B ::= x:A \quad f:A\rightarrow B$

This way of specifying syntactic combination is a kind of BNF notation enhanced in two ways: categories have been labeled with semantic values, and the syntactic categories contain variables over types. The idea is that (2) schematizes over a set of valid BNF rules for any choice of categories $A$ and $B$. For instance, (2) licenses combining the noun phrase John with the intransitive verb phrase left (in
that order) by virtue of the following instantiations: \( A = e, \ B = t, \ f = \text{left}, \) and \( x = j. \) This gives rise to the following syntactic and semantic analysis for the declarative sentence *John left.*

\[
\begin{array}{c}
\text{left} \ j \ t \\
\text{left} \ e \ t \\
\text{John} \ v \ t \ j
\end{array}
\]

The intended interpretation is that a use of the sentence *John left* will be predicted to be true just in case the function denoted by \( \text{left} \) maps the individual referred to by *John* onto the truth value *true*.

Unlike most formal languages, which tend to prefer a uniform direction of functional application, English allows some arguments to precede the function that applies to them, as in the example immediately above, and some to follow. For instance, in the sentence *John saw Mary*, the transitive verb *saw* combines first with *Mary* on its right, then the verb phrase *saw Mary* combines with the argument *John* to its left. Thus we need a second rule for syntactic combination to allow for arguments that appear on the right:

\[(3) \ f : x : B \iff f : A \rightarrow B \quad x : A\]

One of the ways in which the formal grammar developed here is unrealistic is that the syntactic categories of *left* and *saw* as given in (1) do not specify which arguments follow and which precede. Thus in addition to deriving *John left* and *John saw Mary*, it also correctly derives the ungrammatical sentences *\( \star \) Left John* and *\( \star \) Saw John Mary* (the star indicates that the string it is prefixed to is not well-formed).

It is not difficult to build grammars that take linear order into account; but since order will not play an important role in what follows, I have opted to ignore linear order in an effort to keep the correspondence between syntactic category and semantic type as transparent as possible: since the syntactic category of *saw* is *e \rightarrow (e \rightarrow t)*, it is clear that the semantic type of its value will be a function from individuals to functions from individuals to truth values.

### 2.3 Strategy: case studies

My strategy will be to discuss a number of cases in which a continuation-based analysis is elegant and potentially insightful. I will discuss several different cases rather than developing a single analysis in more depth for two reasons: first, because which solutions strike a responsive chord varies from one person to another; and second, for the sheer joy of considering a number of different natural language phenomena.

There will be four case studies: quantification, coordination, focus particles, and misplaced modifiers. (I will introduce each one of these phenomena using concrete examples below, of course.) The treatments of quantification and coordination have been developed in some detail in [1], but the proposals for focus particles and misplaced modifiers are new to this paper. There are a few other continuation-based analyses of natural language already in the literature: Shan [15] proposes a continuation-based analysis of question formation, and Shan and Barker [17] discuss a continuation approach to the phenomena of weak crossover and superiority, which unfortunately requires background discussions that are too complex to attempt here. In addition, at this conference, Shan will present a new continuation-based analysis of negative polarity.

### 3. CASE STUDY: QUANTIFICATION

The most carefully worked out natural-language application of continuations to date concerns quantification ([2], [1], [17]). According to the simple analysis above in section 2, the sentence *John left* denotes the truth value \( \text{left} \) and does not involve quantification; a sentence like *Everyone left*, however, is quantification, and means roughly \( \forall x \text{left} \). (We say that the symbol \( \forall \) is a quantifier, and likewise \( \exists \) below.)

The main problem of interest is what I call ‘scope displacement’: even when a quantification expression such as *everyone* occurs in a deeply embedded position, the quantifier it contributes can take semantic scope over the entire sentence.

\[(4) \ a. \ \text{John (saw everyone)}.
\]

\[(b. \ \forall x \text{saw } x \text{ j} \ \text{.}\]

Here, even though *everyone* is buried within the verb phrase *saw everyone*, in the paraphrase in (4b), the logical quantifier \( \forall \) takes scope over the entire meaning. Continuations, of course, are superb at allowing a deeply embedded operator to take control over a larger expression in which it is embedded, and the analysis described here is based on that ability.

### 3.1 From quantification to continuization

We proceed by deducing a type for quantificational expressions like *everyone, no one, or someone*. Syntactically, these quantificational expressions can be substituted in any position occupied by a proper name. Thus since *John saw Mary* is grammatical, the sentences *Everyone saw Mary, John saw everyone, Someone saw everyone, etc.*, will all be grammatical as well. Yet it is problematic assigning *everyone* type *e*, the same type as *John or Mary*, since there is no particular individual who has all and only the properties that are true of *everyone*. The inadequacy of supposing that quantificational noun phrases denote individuals is even more stark in the case of *no one*: if *No one left* is true, which individual has the property of leaving?

We can solve this dilemma by considering the parallelism in the following two syntactic analyses:

\[1\] I use expressions in the first-order predicate calculus to specify what I have in mind for the meaning. At one level, this suggests that natural language can be translated into some suitable logical language, and this is in fact a perfectly legitimate technique. At a deeper level of analysis, however, I am assuming for the purposes of this paper that words are logical constants denoting some specific individual, truth value, or function, and that the denotations of complex phrases are composed from the denotations of their constituent words entirely through functional application as governed by the various syntactic combination rules as given. In Montague’s [8] terms, the intermediate logical description language is non-essential, and could be dispensed with entirely if desired.

\[2\] The parentheses in (4a) indicate what I take to be a constituent. There is abundant evidence that the transitive verb *saw* forms a constituent with its direct object *everyone* to the exclusion of the subject *John*. To give just one argument, note that verb phrases can be coordinated (see section 5): *John (saw everyone) and (called home)*.
If John has a type that combines with the type of left to form a complex expression of type $t$, then presumably everyone must also have a type that combines with the type of left to produce an expression of type $t$. Since we want to avoid giving everyone the type $e$, the next simplest type that will serve is $(e\to t)\to t$. This is in fact the main insight proposed by Montague [8]: that noun phrases such as everyone and no one can have the type $(e\to t)\to t$, which he called a generalized quantifier (for reasons I won’t discuss here). Intuitively, what this analysis says is that everyone is a function mapping a verb phrase to a truth value.

This naive solution works out fine for the example Everyone left (although it has the rather disturbing effect of reversing the direction of functional application, a point I’ll return to shortly). But if we attempt to place a quantification expression in some other position where a proper name can occur (i.e., instead of John saw Mary we have John saw everyone, as in the diagram on the right of (6)), the solution breaks down: the generalized quantifier type $(e\to t)\to t$ does not suffice, and we seem to need a yet higher type.

This is the point at which continuations enter the picture. Consider again the diagrams in (5) from the point of view of continuations. What is the continuation of the expression John relative to the sentence John left? It will be a function mapping individuals into truth values, i.e., a function of type $e\to t$. Not coincidentally, this happens to also be the type of the verb phrase left. From this perspective, the proposed generalized quantifier type $(e\to t)\to t$ is a function from its continuation (type $e\to t$) to a truth value.

Now we can understand better how to deal with the situation depicted in (6). What is the continuation of everyone in the sentence John saw everyone? Well, once again it will be a function mapping individuals to truth values. More specifically, it will be the function mapping each individual $x$ to true just in case John saw $x$. If only we had a way of providing the denotation of everyone with access to its continuation, a single type would serve in all of the examples considered so far.

In operational terms, we need the control operator $\lambda$: (7) $E[C[M]] \triangleright M(\lambda x.E[x])$

What (7) says is that when $CM$ occurs in an evaluation context $E$, evaluation proceeds by packaging the context as a continuation (i.e., $\lambda x.E[x]$), then replacing the entire computation with $M$ applied to the continuation.

Then if everyone denotes $C(\lambda P.\lambda x.Px)$, John saw everyone will denote $(\lambda P.\lambda x.Px)(\lambda x.saw\ x\ j)$, which is equivalent via $\beta$-reduction to $\lambda x.saw\ x\ j$, as desired.

My denotational implementation of this analysis will be based on the following CPS transform:

(8) a. $\pi = \lambda K.K\alpha$ (for any constant $\alpha$)  
    b. $\bar{MN} = \lambda K.kN(\lambda n.(\lambda m.(\lambda n.(\lambda m.n(mn)))))$

I prove a simulation result for this transform in [1].

Note that the transform does not provide a rule for expressions of the form $\lambda x.M$. It is possible to omit such a rule here because the natural language constructions considered in this paper involve only functional application, and never lambda abstraction.

As a consequence, the type of the CPS transform of an expression of any direct type $X$ will be $(X\to \sigma)\to \sigma$, where $\sigma$ is some answer type. In particular, the type of the CPS transform of a direct expression of type $A\to B$ will simply be $((A\to B)\to \sigma)\to \sigma$, rather than the more traditional (e.g., [7]) type $\lambda X.((B'\to \sigma)\to \sigma)$, where $X$ is the type of the transform of an expression of type $X$. This difference in types accounts for the expression ‘$k(mn)$’ in (8b) rather than the traditional ‘mn$\kappa$’. I find this typing much simpler; however, it seems to be wrenching for people used to the standard transform, so this is one of the places where I must ask the reader to keep an open mind about the right way to do things.

The CPS transform of the lexical items in (1) all depend on the rule governing constants in (8a):

(9) CPS transforms of the lexical entries in (1):

<table>
<thead>
<tr>
<th>Lexical Item</th>
<th>CPS Transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>$(e\to A)\to A$ $\lambda\kappa.j$</td>
</tr>
<tr>
<td>Mary</td>
<td>$(e\to A)\to A$ $\lambda\kappa.m$</td>
</tr>
<tr>
<td>left</td>
<td>$(e\to e\to t)\to A$ $\lambda\kappa.left$</td>
</tr>
<tr>
<td>saw</td>
<td>$(e\to(e\to t))\to A$ $\lambda\kappa.saw$</td>
</tr>
</tbody>
</table>

Thus for the purposes of the transform, left counts as a constant, even though its direct denotation left is a function of type $e\to t$.

Here A is a variable over syntactic categories (in effect, over types). For the sake of quantification alone, we could replace the type variables here with $t$. The more general rules will be needed, however, when we extend the fragment to handle focus in the next section.

The syntactic combination rules for the CPS grammar will be as follows:

(10) $\lambda\kappa.M(\lambda n.(\lambda m.\overline{M}(\lambda n.(\lambda m.(\lambda n.(\lambda m.n(mn)))))): (B\to E)\to D$  
    ::= $\overline{M}((A\to B)\to C)\to D$ $\overline{N}(A\to E)\to C$

(11) $\lambda\kappa.(\lambda n.\overline{M}(\lambda n.(\lambda m.\overline{M}(\lambda n.(\lambda m.n(mn)))))): (B\to C)\to E$  
    ::= $\overline{N}(A\to D)\to E$ $\overline{N}(A\to B)\to C\to D$

These syntactic combination rules just make explicit how to syntactically and semantically combine expressions that have already undergone the CPS transform. Thus (10) is the CPS analog of (3), and (11) is the analog of (2).

Some of the complexity in (10) and (11) could be reduced if we chose $C = D = E$. This would suffice for present purposes except for the treatment of focus in the next section, which depends on allowing a control operator to return an answer that is not of type $t$.

3It is an interesting question whether this abstraction-free approach can be maintained in a more complete analysis of natural language. One strategy is to use continuations instead of lambda abstraction whenever natural language seems to require lambda abstraction; see, e.g., [17] for a continuation-based treatment of relative clause formation.
(A→B)→C \hspace{1cm} E
\hline
A→B \hspace{1cm} 2 \hspace{1cm} A \hspace{1cm} 3
\hline
B \hspace{1cm} E
\hline
E \hspace{1cm} E
\hline
A→E \hspace{1cm} I \hspace{1cm} 3
\hline
C \hspace{1cm} E
\hline
(A→B)→C \hspace{1cm} I \hspace{1cm} 2
\hline
D \hspace{1cm} E
\hline
(B→E)→D \hspace{1cm} I \hspace{1cm} 1
\hline
\end{array}
\vspace{0.5cm}

Figure 1: Derivation of (10): a natural-deduction proof using only → Elimination and → Introduction that \((A→B)→C)→D, (A→E)→C ⊢ (B→E)→D.\)

The proof in figure 1 may help untangle the types. The semantic values in (10) (and hence the application rule in the CPS transform given in (8b)) are just the Curry-Howard labelling of the proof in figure 1. An analogous derivation is available for (11), but is not included here.

These rules give us the following derivation:

\[
\begin{array}{c}
(t→t)→t \\
\hline
(e→t)→t \hspace{1cm} ((e→t)→t)→t \\
\hline
John \left \Rightarrow \left \Rightarrow \hspace{1cm} \text{left} \hspace{1cm} \text{left} \\
\end{array}
\]

\[
[\text{John left}] = \lambda\kappa.[\lambda\kappa.\text{left}](\lambda f.[\lambda\kappa.\kappa](\lambda x.\kappa(fx))) \Rightarrow \kappa.\kappa(\text{left } j)
\]

which, after application to the trivial continuation \(\lambda x.x\), yields the same denotation provided by the direct grammar, namely, \(\text{left } j\).

At this point we can add lexical entries for some quantificational noun phrases that have no direct counterpart.

(12) everyone \hspace{1cm} (e→t)→t \hspace{1cm} \lambda\kappa.\forall x.kx

someone \hspace{1cm} (e→t)→t \hspace{1cm} \lambda\kappa.\exists x.kx

Since everyone has the same type as the CPS transform of John, we now have

\[
[\text{Everyone left}] = \lambda\kappa.[\lambda\kappa.\text{left}](\lambda f.[\lambda\kappa.\forall x.kx](\lambda x.\kappa(fx))) \Rightarrow \kappa.\forall x.\kappa(\text{left } x),
\]

as well as

\[
[\text{John saw everyone}] \Rightarrow \lambda\kappa.\forall x.\kappa(\text{saw } x j)
\]

which yield \(\forall x.\text{left } x\) and \(\forall x.\text{saw } x j\) after application to the trivial continuation, as desired.

Note that continuization has restored the natural direction of functional application (which had been reversed by the first strategy depicted in (6)). That is, not only is the direction of (the CPS analog of) functional application in John left-right-to-left, the direction of application in the continuized grammar for handling Everyone left is also right-to-left.

3.2 Quantificational ambiguity

The transform as specified in (8b) imposes a left-to-right evaluation discipline. But we might just as well have used instead

(13) \(\overline{MN} = \lambda\kappa.\overline{N}(\lambda n.\overline{M}(\lambda n.\kappa(mn)))\)

in which the right-hand component, \(\overline{N}\), is evaluated first. For sentences containing only one quantificational noun phrase, order of evaluation makes no difference. But for sentences containing two or more quantificational noun phrases, a difference in order of evaluation corresponds to a difference in the predicted meaning:

a. Someone saw everyone.

b. \(\exists x.\forall y.\text{saw } x y\)

c. \(\forall y.\exists x.\text{saw } x y\)

Given the availability of both evaluation orders, the sentence in (14a) containing two quantificational noun phrases will have meanings that (after application to the trivial continuation) reduce to either (14b) (for left-to-right evaluation) or (14c) (for right-to-left evaluation).

The consensus in the field is that sentences like (14a) are in fact ambiguous in exactly the way considered here. The interpretation in (14b) corresponds to a situation in which there is some particular person who has the property of seeing everyone, and the interpretation in (14c) corresponds to a situation in which there is a potentially different see-er who saw each person.

Quantifier scope ambiguity has been studied in considerable detail. For more discussion of the predictions of continuation-based analyses with respect to this type of ambiguity, see [1], [2] and [17].

4. CASE STUDY: FOCUS

In this section I propose that a phenomenon called ‘focus’, along with so-called focus particles such as only, behave like Sitaram’s [18, 19] run and fcontrol. One point of interest is that these control operators rely on delimited continuations. Delimited continuations have received a considerable amount of attention recently, and in fact are the topic of at least three papers at the workshop. I believe that they play an important role in natural language as well. In fact, I would be (mildly) surprised to find a natural language operator that behaved exclusively in an undelimited manner.

4.1 fcontrol and run

There are many control operators that rely on delimited continuations, including Fellesien’s control and prompt, shift and reset, and others. I will discuss Sitaram’s fcontrol and run here for the simple reason that they provide exactly the functionality required for describing the behavior of focus particles such as only.

Roughly, fcontrol provides a way to throw a signal, and run catches and handles signals thrown by fcontrol.

The fcontrol operator takes one argument, and throws two values: the argument to fcontrol and the prefix of the continuation of the fcontrol expression up to the closest enclosing run.

The run operator takes two arguments: an expression possibly containing an occurrence of fcontrol, and a “handler” function that processes the two arguments thrown by fcontrol.

For instance, in

\[
\begin{array}{c}
(+ 1 \text{ (run } (+ 2 (\text{fcontrol } 3) 4))) \\
\hline
\text{ (lambda } x k ((k (k x))))
\end{array}
\]

The invocation of fcontrol throws the two values 3 and a (truncated) continuation \(\kappa\). Since \(\kappa\) is the continuation of the expression “(fcontrol 3)” relative to the larger expression only up to the closest occurrence of run, \(\kappa\) will be the function \(\lambda y.((+ y 4))\). The handler procedure will bind \(x\) to 3 and \(k\) to \(\kappa\), and the result of the entire computation will be 37.
The continuation caught by the \textit{fcontrol} is delimited, in the sense that it does not contain any material outside the scope of the enclosing \textit{run}; in particular, the function corresponding to the continuation does not contain the increment operation (there is no “+ 1” inside of \textit{f}). In this example, since the continuation is just a function, it can be called several times without replacing the current evaluation context. This allows expressions like $(k (k x))$ as in the example at hand, in which one occurrence of a continuation takes an expression involving another occurrence of the same continuation as an argument. This is the sense in which delimited continuations are ‘composable’.

Before we can make use of \textit{run} and \textit{fcontrol}, we must consider the meaning of sentences containing focus marking and focus particles.

### 4.2 Focus

Most (probably all) languages provide some way of marking some constituent in a sentence as having extra prominence. In spoken English, this is typically accomplished in part by a local maximum in the fundamental frequency (the lowest frequency at which the vocal folds are vibrating). By convention, the location of such a ‘pitch accent’ is indicated typographically by setting the most affected word in capital letters:

(15)  a. JOHN saw Mary.
    b. John SAW Mary.
    c. John saw MARY.

There is a distinct but elusive difference in meaning among these sentences that depends on the location of the pitch accent. In each case, it remains true that John saw Mary, but which piece of information is being emphasized differs. In traditional terms, the constituent containing the pitch accents is said to be ‘in focus’, which means (very roughly) that it carries the new information provided by the sentence. These observations can be sharpened by noting that the location of the pitch accent correlates with the precise piece of information requested by a question.

(16)  a. Who saw Mary?
    b. What did John do to Mary?
    c. Who did John see?

Thus (15a) can be a suitable answer only to the question in (16a), and not to either (16b) or (16c), and similarly for the other answer/question pairs.

The semantic effect of the location of pitch accent becomes even more tangible in the presence of a focus particle such as \textit{only}, \textit{also}, or \textit{too}.

(17)  a. John only drinks PERrier.
    b. John only DRINKS Perrier.

We say that \textit{only} ‘associates’ with whatever element is in focus. With a pitch accent on the first syllable of \textit{Perrier}, then, \textit{Perrier} will be in focus, and (17a) conveys at least the information paraphrased in (18):

(18)  a. John drinks Perrier.
    b. There is nothing else that John drinks other than Perrier.

The particle \textit{only} then, picks out some element in the situation and contrasts it with other possible alternative choices: John doesn’t drink whiskey, he doesn’t drink milk, he only drinks Perrier.

With a pitch accent on the verb \textit{drinks} in (17b), however, the appropriate paraphrases differ:

    b. There is nothing else that John does with Perrier other than drink it.

It remains true that John drinks Perrier, but now the element of the situation that \textit{only} puts into contrast with other alternatives is the activity of drinking. According to (17b), all John does with Perrier is drink it: he doesn’t sell it, he doesn’t photograph it, he doesn’t bathe in it.

Note that the conditions under which (17a) and (17b) will be true are mutually distinct: (17a) can be true even if John sometimes bathes in Perrier, and (17b) can be true even if John sometimes drinks whiskey. Thus in the presence of \textit{only}, the location of the pitch accent can determine whether a sentence is true or false.

I will treat pitch accent in a sentence as if it were an operator \textit{F} immediately preceding the constituent that is in focus (i.e., that bears the pitch accent). I will continue to use capitals to help guide pronunciation.

(20)  a. John only drinks \textit{F} (PERrier).
    b. John only \textit{F} (DRINKS) Perrier.

Now we’re ready to attempt an analysis in terms of \textit{run} and \textit{fcontrol}. The key is to ask the following question: given pitch accent on \textit{Perrier}, what precisely is the relation that holds between John and Perrier and nothing else?

(21)  a. John only drinks \textit{F} (PERrier).
    b. \textit{λxy}. \textit{drink} \textit{x} \textit{y}

The answer is the relation that holds between John and some object \textit{x} if John drinks \textit{x}. In other words, (21b) is a continuation, namely, the continuation of the focussed expression delimited by the enclosing \textit{only}.

Considering next the contrasting example with pitch instead on \textit{drinks}, what is the relation that holds between John and the activity of drinking and nothing else?

(22)  a. John only \textit{F} (DRINKS) Perrier.
    b. \textit{λxy}. \textit{Perrier} \textit{y}

The answer this time is the relation that holds between John and some activity \textit{x} if John does \textit{x} to Perrier. Once again, the desired relation is the continuation of the focussed word up to but not including \textit{only}.

Based on these examples, we can now guess that the semantic effect of pitch accent on a constituent is exactly \textit{fcontrol}.$^4$ So where I wrote ‘\textit{F}’ in (22), the meaning is \textit{fcontrol}. (In a happy coincidence, in this application we can construe the ‘f’ of \textit{fcontrol} as mnemonic for ‘focus’.) Similarly, the meaning of \textit{only} will invoke \textit{run}. It remains only to specify the handler routine that unpacks the information provided by the use of \textit{fcontrol}:

\[
[\textit{only} \textit{P}] = \textit{run} P \textit{λxy}. (\textit{and} (\textit{κxy}) (\forall z (\textit{or} (\textit{equal} \textit{x} \textit{z}) (\textit{not} (\textit{κzy})))))
\]

This denotation gives rise to the following analyses:

(23) John only drinks \textit{F} (PERrier).
    (\textit{and} (\textit{drinks} \textit{Perrier} \textit{j}))
    (\forall z (\textit{or} (\textit{equal} \textit{Perrier} \textit{z})
    (\textit{not} (\textit{drinks} \textit{z} \textit{j}))))

(24) John only \textit{F} (DRINKS) Perrier.
    (\textit{and} (\textit{drinks} \textit{Perrier} \textit{j}))
    (\forall z (\textit{or} (\textit{equal} \textit{drinks} \textit{z})
    (\textit{not} (z \textit{Perrier} \textit{j}))))

$^4$Chung-chieh Shan first noticed the similarity between my analysis of focus and Sitaram’s operators.
These meanings are equivalent to the paraphrases we started with in (18) and (19).

Now, it only makes sense to bother building continuations if the meaning captured by the continuation can be arbitrarily complex. The examples we have discussed so far have been as simple as possible, so it is worth considering examples in which the delimited continuation is more complicated.

(25) Mary only tried to dance with F(JOHN).
\[ (\text{and (tried-to-dance-with j m)}) \]
\[ (\forall z (\text{or (equal) } z) \text{ (not (tried-to-dance-with } z \text{ m))))) \]

(26) Mary only tried to F(DANCE) with John.
\[ (\text{and (tried-to-dance-with j m)}) \]
\[ (\forall z (\text{or (equal dance z) (not (tried-to--z-with } j m)))) \]

The simple denotational fragment in section 8 does not attempt to reconstruct the full power of run and fcontrol, but it does handle the examples discussed here.\(^5\)

5. CASE STUDY: COORDINATION

One of the distinctive features of natural language is the pervasive use of polymorphic coordination.

(27) a. [John left] and [Mary left]. t
b. John [left] and [slept]. e→t
c. John [saw] and [remembered] Mary. e→(e→t)
d. [John] and [Mary] left. e

The types in the right column of (27) correspond to the (direct, pre-CPS) type of the bracketed expressions coordinated by and. In (27a), two clauses (type t) coordinate to form a complex sentence; in (27b), two verb phrases (type e→t) coordinate to form a complex verb phrase; and so on.

What about quantificational noun phrases? They also can coordinate:

(28) a. Someone or everyone left.
   b. John or everyone left.

Even more interesting, they can more or less freely coordinate with proper names as in (28b), providing confirmation that proper names and quantificational noun phrases are syntactically interchangeable, as predicted by the analysis in section 3.

We can arrive at a continuation-based analysis of coordination if we consider that (27d) means the same thing as John left and Mary left.\(^6\) The continuation of the coordinated phrase is λx.left x; what the conjunction does is take this continuation and distribute it across each of the conjuncts. The resulting analysis of conjunction involves adding two new rules for syntactic combination:

(29) \(\lambda x.\text{and}((\text{R})(\text{L}))\): \(A \quad := \quad \text{L}:A \quad \text{“and”} \quad \text{R}:A\)

(30) \(\lambda x.\text{or}((\text{R})(\text{L}))\): \(A \quad := \quad \text{L}:A \quad \text{“or”} \quad \text{R}:A\)

These rules differ from one another only in substitution of or for and.

The syntactic parts of these rules simply say that anywhere that an expression of type A can occur, an expression of the form “A₁ and A₂” or “A₁ or A₂” can also occur, as long as A₁ and A₂ are themselves expressions of category A.

What the semantic parts of the rules say is: whatever you were planning on doing to the value provided by the expression in this position, first do it to the value of the left conjunct, then do it to the value of the right conjunct, and conjoin the two results. This schema guarantees the following example paraphrases:

(31) a. John left and John slept.
   b. John and Mary left.
   c. John or everyone left.

One sign of the utility of coordination is that in Wall Street Journal text, and is the second most commonly used word (first place goes to the). If suitably constrained with syntactic marking (such as parentheses) overtly marking the syntactic strings involved, coordination as a control operator could provide a rather appealing programming device. Imagine being able to write if (x == (2 or 3)) then ... and have it mean if (x == 2) or (x == 3) then ... ?

6. CASE STUDY: MISPlACED MODIFIERS

As a final example of a natural language construction that might profit from a continuation-based analysis, consider the following data:

(32) a. An occasional sailor walked by.
   b. John drank a quiet cup of tea.

The modifier occasional is misplaced: \(^8\) there is no specific sailor (nor even any set of sailors) that has the property of being occasional; rather, it is the event of a sailor walking by that happens occasionally compared with other relevant events. Similarly, in (32b), it is not the cup of tea that is quiet in the relevant sense, but the activity of drinking the tea.

Once again, we have an embedded element that needs to take semantic force at a higher level, and once again we can allow that expression to take control by providing it with access to its continuation. (Chris Potts (personal communication) first suggested using continuations to analyze misplaced modifiers, though he shouldn’t be held responsible for the shortcomings of my treatment here.) Assuming we know what the adverb occasionally means, we can give the misplaced adjective occasional the following denotation: \(\lambda x.\text{occasionally}(s \cdot x \cdot x)\), and similarly for quietly plays only the kind of behavior characterized by the rule given in (30).

\(^8\) Hayo Thielecke points out that J, an APL-like language whose web site (http://www.jsoftware.com) claims that it uses “constructs and syntax which closely mirror those of natural language”, has a construction that is a special case of the operator imagined here. More specifically, under certain circumstances \( x (f g h) y \) evaluates as \( (x f y) g (x h y) \), which is isomorphic to example (27c).

\(^5\) There is an important dependence on the context of utterance that goes unrecognized in this analysis. If John only DRANK Perrier quantified over absolutely everything that John might have done with Perrier, then it would entail that he didn’t buy Perrier, that he didn’t swallow Perrier, that he didn’t taste Perrier, that he didn’t raise Perrier to his lips, that he didn’t do all kinds of things to Perrier that he must have done. The standard assumption is that the quantification over alternatives must be context-dependent, so that what a use of only really means is that there is no other contextually-relevant thing that John did to Perrier. How precisely to calculate what ought to count as a contextually-relevant activity is a problem for AI, and not for linguistics.

\(^6\) There are other kinds of coordination not treated here. For instance, John and Mary are a happy couple cannot be paraphrased as *John is a happy couple and Mary is a happy couple. Interestingly, unlike conjunction, disjunction discards
and quiet. The idea is to replace the misplaced adjective with the trivial adjective meaning $\lambda x.x$, and then let its adverbial counterpart take scope over the complete sentence.

This analysis gives the following equivalences:

(33) a. An occasional sailor walked by.
Occasionally, a sailor walked by.

b. John drank a quiet cup of tea.
Quietly, John drank a cup of tea.

Additional details (such as the type of adjectives and nouns) are given in the cumulative fragment below in section 8.

7. HISTORICAL NOTES

If natural language is rife with phenomena that beg for a continuation-based analysis, why hasn’t anyone noticed before? The answer, of course, is that some people have noticed. In fact, I would suggest that in addition to the many independent discoveries of continuations described by Reynolds [12], Richard Montague [8] also invented a technique that relies in a limited way on continuations.

Montague was a logician at UCLA who was one of the major figures in the early days of establishing formal approaches to natural language semantics. In 1970 he constructed the first popular formal analysis of quantification [8]. He did this by suggesting that quantificational noun phrases such as everyone differ from non-quantificational noun phrases such as John in just the way I proposed in section 3: quantificational noun phrases are in effect control operators whose meanings are properly expressed as functions on their own continuations.

The main limitation of Montague’s approach is that noun phrases were the only type of expression that had access to their continuations; nevertheless, with hindsight, anyone familiar with continuation-passing style programming will immediately recognize a primitive form of continuation passing in Montague’s formal grammars. Joe Goguen, my colleague at UCSD, was a colleague of Montague’s at UCLA during the 70’s, and he tells me that the connection between Montague’s techniques and continuations was noticed long ago, at least in the folklore, if not in the literature.

It has only been recently, however, that people have explored this connection systematically, providing a more general mechanism for accessing continuations in natural language analyses. In particular, Herman Hendriks’ 1993 dissertation [6] proposes a type-shifting system that in effect produces CPS transforms as needed (and may deserve to be counted as yet another independent invention of a continuation-based system). Philippe de Groote [2] provides an analysis of simple quantification based on the $\lambda\mu$-calculus, and a paper by me [1] explores a continuation-based approach to quantification in considerable detail. Since then, Chung-chieh Shan has proposed a number of continuation-based analyses of natural language phenomena ([14], [15], [16]), including his paper at this conference.

Incidentally, if natural languages do make abundant use of continuations, then Reynolds was doubly right to speak of the ‘discovery’ of continuations rather than their ‘invention’. It is intriguing to consider the possibility that that the presence of continuation-based control operators in the native language of the various discoverers may even have inspired their proposals at some subconscious level.

8. CUMULATIVE FRAGMENT WITH RESET

Figure 2 gives a CPS grammar describing a fragment of English with a context-free syntax and a denotational semantics. The fragment includes versions of the four analyses discussed above for quantification, focus, coordination, and misplaced modifiers.

In addition, the fragment provides a reset operator, associated here with the complementizer that (as discussed immediately below). This section will argue for all of the four phenomena that the relevant continuations must at least sometimes be delimited.

The meanings listed in the examples below are guaranteed to be equivalent to the denotations provided by the fragment up to $\beta$-reduction and application to the trivial continuation $\lambda x.x$.

For instance, here are some simple examples involving zero, one, and two quantificational noun phrases:

\[(34)\text{ a. John left. left }_j\]
\[\text{ b. Everyone left. } \forall x. \text{left }_x\]
\[\text{ c. John saw Mary. saw }_m j\]
\[\text{ d. John saw everyone. } \forall x. \text{saw }_x j\]
\[\text{ e. Someone saw everyone. } \exists x. \forall y. \text{saw } y x\]

Since the fragment does not include extra combination rules for right-to-left evaluation, there is only one interpretation for Someone saw everyone (see section 3.2 for discussion).

I now introduce embedded clauses and a reset operator.

\[(35)\text{ a. John claimed [Mary left]. claimed (left m) }_j\]
\[\text{ b. John claimed [everyone left]. } \forall x. \text{claimed (left x) }_j\]
\[\text{ c. John claimed that [everyone left]. claimed (forall x. left) }_j\]

Unlike saw, which denotes a relation between two individuals, claimed relates an individual and a proposition. Syntactically, claimed takes a clause as its first argument. Thus the bracketed strings in (35) are all complete clauses in their own right.

The interpretation in (35b) says that John made several different claims, one for each person. The interpretation in (35c) says that he made one single claim, a claim about everybody. Both interpretations seem to be valid.

In continuation terms, we need to be able to delimit the continuation for everyone so as to extend only as far as the material in the embedded clause. In order to experiment with delimitation, I will adopt the following expository strategy: as shown by comparing (35b) with (35c), claimed optionally allows its argument clause to be introduced by the complementizer that. By giving that the semantics of a reset operator, we can add or subtract delimitation at will and see what happens. As shown in (35c), for instance, the presence of the that delimits the embedded clause and produces the second legitimate interpretation.

The following set of sentences illustrates the analysis of focus.

\[\ldots\]
left \((e\rightarrow t)\rightarrow A\) \rightarrow A \ \lambda \kappa. \text{left}

saw \((e\rightarrow(e\rightarrow t))\rightarrow A\) \rightarrow A \ \lambda \kappa. \text{saw}

claimed \((t\rightarrow(e\rightarrow t))\rightarrow A\) \rightarrow A \ \lambda \kappa. \text{claimed}

John \((e\rightarrow t)\rightarrow t\) \ \lambda \kappa. \text{j}
Mary \((e\rightarrow t)\rightarrow t\) \ \lambda \kappa. \text{m}

everyone \((e\rightarrow t)\rightarrow t\) \ \lambda \kappa. \forall \kappa. \text{KE}

someone \((e\rightarrow t)\rightarrow t\) \ \lambda \kappa. \exists \kappa. \text{KE}

a \(((e\rightarrow t)\rightarrow e)\rightarrow t\) \ \lambda \kappa. \text{a}

sailor \(((e\rightarrow t)\rightarrow e)\rightarrow t\) \ \lambda \kappa. \text{sailor}

tall \(((e\rightarrow(t\rightarrow e))\rightarrow t)\rightarrow t\) \ \lambda \kappa. \text{tall}

occasional \(((e\rightarrow(t\rightarrow e))\rightarrow t)\rightarrow t\) \ \lambda \kappa. \text{occasionally}(\kappa(\lambda\kappa.x))

\[
\begin{align*}
\lambda \kappa. M(\lambda n. N(\lambda m. (m n))) : (B \rightarrow E) \rightarrow D & \quad \rightarrow M : ((A \rightarrow B) \rightarrow C) \rightarrow D \\
\lambda \kappa. N(\lambda n. M(\lambda m. (m n))) : (B \rightarrow C) \rightarrow E & \quad \rightarrow N : (A \rightarrow D) \rightarrow E \\
\lambda \kappa. \text{and}(\lambda \kappa. \text{R}) : \text{A} & \quad \rightarrow \lambda \kappa. \text{and} \ : \text{A} \\
\lambda \kappa. \text{or}(\lambda \kappa. \text{R}) : \text{A} & \quad \rightarrow \lambda \kappa. \text{or} \ : \text{A} \\
\lambda \kappa. (X, \kappa) : \text{A} \rightarrow ((A \rightarrow t) \times A) & \quad \rightarrow \lambda \kappa. (X, \kappa) : \text{A} \rightarrow t \\
\lambda \kappa. \text{only}(X, \kappa) : \text{A} \rightarrow t & \quad \rightarrow \text{only} \ : \text{A} \rightarrow ((B \rightarrow t) \times B) \\
\lambda \kappa. (X, \kappa) : (\lambda x. \text{A}) & \quad \rightarrow \text{only} \ : \text{A}
\end{align*}
\]

Figure 2: A grammar in continuation passing style covering most of the examples discussed in the text.

(36) a. John only saw F (MARY).
\text{only}(\lambda \kappa. \text{Saw}, \lambda x. \text{Saw } x \text{ j})

b. John only F (SAW) Mary.
\text{only}(\lambda \kappa. \text{Saw}, \lambda x. x \text{ m j})

c. John only F (SAW MARY).
\text{only}(\lambda \kappa. (\lambda \kappa. \text{Saw } m), \lambda x. x j)

d. John only claimed F (MARY LEFT).
\text{only}(\lambda \kappa. (\lambda \kappa. \text{left } m), \lambda x. x \text{claimed } x j)

e. John claimed Mary only saw F (TOM).
\text{claimed}(\text{only}(\lambda \kappa. \text{Saw } m), \lambda x. x \text{Saw } x \text{ m j})

f. John claimed that Mary only saw F (TOM).
\text{claimed}(\text{only}(\lambda \kappa. \text{Saw } m), \lambda x. x \text{Saw } x \text{ m j})

Here \text{only} is a function taking an ordered pair \((X, \kappa)\) and returning \text{true} just in case \(\overline{X} \kappa\) is true and there is no other (contextually relevant) meaning \(\overline{Y}\) of the same type as \(\overline{X}\) such that \(\overline{Y} \kappa\) is true. Thus the interpretation for John only saw MARY given (36a) entails that John saw Mary and that there is no other individual that John saw.

Examples (36a) and (36b) show that the pitch accent marker F can take arguments of different type (here, a transitive verb \text{Saw} rather than a noun phrase \text{Mary}).

Examples (36b), (36c), and (36d) show that the focus marker can take either a single word (e.g., \text{SAW}), a complex phrase (\text{SAW MARY}), or even an entire clause (\text{MARY LEFT}) as its argument. (In the last case, the only thing that John claimed was the Mary left; he did not claim that Tom left, he did not claim that Mary called, etc.)

Comparing (36e) with (36f), we see that once again embedded clauses must be delimited in order to arrive at the correct interpretation: (36e) does not mean that Tom was the only person that John claimed Mary saw; rather, it means that John claimed that Tom was the only person Mary saw. Thus the presence of the reset operator provided by \text{that} in (36f) is crucial to correctly delimiting the scope of \text{only}.

The next pair of examples shows the interaction of focus with quantificational noun phrases.

(37) a. John only F(SAW) someone.
\text{only}(\lambda \kappa. \text{Saw } m, \lambda y. \exists x. x \text{ j})

b. John only F (SAW SOMEONE).
\text{only}(\lambda \kappa. \exists x. x \text{Saw } x \text{ j})

c. John only saw F (SOMEONE).
\text{only}(\lambda \kappa. \exists x. x \text{Saw } x \text{ j})

Because the focus marker F takes a continuized meaning as its argument, it is able to handle quantificational foci with no trouble.

The next examples demonstrate the analysis of coordination.

(38) a. John left and Mary left. \quad \text{and(left )}(\text{left m})

b. John and Mary left. \quad \text{and(left )}(\text{left m})

c. John saw Mary and left. \quad \text{and}(\text{Saw } m)(\text{left j})

d. John saw Mary or everyone. \quad \text{or}(\text{Saw } m)(\forall x. \text{Saw } x \text{ j})

Note that (38d) involves coordinating a proper name with a quantificational noun phrase, which works fine.

(39) a. John claimed Mary or Tom left.
\text{or}(\text{claimed (left m}j)(\text{claimed (left t j})

b. John claimed that Mary or Tom left.
\text{claimed(}\text{or(claimed (left m}j)(\text{claimed (left t j})

Examples (39a) and (39b) show the behavior of coordination with and without a reset at the level of the embedded clause. In (39a), John either claims Mary left or claims Tom left; in (39b), John makes a single indeterminate claim that either Mary or Tom left. Since the interpretation in (39b) is certainly possible for this sentence, it provides additional evidence that a reset for delimiting embedded clauses must be available.

Finally, we have misplaced modifiers:
9. CONCLUSIONS

I have suggested that continuations provide an appealing analysis of a variety of natural language phenomena. It is possible for a skeptic to claim that natural language semantics has gotten by well enough without continuations so far. To be sure, each of the phenomena discussed above have well-established treatments in the linguistic literature that do not mention continuations. In the same way, computer scientists were perfectly able to deal meaningfully with the difference between call-by-value versus call-by-name before Plotkin's CPS analysis [10]. Yet I take it that these days everyone recognizes that a full understanding of evaluation disciplines requires continuations (or else something very like continuations). It is my hope, then, that the examples in this paper, or other analyses yet to be discovered, either individually or cumulatively, will someday make continuations seem as indispensable for the description of natural language as they currently are for the theory of computation and logic.

In closing, I would like to add one more subquestion to the list of questions discussed in the introductory section.

- [Innovation] Are there uses for continuations in natural language that computer scientists haven’t thought up yet?

Besides whatever intrinsic interest there might be in finding continuations in a natural setting, it is conceivable that once we look more closely, natural language will do interesting things with continuations that have not yet been dreamt up by theoretical computer scientists. I doubt that any of the natural language constructions treated above will seem breathtakingly new to an experienced continuation hacker; but then, I have chosen these examples precisely in order to maximize the degree to which they seem like garden-variety control operators. In the same spirit that pharmaceutical companies survey compounds harvested from tropical rain forests in hopes of finding medically useful substances unknown to laboratory scientists, we should consider that there are a lot of poorly understood languages out there—who knows what amazing control operators lurk in languages spoken in the forests of New Guinea?

10. ACKNOWLEDGMENTS

Thanks to Harry Mairson, Hayo Thielecke and Chung-chieh Shan.

11. REFERENCES


Direct compositionality on demand

Chris Barker (University of California, San Diego)

Abstract. This paper starts with the assumption that the expression saw everyone, as in John saw everyone, is a constituent. After all, it can be coordinated, clefted, focussed, questioned, pronominalized, you name it. According to the hypothesis of Direct Compositionality, just like any constituent, this phrase must have a well-formed semantic denotation. But everyone (typically) takes scope over an entire clause, so the semantics of everyone must somehow gain access to more material than just the constituent saw everyone. The standard treatment in terms of Quantifier Raising connects the quantifier directly with its nuclear scope, but does not associate any concrete denotation with the... paper proposes an explicit new account that does better in certain precise respects. The main innovation are what I call rules of disclosure, which provide a way for quantifiers to reveal their presence within arbitrary constituents. This allows the formal grammar presented below to link a quantifier directly with its scope, yet still characterize the syntactic and semantic content of intervening constituents. Because the fine-grained semantic analysis of the intervening constituents is optional, and so can be calculated only when we choose to examine the details, we have Direct Compositionality on demand.

1. Two equally valid views of the syntax-semantics interface

The problem is that many natural language expression types lead a double life, simultaneously here and there, masquerading as a local lump but somehow interacting directly with distant elements. The two main examples discussed here are bound anaphors, in which an anaphor depends for its value on some distant binder; and quantification, in which some local element takes semantic scope over a properly containing constituent. Such action-at-a-distance confronts theories of the syntax-semantics interface with a dilemma: should we interpret these elements locally, where they enter into the syntactic structure, or globally, where they take semantic effect?

Both approaches have staunch defenders. One well-established approach (e.g., Heim and Kratzer (1998)) emphasizes the global perspective, relegating the interpretation of anaphors to variable assignment functions, and postponing the interpretation of quantifiers until their scope has been revealed via Quantifier Raising at a level of Logical Form.

Another tradition, going back at least to Montague, and recently championed by many working within Categorial Grammar and its related frameworks, emphasizes local interpretation. Jacobson, in a series of papers (e.g., Jacobson (1999)), names this approach Direct Compositionality: roughly (to be made somewhat more precise below), expressions must deliver their entire semantic payload at the moment they enter into a syntactic relationship.

I claim that both views are indispensable, and that any grammar which ignores either mode of meaning is incomplete. Can we hope for a system that adheres to Direct Compositionality, but without giving up the clarity and simplicity of a global view?

Of course, there is an uninteresting, trivial answer, which involves simply combining a directly compositional grammar with an empirically equivalent, redundant grammar that allows non-local action.

I will propose what I believe is a more interesting answer, in the form of DCOD (for “Direct Compositionality On Demand”), a grammar in which the long-distance and local analyses arise from one and the same set of rules, none of which are redundant. The on-demand property allows us to have our cake and eat it too: we can connect an anaphor directly with its antecedent, and we can connect a quantifier to its scope in a single intuitive leap; or else, if we prefer, we can articulate each derivation into incremental steps that reveal the semantic contribution of each syntactic constituent.

To give a slightly more detailed preview, for every derivation in DCOD in which an expression is bound at a distance or takes wide scope, there will be a syntactically and semantically equivalent derivation on which the semantic contribution of each constituent is purely local. Furthermore, the interconvertibility of the two styles of derivation does not simply follow from grafting a direct-compositional grammar onto an action-at-a-distance grammar; rather, the duality in the syntax-semantics interface follows from a natural symmetry in the grammar itself. The symmetry concerns Gentzen’s rules of use and rules of proof. Roughly, in the grammar below, rules of use connect expressions directly over long distances, and embody the global view. Rules of proof help characterize the contribution of individual expressions within a complex constituent. Crucially, I introduce rules of disclosure, which establish an explicit connection between the long-distance semantic effect of an element with its local denotation. As a consequence, we can use the global-style rules of use without the slightest hesitation, confident that we can produce a parallel, strictly directly compositional elaboration upon demand.

The on-demand property, at least, as I envision it below, is by no means a general characteristic of type-logical grammars. In particular, Jäger’s (2005) LLC+q (LLC supplemented with rules for quantification) does not have the DCOD property. Nor do Display Calculi (Goré (1998), Bernardi (2002)) necessarily have the property, despite the fact that Display Calculi guarantee a similar constraint, the “display property”: that any syntactic constituent can be given a self-contained denotation. (More technically, that any structural element can be isolated on the left-hand side of a sequent.) The problem with these grammars is that the value determined by exercising the display guarantee often incorporates information about the derivation of elements external to the constituent in question. This is a fairly subtle but important point, and is developed in section 8.

This paper owes debts to three people: first and foremost, Pauline Jacobson, whose work inspired the workshop from which this paper developed, and who discussed many of the ideas below with me during my sabbatical visit to Brown in the fall of 2003. Second, David Dowty, whose remarks on the formal nature of direct compositionality expressed at the workshop and in conversation planted the seed that grew into this paper. Third, Gerhard Jäger, who first introduced me to the display property for type logics (“it is always possible to glue two adjacent constituents together before building the larger constituent. This requires a lot of cut applications...”) (in email from fall 2003), which approximates my on-demand property. (See section 8 for a discussion of the display property in comparison with the DCOD property.) Finally, I would like to point out that the crucial technical innovation that gives the logic presented below the on-demand property, namely, the rule of disclosure for quantification, was directly inspired by Jäger’s (2005) similar rule for his binding analysis. In some sense, then, this paper merely emphasizes the importance of a symmetry already present in Jäger’s LLC, and extends the same symmetry to quantification. Substantial improvements over the first draft are due to discussions with Chung-chieh Shan and Anna Szabolcsi.
2. Three characteristics of direct compositionality

The essence of direct compositionality is that every syntactic operation that combines two smaller expressions into a larger expression comes along with a semantic operation that combines the meanings of the smaller expressions to arrive at the meaning of the larger expression. This is just an informal characterization of Montague’s Universal Grammar (an instantiated in, e.g., Montague (1974)).

The conceptual simplicity and straightforwardness of direct compositionality has tremendous appeal. On the other hand, direct compositionality requires a certain amount of book-keeping: as we shall see, every semantically relevant aspect of a constituent must be available at the point at which the constituent enters into a syntactic structure, whether that aspect is relevant at that point or not. Thus combination rules must be adjusted to make use of information when it is required, and ignore it (but without discarding it!) when it is not.

2.1. No postponement. On the direct compositional view, the order of syntactic combination is identical to the order of semantic combination.

This view of the syntax-semantics interface contrasts with the standard Quantifier-Raising approach to quantifier scope:

\[
\text{S} \quad \text{NP} \quad \text{VP} \quad \text{QR} \\
\text{John} \quad \text{V} \quad \text{NP} \quad \text{everyone} \\
\text{saw} \quad \text{NP} \quad \text{S} \\
\]

The reason that this approach fails to be directly compositional is that there is a point in the derivation at which everyone has already been combined syntactically with saw to form a verb phrase constituent (as evident in the leftmost tree), but the verb phrase does not yet have a well-formed semantic denotation. In particular, there is no (relevant) way to directly combine a transitive verb such as saw of type \((e, (e, t))\) with a generalized quantifier such as everyone of type \((((e, t), t), t)\). Nor is there any obvious way of assigning a suitable denotation to the constituent saw everyone. It is only after the complete nuclear scope has been constructed (as in the rightmost tree) that Quantifier Raising resolves the type mismatch and a complete meaning emerges.

The QR approach is rather like building a car on an assembly-line: it may be convenient to install the steering wheel before attaching the doors, even if the control cables and electronic connections that allow the steering wheel to guide the wheels don’t get attached till later in the assembly process. Just so, in the QR model, the generalized quantifier everyone is inserted in direct object position, but its semantic control cables are left dangling until the rest of its clause has been assembled. I will call this sort of delayed evaluation POSTPONEMENT.

Forbidding postponement entails that no denotation has access to material contributed by an expression that is not part of the immediate syntactic expression. For instance, a pronoun must make its semantic contribution as soon as it enters into the syntactic construction, and cannot wait to find out what its antecedent is going to be.

The direct compositional ideal is a kind of zen semantics, living entirely in the moment of combination, unaware of what has happened or what is to come.

2.2. Full disclosure. It follows from forbidding postponement that all syntactically and semantically relevant aspects of elements within a constituent must be accessible when the constituent combines with other elements. In other words, if a constituent contains a bindable pronoun or a quantifier, then that fact must be evident by inspecting the syntactic and semantic features of the constituent.


2.3. Self-reliance. The flip side of full disclosure is that the analysis of constituents should be self-contained. This means that the syntactic category and the semantic value should depend only on the lexical items dominated by the constituent and on the structure internal to the constituent. The analysis should certainly not depend in any way on any element external to the constituent.

I will suggest below that although the display property possessed by most type-logical grammars is capable of providing analyses of any constituent, those analyses often violate self-reliance. (See especially section 8 below.) If the analysis of a subconstituent incorporates details that anticipate the specific structures in which it will be embedded, that is just a way of sneaking postponement through the back door.

3. DCOD, a logic with direct compositionality on demand

This section presents DCOD, the grammar analyzed in later sections. DCOD is a type-logical grammar. There are several quite different but more or less equivalent ways to present TLG. The two main styles are Natural Deduction versus the Sequent Calculus presentation. I will use the sequent presentation here, since that will best facilitate the discussion of cut elimination further below. I have done my best to keep in mind readers who are not already familiar with sequent systems, but it may be helpful to consult more leisurely presentations of sequent logics such as Moortgat (1997), Restall (2000), or Jäger (2005).

3.1. General strategy. The cut rule plays a special role in providing direct compositionality on demand. As discussed below, the cut rule expresses a kind of transitivity governing inference. Most discussions of the cut rule concentrate on proving that all cuts can be eliminated without reducing the generative power of the system, and thus that the cut rule is logically redundant. Cut elimination is important in order to prove such properties as logical consistency, guaranteed termination for the proof search algorithm, or that there will be at most a finite number of distinct interpretations for any given sentence. When
Axiom
t: A ⇒ t: A  
\[⇒\] m: A \[⇒\] p: C  
Cut
\[⇒\] m
p{m/ t}: C  

Rules of use (left rules):
\[⇒\] t  \[⇒\] f: (A/B), \[⇒\] g  \[⇒\] n  \[⇒\] C  
\[⇒\] m: A \[⇒\] p: C  
\[⇒\] m
p{m/ t}: C  

Rules of proof (right rules):
\[⇒\] t  \[⇒\] f: (A/B)  
\[⇒\] m: A \[⇒\] p: C  
\[⇒\] n: B  
\[⇒\] m
p{m/ t}: C  

Rules of disclosure (left-right rules):
\[⇒\] t  \[⇒\] f: (A/B), \[⇒\] g  \[⇒\] n  \[⇒\] C  
\[⇒\] m: A \[⇒\] p: C  
\[⇒\] m
p{m/ t}: C  

Figure 1. DCOD: a resource-sensitive logic with binding and quantification that guarantees Direct Compositionality On Demand. The only novel element compared with J"|eger's (2005:100) LLC is qLR.

considering these results, it is easy to get the impression that getting rid of cuts is always a very good thing.

But in fact it is being able to get rid of cuts that is the good thing. Cuts themselves are often quite useful: they correspond to proving a lemma, which you can then use and reuse in different contexts without having to reprove it each time. The connection with constituency is that it is possible to think of deriving, say, a noun phrase as a lemma. To say that a noun phrase is a constituent is to say that we could, if we chose, derive the noun phrase separately from the rest of the proof as a lemma, and then insert the lemma into the larger derivation in the position that the noun phrase occupies.

3.2. DCOD.

Each rule in Figure 1 relates either one or two antecedent sequents (appearing above the line) with exactly one consequent (below the line). Here, a sequent consists of a structure on the left-hand side of the sequent symbol (\(\Rightarrow\)) and a single formula on the right-hand side. A structure is either a single formula or an ordered pair \((\Gamma, \Delta)\) in which \(\Gamma\) and \(\Delta\) are both themselves structures. A formula is either a symbol such as \(np\), \(n\), or \(s\), or else has the form \(A\backslash B\), \(A\slash B\), \(A^{\dag}\), or \(q(A, B, C)\), where \(A\), \(B\), and \(C\) are metavariables over formulas. For instance, the sequent \((np/n, n) \Rightarrow np\) says that a structure consisting of a determiner of category \(np/n\) followed by a noun of category \(n\) can form an NP of category \(np\). From a linguistic point of view, formulas serve the role of syntactic categories, and structures indicate constituency. (You can think of a structure as a standard linguistic tree but without any syntactic category labels on the internal nodes.)

There are two kinds of information present in the rules given in Figure 1: logical information, encoded by formulas expressing types; and semantic information, encoded by terms in the λ-calculus expressing how the meanings of the elements in the inference rule relate to one another. Thus \(t\) stands for an expression of category \(B\) whose semantic value is named by \(x\). In the derivations below, I will often omit the semantic part of the derivation when it enhances clarity.

Two rules are special, the Axiom rule and the Cut rule. The Axiom rule is a simple fundamental kind of logical transitivity: given \(\Delta \Rightarrow A\), which says that \(\Delta\) constitutes a proof of the formula \(A\); and given \(\Gamma[A] \Rightarrow B\), which says that \(\Gamma\) is a proof of \(B\) that depends on assuming \(A\), it follows that \(\Gamma[A] \Rightarrow B\): substituting the reasoning that led to the conclusion \(A\) into the spot in the proof \(\Gamma\) where \(\Delta\) depends on assuming \(A\) constitutes a valid complete proof of \(B\). For instance, if you can prove that the plus \(dog\) forms a noun phrase (expressed in syntactic categories, \((np/n, n) \Rightarrow np\)), and if you can prove that any noun phrase plus \(barked\) forms a complete sentence (i.e., \((np, np) \Rightarrow s)\), then the cut rule entitles you to conclude that \((the\ dog)\ barked\) forms a complete sentence \(((np/ n, n) \Rightarrow np/s)\Rightarrow s)\).

In the cut rule, the notation \(\Gamma[A]\) schematizes over structures \(\Gamma\) that contain at least one occurrence of the formula \(A\) somewhere within it. The notation \(\Gamma[\Delta]\) in the consequent represents a structure similar to \(\Gamma\) except with \(\Delta\) substituted in place of (the relevant occurrence of) \(A\). (Examples immediately below will illustrate how this works.)

Apart from Axiom and Cut, most of the remaining rules introduces a single new logical connective (i.e., one not occurring in the antecedents) into the consequent, either on the left-hand side of the sequent symbol (the ‘left’ rules) or else on the right-hand side (the ‘right’ rules). As discussed below, the two crucial rules for the discussion here introduce a new connective on both sides of the sequent, and so are ‘left-right’ rules. (Purely for the sake of expository simplicity, I have omitted logical rules for the • (“product”) connective, which plays a prominent role in many type-logical discussions, but is not needed for any of the derivations below.)

A derivation is complete if all of its branches end (reading bottom up) using only instances of the Axiom as antecedents, and if each conclusion is derived from the antecedents.
above it via a legitimate instantiation of one of the logical rules. Here is a complete derivation proving that the dog barked is a sentence:

\[
\begin{align*}
\text{Axiom:} & \quad n \Rightarrow n \\
\text{Axiom:} & \quad np \Rightarrow np \\
\text{Axiom:} & \quad np \Rightarrow np \\
\text{Axiom:} & \quad s \Rightarrow s \\
\text{Cut:} & \quad \left( np/n, n \right) \Rightarrow np \\
\text{Cut:} & \quad \left( np/n, s \right) \Rightarrow s
\end{align*}
\]

I have placed boxes around the formulas that instantiate the A’s targeted by the cut rule. (From this point on, I will assume that applications of the Axiom rule are obvious, and so do not need to be explicitly indicated.)

An expression is generated by a type-logical grammar just in case there are lexical items whose category labels match the formulas in the result (bottommost) sequent and which appear in the same order as the formulas in the sequent. Thus if the word the has meaning the and category np/n, dog has meaning dog and category n, and barked has meaning barked and category np/s, then the derivation here proves that the dog barked has category s, with semantic interpretation barked(the(dog)). Furthermore, in that derivation, the dog forms a constituent.

In this case, the structure of the derivation corresponds to the constituent structure of the conclusion sequent: the NP (np/n, n) is a constituent in the conclusion sequent, and the main subderivation on the left (the first antecedent of the lowest inference) shows how to construct a subproof that (np/n, n) forms an NP. Unfortunately, this graceful correspondence between the form of the derivation and constituency (as determined by the structure in the conclusion sequent) is not guaranteed. Indeed, arriving at such a guarantee is the main topic of this paper.

4. Rules of use, rules of proof, and rules of disclosure

Following Lambek (1958) (in turn following Gentzen), the distinctive feature of TLG compared to plain categorial grammar is the ability to employ hypothetical reasoning. In terms of the sequent presentation used here, there are two types of logical rules, called left rules and right rules, or rules of use and rules of proof. Plain categorial grammar gets by with only (the equivalent of) the left rules, the rules of use. Because rules other than rules of use will be crucial for my main argument, this section briefly motivates the utility of rules of proof for linguistic analyses.

The usual motivating examples typically involve either function composition or relative clause formation. But these examples also require structural postulates that render function application associative, which would significantly complicate exposition at this point (structural postulates are discussed below in section 9).

But even in the absence of associativity, rules of proof can perform useful linguistic work by deriving a certain class of lifted predicates. Dowty (2000) suggests that adjectives may sometimes be re-analyzed as arguments. If sing has category v (where v abbreviates the category (np/s)/np), then if well is a verbal adjunct of category v\(\backslash\)v, sing well is correctly predicted to be a complex v, as shown here:

\[
\frac{v \Rightarrow v \quad v \Rightarrow v}{\left( v \text{ sing} \right) \Rightarrow \text{well}(\text{sing})}
\]

It is also possible for a higher-order verb to take a verbal modifier as an argument:

\[
\frac{v/v \Rightarrow v/v \quad v \Rightarrow v}{L}
\]

A language learner may not be able to tell whether to assign sing to the category v or the category v\(\backslash\)v. This is harmless, however, since as the following derivation shows, it is a theorem of DCOD (and any Lambek grammar) that anything in the category v can behave as if were of category v\(\backslash\)v(v\(\backslash\)v), the same category as behave:

\[
\frac{v \Rightarrow v \quad v \Rightarrow v}{v/v \Rightarrow v/v}
\]

Thus whether you think of sing in well as the argument of well or else as a lifted predic-icate that takes well as an argument is completely optional as far as the logic is concerned. We can assume, then, that the simplest lexicon will be one in which sing has category v, but verbs that require a modifier must be of category v\(\backslash\)v(v\(\backslash\)v).

One prediction that this analysis makes is that it should be possible to conjoin a simple verb like sing with a verb of higher category like behave. This is a good prediction, since sing and behave well certainly is a legitimate coordination:

\[
\frac{v/v \Rightarrow v/v \quad v \Rightarrow v}{v/v \Rightarrow v/v}
\]

In any case, rules of proof can be motivated independently of the main issues of this paper.

The innovative rules in DCOD, however, are not straightforward rules of proof. True rules of proof such as R \(\rightarrow\) R or \(\rightarrow\) R introduce a new connective only on the right side of the sequent. The rules of special interest here do introduce a new connective on the right side of the sequent, but they also introduce a new connective on the left side of the sequent at the same time. Following a suggestion of Chung-chieh Shan (p.e.), I will label such rules LR rules: simultaneously left rules and right rules. (Thus the rule Jäger (2005) refers to as \(\uparrow\)R I will call \(\uparrow\)LR.) These LR rules are not really rules of use, since they are not sufficient to license the use of, say, a pronoun or a quantifier; and they are not really rules of proof, since they do not discharge any hypotheses in the way that \(\rightarrow\)R or \(\rightarrow\)R do. (Section 7 will discuss other unusual properties of the LR rules with respect to cut elimination.) As we
shall see, what these new rules do is transmit information about subconstituents to higher levels. Therefore I will call them rules of disclosure.

Thus DCOD leaves open the question of what a true rule of proof would be for binding or for quantification. This is an area of active research; for one detailed view of how to handle scope-taking in a type-logical grammar, see Barker and Shan (in press).

5. First case study: binding

Here is the swooping (cut-free) derivation of \( \text{John, said he, left.} \)

\[
\begin{align*}
\forall & \Rightarrow \forall \\
\text{np} & \Rightarrow \text{np} \\
\text{np} & \Rightarrow \text{np} \\
\text{np} & \Rightarrow \text{np} \\
\text{np} & \Rightarrow \text{np} \\
\text{np} & \Rightarrow \text{np} \\
\text{np} & \Rightarrow \text{np} \\
\text{np} & \Rightarrow \text{np} \\
\end{align*}
\]

The form of the binding rule in Figure 1 requires some comment. In the second antecedent, \( \Gamma \wedge \Delta \vDash \L \) matches a structure \( \Gamma \) that contains an occurrence \( \alpha \) of the formula \( A \) and an occurrence \( \beta \) of formula \( A' \). In addition, \( \alpha \) must precede \( \beta \), but linear precedence will not be an important factor below. In the conclusion, \( \Gamma \wedge \Delta \vDash \L \) indicates the structure constructed by starting with \( \Gamma \), replacing \( \alpha \) with \( \Delta \), and replacing \( \beta \) with the formula \( A' \). In the diagram immediately above, \( \alpha \) is the left box, and \( \beta \) is the right box. In the conclusion, \( \alpha \) is replaced with \( np \) (no net change), and \( \beta \) is replaced with \( np'' \).

In this derivation, there is exactly one application of the binding rule \( \L \), operating over a potentially unbounded distance. I’ve boxed the NP targets of the \( \L \) rule, i.e., the elements to be bound. Intuitively, the rule in effect coindexes the boxed elements, guaranteeing that their interpretations will be linked. Naturally, the pronoun denotes a function from NP meanings to NP meanings, in this case, the identity function (see, e.g., Jacobson (1999) for discussion on this point).

Crucially, this analysis does not provide self-contained interpretations for each constituent. That is, this derivation is not directly compositional. In particular, consider the constituent \( he \ left \). If any substring of the larger sentence deserves to be a constituent, it is the embedded clause! The final sequent recognizes that \( he \ left \) is a syntactic constituent, since \( he \) and \( \left \) are grouped together into a substructure (as indicated by the parentheses in the bottommost sequent) yet there is no point at which the denotation of the constituent is treated as a semantic unit.

To see why, focus on the second (lower) instance of \( \L \), repeated here with full Curry-Howard labeling:

\[
\begin{align*}
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\end{align*}
\]

This is the step that articulates the embedded clause into a subject and a verb phrase, so this is the step that justifies the claim that \( he \ left \) are structural siblings. The contribution to the semantic value made by this step is the expression \( \text{left}(x) \), where \( x \) is a variable introduced by the instance of the Axiom rule that justifies the leftmost antecedent. This variable does not receive a value until the binding rule \( \L \) applies in the next derivational step, at which point the value of the embedded subject is bound to the value of the matrix subject. But the matrix subject is external to the constituent in question, so this labeling constitutes semantic postponement, and violates semantic self-reliance.

In other words, the cut-free derivation above clearly associates the position of the pronoun with its antecedent, so it accounts beautifully for the long-distance aspect of binding. However, quite sadly, it does not provide a full account of the sense in which \( he \ left \) is a constituent in its own right.

The main result of this paper guarantees the existence of a distinct derivation in DCOD that arrives at the same constituent structure and that has an identical semantic value, but in which each constituent can be associated with a self-contained semantic value. It is this second, related, derivation that provides the missing piece of the puzzle, and that characterizes \( he \ left \) as a constituent.

Arriving at such an alternative derivation involves use of the rule of proof \( \L \) and several applications of the cut rule. Here’s one way to do it:

\[
\begin{align*}
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\end{align*}
\]

This derivation for the verb phrase constituent \( \text{said he left} \) can participate in a complete derivation as follows:

\[
\begin{align*}
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\forall \Rightarrow \forall \\
\end{align*}
\]

The final sequents of the two completed derivations are identical, so both derivations provide the same constituency and the same final semantic interpretation. However, the second derivation provides a self-contained denotation for each constituent, where ‘self-contained’ means that each element in the denotation either turns out to be the denotation of a word in the final sequent (e.g., \( left \)) or else is a variable that is bound within the denotation (e.g., \( x \in \lambda x. \text{left}(x) \) but not in \( \text{left}(x) \)).

This cut-full derivation beautifully captures the intuition that \( he \ left \) is a constituent. However, now the link between the pronoun and its binder has been sadly obscured.

In order to emphasize the role of the cut rule in encapsulating the meaning of a subconstituent, the diagram above provides Curry-Howard semantic labels only for the final sequent in a derivation and for the first antecedent of each application of the cut rule, i.e., the antecedent that corresponds to an encapsulated constituent. (It should be clear how to complete the labelings of each of the derivations based on the labeling annotations in
Figure 1.) Here is what a complete labeling for the constituent he left would look like:

\[ x: \text{np} \Rightarrow x: \text{np} \]
\[ p: s \Rightarrow p: s \]
\[ (\text{he left}) \Rightarrow x: \text{np} \]
\[ (\text{np s})/\text{np}, \text{np} \]
\[ \lambda x. \text{left}(\text{he}(x)) \]

This complete labeling of the constituent shows how the \([LR]\) rule binds the argument of the pronoun, making the denotation self-contained. The troublesome element in the swooping derivation was the variable \( x \), whose value was determined from outside the constituent. As Jäger points out, \( \lambda \) rules in the DCOD grammar are (simplified) versions of Jäger’s rules,\(^2\): \( \Rightarrow \)

\[ \text{np} \]
\[ \text{s} \]
\[ \text{LR} \]
\[ \text{\((\text{np s})/\text{np}, \text{np} \))} \Rightarrow \text{np} \]
\[ \text{qLR} \]
\[ \text{\((\text{np s})/\text{np}, \text{np}, \text{np})\)} \Rightarrow \text{np} \]
\[ \text{eo}(\lambda x. f(\text{np}(x))) \]

The derivations are the same until the last step. Instead of applying the \([L]\) rule in order to bind the pronoun to the matrix subject, the \([LR]\) rule discloses the presence of the bindable pronoun at the top level of the derivation. (There is, of course, an equivalent directly compositional derivation as well.)

In fact, to the extent that Jäger’s left and left-right rules are indeed two logical aspects of a single inference type, they arguably provide an analysis of bound versus deictic pronouns that is more unified even than Jacobson’s, since on her analysis, the analog of the binding rule (\([L]\), her \( z \) type-shifter; \( g \)) do not resemble each other in any obvious way.

6. Second case study: quantification

The analysis of binding in DCOD is (intended to be) identical to Jäger’s (2005) treatment. Therefore his LLC has direct compositionality on demand, at least with respect to binding. This section considers quantification, a different kind of action at a distance, and shows how the \([Q]\) rule in Figure 1 guarantees the DCOD property for quantification as well.

Here is the swooping derivation of John saw everyone:

\[ \text{np} \Rightarrow \text{np} \]
\[ s \Rightarrow s \]
\[ (\text{np}, \text{np})/\text{np}, \text{np} \]
\[ \Rightarrow s \]
\[ \text{LR} \]
\[ \text{\((\text{np s})/\text{np}, \text{np}, \text{np})\)} \Rightarrow \text{np} \]
\[ \text{qLR} \]
\[ \text{\((\text{np s})/\text{np}, \text{np}, \text{np})\)} \Rightarrow \text{np} \]
\[ \text{\((\text{np s})/\text{np}, \text{np}, \text{np})\)} \Rightarrow \text{np} \]
\[ \text{\((\text{np s})/\text{np}, \text{np}, \text{np})\)} \Rightarrow \text{np} \]
\[ \text{\((\text{np s})/\text{np}, \text{np}, \text{np})\)} \Rightarrow \text{np} \]
\[ \text{\((\text{np s})/\text{np}, \text{np}, \text{np})\)} \Rightarrow \text{np} \]
\[ \text{\((\text{np s})/\text{np}, \text{np}, \text{np})\)} \Rightarrow \text{np} \]

The \([Q]\) rule applies only after the entire scope of the quantificational element is in view, linking the quantifier with its nuclear scope in one leap. Nevertheless, the final structure clearly indicates that saw everyone is a syntactic constituent.

Now for the corresponding direct compositional derivation.

\[ \text{np} \Rightarrow \text{np} \]
\[ \text{np} \Rightarrow \text{np} \]
\[ \text{LR} \]
\[ \text{\((\text{np s})/\text{np}, \text{np}, \text{np})\)} \Rightarrow \text{np} \]
\[ \text{\((\text{np s})/\text{np}, \text{np}, \text{np})\)} \Rightarrow \text{np} \]
\[ \text{\((\text{np s})/\text{np}, \text{np}, \text{np})\)} \Rightarrow \text{np} \]
\[ \text{\((\text{np s})/\text{np}, \text{np}, \text{np})\)} \Rightarrow \text{np} \]

The version of \([LR]\) given here is simplified from Jäger’s (2005:100) more general rule. Jäger’s rule handles cases in which more than one pronominal element is bound by the same antecedent.
This derivation for the verb phrase constituent *saw everyone* can participate in a complete derivation as follows:

\[
(np\langle s\rangle)/np, q(np, s, s, s) \Rightarrow q(np\langle s, s, s\rangle)
\]

\[
(np, np\langle s\rangle) \Rightarrow s \quad s \Rightarrow s \quad qL
\]

\[
(np, np\langle q(s, s, s)\rangle) \Rightarrow s \Rightarrow s \\
\text{Cut}
\]

Once again, *saw everyone* is a constituent. It is not, however, a simple VP of category *np\langle s\rangle*; rather, full disclosure requires that the be of category *q(np\langle s, s, s\rangle)*, that is, a quantificational verb phrase: something that functions locally as a verb phrase, takes scope over a clause, and produces a clause as a result. In other words, arriving at a directly compositional treatment has turned the verb phrase transparent, so that the category reveals the presence of a quantificational element somewhere within.

As in the previous case study, once again I will provide a full Curry-Howard labeling for the lemma analyzing the constituent under study, *saw everyone*:

\[
\begin{aligned}
\text{x: np} & \Rightarrow \text{x np} \quad r: (np\langle s\rangle) \Rightarrow r: (np\langle s\rangle) \\
\text{saw} & \Rightarrow q(np\langle s, s, s\rangle) \\
(np\langle s\rangle)/np & \Rightarrow q(np\langle s, s, s\rangle) \\
\text{saw everyone} & \Rightarrow \forall x \text{everyone}(\forall x f(saw(x)))
\end{aligned}
\]

Here \(f\) is a function from verb phrase denotations of category *np\langle s\rangle* to clause denotations of category *s*. As a result, the semantic type of this constituent is \(\langle\langle\langle e, t, t\rangle\rangle t\rangle\): a function from a function from verb phrase meanings to truth values to truth values. This is exactly the type for a continuized verb phrase in the (directly compositional) continuation-based approach to quantification described in Barker (2002).

Once again, the swooping derivation is simpler, both conceptually and practically, an advantage that increases dramatically as the quantifier takes wider and wider scope. And once again the DC version imposes the discipline of full disclosure and self-reliance, giving a full accounting of the syntactic and semantic nature of each syntactic constituent.

### 6.1. The locus of scope ambiguity

When more than one quantificational expression is present, the order in which the quantifiers are introduced into the derivation can give rise to scope ambiguity. Because we have the DCOD property, we know that every distinct constituent will have an equivalent derivation in which each constituent is given a self-contained denotation. We might ask, therefore, which constituents must or can participate in scope ambiguities.

In order to explore this question, I will consider a sentence that contains three quantificational expressions, such as *Most people gave something to every child*. Assume that this sentence has a construal on which it entails that for every child, most people gave that child something, and that the scope relations are *every > most > some*. For expository simplicity, I will ignore the preposition to.

Let \(Q\) abbreviate the category *q(np, s, s)*. Then on the swooping derivation, the scope construal depends in a straightforward manner on the order of the instantiations of the \(qL\) rule:

\[
\begin{aligned}
np & \Rightarrow np \quad s \Rightarrow s \\
(np, np\langle s\rangle) & \Rightarrow (np, np\langle s\rangle) \\
((np\langle s\rangle)/np) & \Rightarrow (np\langle s\rangle)/np \\
\text{saw everyone} & \Rightarrow Q(np\langle s\rangle)/np \\
((np\langle s\rangle)/np, np) & \Rightarrow Q((np\langle s\rangle)/np) \\
(np, np\langle q(s, s, s)\rangle) & \Rightarrow (np, np\langle q(s, s, s)\rangle) \\
(np, np\langle q(s, s, s)\rangle) & \Rightarrow Q((np\langle q(s, s, s)\rangle)/np, np) \\
((np\langle q(s, s, s)\rangle)/np, Q) & \Rightarrow Q((np\langle q(s, s, s)\rangle)/np, Q)
\end{aligned}
\]

Because we could have instantiated the two rightmost applications \(qL\) rules in the opposite order, there are two distinct possible analyses of the verb phrase. These alternatives correspond to the two relative scope relations between \(Q_1\) and \(Q_2\). In general, in a directly compositional derivation, the derivation of each constituent fully determines the relative scope of all (and only) those quantifiers contained within that constituent.

At this point, we need only show how to compose the subject quantifier with the verb phrase:

\[
\begin{aligned}
np & \Rightarrow np \quad s \Rightarrow s \\
(np, np\langle s\rangle) & \Rightarrow (np, np\langle s\rangle) \\
(np, np\langle q(s, s, s)\rangle) & \Rightarrow (np, np\langle q(s, s, s)\rangle) \\
((np\langle s\rangle)/np) & \Rightarrow (np\langle s\rangle)/np \\
(np, np\langle q(s, s, s)\rangle) & \Rightarrow (np, np\langle q(s, s, s)\rangle) \\
(np, np\langle q(s, s, s)\rangle) & \Rightarrow (np, np\langle q(s, s, s)\rangle) \\
(np, Q_1(np\langle s\rangle)/np, np) & \Rightarrow Q_1(np\langle s\rangle)/np \\
((np\langle s\rangle)/np, Q_1(np\langle s\rangle)/np) & \Rightarrow Q_1((np\langle s\rangle)/np, Q_1(np\langle s\rangle)/np)
\end{aligned}
\]

To complete the derivation of the entire sentence, simply cut the derivation of the verb phrase given just above against this derivation of a complete clause.
The order of the \(\forall L\) rules here controls whether the subject outscopes both quantifiers in the verb phrase, or else takes intermediate scope (as shown here), or else takes narrow scope with respect to both verb phrase quantifiers. However, this part of the derivation has no power to affect the relative order of the quantifiers within the verb phrase, since that is fully determined by the derivation of the verb phrase, as described above.

Thus in a directly compositional derivation with full disclosure, the derivation of each constituent determines the relative scoping of all of the quantifiers it contains. In particular, even though the quantifiers involved take scope only over complete clauses (and not over verb phrases), the verb phrase nevertheless exhibits its own local scope ambiguity, and it is those local scope relations that determines the contribution of the verb phrase to the scope relations of the larger derivation in which it is embedded. In other words, semantic self-reliance requires that each constituent take full responsibility for every aspect of the contribution of its contents within the larger derivation.

7. Direct compositionality on demand

Having provided two examples in detail, I now establish that it is always possible to provide both a swooping and a directly compositional analysis for any sentence generated by DCOD. Furthermore, the two analyses are guaranteed equivalent structurally and semantically, and interconvertible in both directions.

Converting from the DC analysis to the swooping analysis requires eliminating cuts, so I first show that the cut rule is admissible in DCOD (i.e., that cuts can always be eliminated). Then I show how to add cuts back in where desired in order to construct a fully DC analysis.

7.1. Cut elimination for DCOD. Jäger (2005:102) sketches cut elimination for LLC (see pages 43ff, 102ff, and 127), a logic similar to DCOD except in two respects: (1) DCOD leaves out the \(\exists L\) and \(\exists R\). Therefore we need only show that adding \(\forall L\) and \(\forall R\) does not interfere with cut elimination. The only potentially troublesome situations are ones in which some application of \(\forall R\) is cut against an instance of either \(\forall L\) or \(\forall R\). Consider first a derivation containing a cut of \(\forall R\) against \(\forall L\). For each derivation of this form, there will necessarily be an equivalent derivation of the following form:

\[
\Pi[A] \Rightarrow B \quad \Pi[\forall R(A, C, D)] \Rightarrow \forall R(B, C, D) \quad \Gamma[D] \Rightarrow E \quad \forall L
\]

\[
\Gamma[\forall \Pi[\forall R(A, C, D)]] \Rightarrow E \quad \text{Cut}
\]

For each derivation of this form, there will necessarily be an equivalent derivation of the following form:

\[
\Pi[A] \Rightarrow B \quad \Delta[D] \Rightarrow E \quad \forall L
\]

\[
\Gamma[D] \Rightarrow E \quad \text{Cut}
\]

The replacement derivation still has a cut, but it is a cut of a lower degree (see, e.g., Jäger (2005) for a suitable definition of degree; the intuition is that the cut involves a lemma covering a smaller amount of material). As long as it is possible to replace any cut with one of strictly lower degree, cuts can be pushed upwards until they reach an axiom and can be entirely removed.

If \(\forall R\) were a true rule of proof, there would be nothing more left to say concerning cut elimination. But because the \(\forall R\) rule is two-sided—that is, it introduces the \(\forall\) connective on both the left and the right—then just as with Jäger’s binding rules, we must also consider cutting an instance of \(\forall R\) with another instance of \(\forall L\):

\[
\Pi[A] \Rightarrow B \quad \Pi[\forall R(A, C, D)] \Rightarrow \forall R(B, C, D) \quad \Gamma[B] \Rightarrow E
\]

\[
\Gamma[\forall \Pi[\forall R(A, C, D)]] \Rightarrow \forall R(E, C, D) \quad \text{Cut}
\]

Once again, given the resources provided by the antecedents of the first derivation, we can reconfigure the proof with a cut of strictly lesser degree.

\[
\Pi[A] \Rightarrow B \quad \Gamma[B] \Rightarrow E \quad \text{Cut}
\]

\[
\Gamma[\Pi[\forall R(A, C, D)]] \Rightarrow \forall R(E, C, D) \quad \text{Cut}
\]

Thus the cut rule is admissible in DCOD, which is to say that any derivation in DCOD containing a cut can be replaced with an equivalent derivation that is cut-free.

7.2. Adding cuts back in to provide Direct Compositionality on Demand. Now that we know that is possible to eliminate cuts completely, we can consider adding back some cuts in order to provide direct compositionality.

**Definition** (Directly composable) Consider a specific derivation in DCOD with conclusion \(\Gamma[\forall \Pi]\) \(\Rightarrow A\). That derivation is **directly composable** just in case there exists a formula \(X\) and an equivalent derivation of the following form:

\[
(\Delta, \Pi) \Rightarrow X \quad \Gamma[X] \Rightarrow A
\]

\[
\text{Cut}
\]

in which each of the antecedents are directly composable. (Two derivations counts as equivalent if they have identical conclusion sequents with Curry-Howard labelings that are equivalent up to \(\alpha\) and \(\beta\) equivalence.)

The first step in establishing that DCOD is directly compositional is the following simple but important observation:

**Lemma** (Categorization) Every structural constituent can be assigned a category.

**Proof.** Only two rules create complex constituents, i.e., structures of the form \(\Delta, \Pi\), namely, \(\forall L\) and \(\forall R\). Consider \(\forall L\):

\[
\Delta \Rightarrow A \quad \Gamma[B] \Rightarrow C
\]

\[
\Gamma[\forall \Pi] \Rightarrow C \quad \text{\(\forall L\)}
\]

The rule itself provides us with a suitable choice for categorizing the constituent, namely, \(B\), and we can replace the \(\forall L\) step just given with the following reasoning:

\[
\Delta \Rightarrow A \quad B \Rightarrow B
\]

\[
(\Delta, \forall L) \Rightarrow C \quad \text{\(\forall L\)}
\]

In other words, using \(\text{Cut}\) we can arrange for every application of \(\forall L\) to have an Axiom instance as its second antecedent.
The situation for /L is symmetric.

The categorization lemma guarantees that each constituent can be associated with a syntactic category, but that is not enough to ensure full disclosure and semantic self-reliance. There are only two situations in DCOD in which self-reliance might be violated: when the binding rule /L links an anaphor inside the constituent in question with an antecedent outside the constituent; or when the quantification rule qL links a quantifier inside the constituent with a nuclear scope that properly contains the constituent in question. In each case, it will be necessary to adjust the syntactic category identified by the categorization lemma in order to align it with the principle of full disclosure.

**Full disclosure for binding.** A violation of self-reliance due to binding has the following form:

\[ \Sigma \Rightarrow A \quad \Pi[A][\Delta[B], \Gamma] \Rightarrow C \]

This is just an application of /L where the anaphor (but not the antecedent) is inside the constituent (Δ, Γ).

First, assume that the derivations of the antecedents above the line are directly composable. Then there is some formula X such that we can cut out the constituent under consideration, (Δ, Γ), as a separate subproof.

\[ (\Delta[B], \Gamma) \Rightarrow X \]

\[ \Pi[X] \Rightarrow C \]

\[ \Sigma \equiv A \]

\[ \Pi[A][\Delta[B], \Gamma] \Rightarrow C \]

\[ \Rightarrow \]

\[ /L \]

Next, we use /LR to disclose the fact that the constituent contains an anaphor, replacing the category X with X^A. We can also bind the anaphor at the stage at which we form X^A, as long as we interchange the order of /L with Cut:

\[ (\Delta[B], \Gamma) \Rightarrow X \]

\[ \Pi[X] \Rightarrow C \]

\[ \Sigma \Rightarrow A \]

\[ \Pi[A][\Delta[B], \Gamma] \Rightarrow C \]

\[ \Rightarrow \]

\[ /L \]

\[ \Pi[A][\Delta[B^A], \Gamma] \Rightarrow C \]

\[ \Rightarrow \]

\[ Cut \]

Then the leftmost antecedent of the cut inference constitutes a complete and self-contained proof that (Δ, Γ) is a constituent of category X^A, and the original derivation is directly composable.

The situation when the anaphor is inside Γ instead of Δ is closely analogous.

In constructing the DC derivation, a instance of /LR is cut against an instance of /L. This is exactly the sort of situation that the cut elimination theorem is at pains to eliminate. But this is exactly what we need in order to arrive at a self-contained analysis of the syntactic constituent under consideration.

**Full disclosure for quantification.** A violation of self-reliance due to quantification has the following form:

\[ \Sigma \equiv \Pi[[\Delta[B], \Gamma] \Rightarrow C \quad \Sigma[D] \Rightarrow E \]

\[ \Rightarrow \]

\[ qL \]

This is just an application of qL where the quantifier and its scope are on different sides of the targeted constituent boundary. The argument proceeds exactly as for the binding case.

Once again, assume that the antecedents are directly composable, so that we can cut out the constituent under consideration, (Δ, Γ), as a separate subproof.

\[ (\Delta[B], \Gamma) \Rightarrow X \quad \Pi[X] \Rightarrow C \quad \Sigma[D] \Rightarrow E \]

\[ \Pi[[\Delta[B], \Gamma]] \Rightarrow C \]

\[ \Sigma[[\Delta[q(B, C, D)], \Gamma]] \Rightarrow E \]

\[ qL \]

Next, we use qLR to disclose the fact that the constituent contains a quantifier, replacing the category X with q(X, C, D). At the same stage, we can fix the scope of quantificational quantifiers, and since none of these operations introduces new constituents, the process is guaranteed to terminate.

**Proposition (Direct Compositionality On Demand)** For any valid sequent generated by DCOD, there is an equivalent swooping (cut-free) proof in which anaphors and their antecedents are coinjected by a single application of /L, and in which quantificational elements and their scope are related by a single application of qL. For each such derivation, there is another, equivalent, directly composable proof in which each constituent receives a self-contained analysis that obeys full disclosure and semantic self-reliance.

**Proof.** For each constituent in the final conclusion structure, working from smallest to largest, and beginning with the axioms and working downwards, apply the categorization lemma and impose full disclosure. Since each application of a rule (other than cut) adds at least one logical element and their scope are related by a single application of /L, and since none of these operations introduces new constituents, the process is guaranteed to terminate.

One of the pleasant properties of cut-elimination is that it leads to an algorithm for deciding whether a sequent is derivable. Since each application of a rule (other than cut) adds at least one logical connective, the conclusion will always be strictly more complex than the antecedents. One can simply try all applicable rules, stopping when a valid proof is found.

Since the on-demand theorem introduces cuts, it is worth considering whether there is an algorithm for finding the DC proof. Since there is an algorithm for finding a cut-free proof, and since there is an algorithm for constructing the equivalent DC proof, there is an algorithm for constructing the DC proof.

In fact, there is an alternative parsing strategy for logics that satisfy the DC on-demand property. Given a set of lexical categories, since the DC analysis of each constituent has the subformula property, the complete set of usable proofs for each constituent can be constructed in finite time. For a string of length n, the number of possible constituents is at most n^2, and for each constituent the time cost for trying all relevant cuts is proportional to n, giving a time cost of order n^3 in the length of the string.

**8. Why the display property alone is not sufficient**
There is a class of substructural logics called Display Logics that are relevant for type-logical grammar. Display Logics have what is called the Display Property. As Bernardi (2001:31) puts it, “any particular constituent of a sequent can be turned into the whole of the right or the left side by moving other constituents to the other side [of the sequent symbol]” (Bernardi 2001:31).

At first blush, the display property sounds like exactly what we want, since it guarantees that it is always possible to arrive at a syntactic category and a self-contained denotation for any syntactic constituent. Furthermore, there will be such an analysis for each distinct interpretation provided by the grammar.

Certainly no grammar can be directly compositional without having the display property. For instance, the standard QR story (e.g., as presented in Heim and Kratzer (1998)) does not have the display property, and is not directly compositional, since there is no way to factor out a syntactic and semantic analysis of a verb phrase such as saw everyone (at least, not when the quantifier takes scope over an entire clause).

However, in this paper I am advocating the desirability of an even stronger property than the display property. I will first give two concrete examples of the sort of analyses the display property gives for constituency, then I will point out some shortcomings that motivate seeking a stronger property such as DCOD.

First, here is an analysis in which the display property provides a category and a denotation for the constituent saw everyone in John saw everyone:

\[
\begin{align*}
\text{[as derived in section 6]} \\
(np, ((np,s)/np, q(np,s,s))) & \Rightarrow s & \text{L} \\
((np,s)/np, q(np,s,s)) & \Rightarrow np/s & \text{R} \\
(np, np/s) & \Rightarrow s & \text{Cut} \\
\text{[np John saw everyone]} & \Rightarrow eo(\lambda x.\lambda y.\text{assaw}(x)(j))
\end{align*}
\]

The peculiar thing about this derivation is that it begins with a complete derivation of the entire sentence: the boxed sequent is identical to the final conclusion sequent. Having the final conclusion before us in this way allows us to work backwards to figure out what the contribution of the verb phrase must have been, using a technique familiar from basic algebra:

\[
p \ast (r \ast (y \ast x)) = s \\
(r \ast (y \ast x)) = s/p \\
(y \ast x) = (s/p)/r
\]

Thus the logic of basic algebra has the display property.3 In this instance of the display property approach, the constituent saw everyone has category np/s, and although that hides the fact that the verb phrase is quantificational, it certainly works out to the correct final result. The problem is that this analysis for the constituent only works in situations in which the quantifier does not need to take wider scope. That is, the analysis of the constituent is sensitive to the material that surrounds it, violating the spirit of self-reliance.

In order to demonstrate this sensitivity to factors external to the constituent, consider an analogous treatment of “Someone claimed John saw everyone” (with the interpretation on

\[
\begin{align*}
\text{[as derived in section 6]} \\
(np, ((np,s)/np, q(np,s,s))) & \Rightarrow s & \text{L} \\
((np,s)/np, q(np,s,s)) & \Rightarrow np/s & \text{R} \\
(np, np/s) & \Rightarrow s & \text{Cut} \\
\text{[np John saw everyone]} & \Rightarrow eo(\lambda x.\lambda y.\text{assaw}(x)(j))
\end{align*}
\]

Thus the logic of basic algebra has the display property.3 In this instance of the display property approach, the constituent saw everyone has category np/s, and although that hides the fact that the verb phrase is quantificational, it certainly works out to the correct final result. The problem is that this analysis for the constituent only works in situations in which the quantifier does not need to take wider scope. That is, the analysis of the constituent is sensitive to the material that surrounds it, violating the spirit of self-reliance.

In order to demonstrate this sensitivity to factors external to the constituent, consider an analogous treatment of “Someone claimed John saw everyone” (with the interpretation on
provide a completely satisfying characterization of the contribution of the constituent (at least, not of the contribution proper to the constituent itself). It’s an indirect method at best, as if we tried to describe a hand exclusively by showing what it looks like inside a variety of gloves and mittens.

The analysis of quantification in Hendriks’ (1993) Flexible types has similar properties, in that the category of a constituent containing a quantifier can be arbitrarily complex depending on how wide a scope the quantifier needs to take. See Barker (2005) for a discussion of Hendriks’ system in the context of Jacobson’s variable-free, directly compositional treatment of binding.

9. Structural postulates

Unlike most type-logical grammars, the DCOD grammar as presented above does not contain any structural postulates. Most type-logical grammars contain structural postulates that at least make implication associative. There is strong linguistic motivation for making implication associative (at least under highly constrained circumstances), including analyses of so-called non-constituent coordination (Steedman 1985, Dowty 1988, 1997)) and various applications of function composition (e.g., Jacobson 1999)), as well as many multi-modal analyses (see Moortgat 1997 for a survey).

Structural postulates complicate the discussion of direct compositionality considerably. The reason is that direct compositionality is all about constituency, and the express purpose of structural postulates is to scramble constituent structure. For instance, here is a structural postulate that provides associativity:

\[ (A, (B, C)) \Rightarrow D \]

\[ ((A, B), C) \Rightarrow D \]

\[ \text{ASSOC} \]

The double line indicates that the inference is valid in both directions: given the top sequent, infer the bottom one, or given the bottom sequent, infer the top one. Without using this postulate, we can easily prove that \([John \ saw \ Mary] \] is a sentence, i.e., \( (np, ((np, s)/np, np)) \Rightarrow s \). As a result of adding this postulate to the grammar, we can also prove that \([John saw Mary], \) with the opposite constituency, is a sentence. The display property allows us to calculate an appropriate self-contained denotation for the pseudo-constituent \( \text{John saw}; \)

\[
\begin{align*}
np & \Rightarrow np & \Rightarrow s & \Rightarrow s & \Rightarrow s \\
np & \Rightarrow np & \Rightarrow s & \Rightarrow s & \Rightarrow s \\
\text{L} & \Rightarrow \text{L} & \Rightarrow \text{L} & \Rightarrow \text{L} & \Rightarrow \text{L} \\
\text{R} & \Rightarrow \text{R} & \Rightarrow \text{R} & \Rightarrow \text{R} & \Rightarrow \text{R} \\
\text{Cut} & \Rightarrow \text{Cut} & \Rightarrow \text{Cut} & \Rightarrow \text{Cut} & \Rightarrow \text{Cut} \\
\end{align*}
\]

As mentioned above, there is fairly compelling evidence motivating associativity as desirable from a linguistic point of view. For instance, adding associativity allows deriving Right Node Raising examples such as \( \text{John saw and Tom called Mary}, \) in which the alleged constituent \( \text{John saw} \) coordinates with \( \text{Tom called}. \)

At least for terminological purposes, it is convenient to discriminate between two notions of constituent: NATURAL constituency, as determined by the function/argument structure of the lexical predicates involved, versus CALCULATED constituency, as derived from natural constituency via structural postulates. The display property will always provide an appropriate analysis for calculated constituents.

In contrast to my remarks in the previous section criticizing the result of using the display property to arrive at analyses of natural constituents, the display property technique seems to be exactly the right way to understand a calculated constituent such as \( \text{John saw}: \) it is the quotient of a complete sentence after factoring out the direct object.

Full disclosure of anaphors and quantifiers still applies to calculated constituents. For instance, in \( \text{John, saw, and his, mother called, Mary}, \) in a DC analysis the calculated constituent \( \text{his mother called} \) will have category \((s/np)\text{np}^\psi\) with corresponding denotation.

10. Conclusions

Bernardi describes (a Natural Deduction version of) the \( q. \) rule given above, and remarks (p. 97) that “in the multimodal setting ... the \( q \) connective of course cannot be a primitive connective”. Instead, Bernardi suggests synthesizing \( q \) via a collection of multimodal logical and structural rules (see Moortgat (1997) for one concrete implementation of this strategy), and defining the swooping \( q \) as a “derived inference”. This is perfectly coherent and feasible, of course; but it relegates the long-distance mode of analysis to a rule that is entirely redundant and eliminable (“admissible” in the logical jargon).

This paper explores the possibility of finding a grammar in which both views of constituency are simultaneously present, but each one of whose rules is indispensable. For instance, unlike Bernardi’s derived inference rule for \( q \), none of the rules in the DCOD logic given above is admissible. That is, eliminating any rule other than \( \text{Cut} \) would reduce the number of valid sequents. In particular, it is only possible to prove the sequent \( (\lambda x. f(ax(x))) \Rightarrow q(np/np, np) \Rightarrow s \) using \( q.L \), so \( q.L \) is not admissible.

There are many pressures on the design of a grammar, and I do not expect that any system based on DCOD will serve all purposes. Rather, I offer DCOD here as an example showing that it is possible to reconcile the local and long-distance aspects of the syntax-semantic interface within a single unified grammar. With any luck, there will be other grammatical systems that can semantically link distant elements directly, yet still provide complete, self-contained constituent analyses with full disclosure: long-distance linking, but with direct compositionality on demand.

References


Introduction: Direct Compositionality

CHRIS BARKER AND PAULINE JACOBSON

The papers collected in this volume grew out of a workshop “Direct Compositionality: A Workshop” sponsored by NSF Grant BCS-0236496 and held in June 2003 at Brown University. In addition to the eleven papers published here, the workshop featured three additional papers by Daniel Büring, Alexis Dimitrakoudis, and Danny Fox. Moreover, all of the contributors here plus the three mentioned above served as discussants for one of the other workshop papers. The quality of the discussant comments, the contribution of this aspect of the conference to the workshop, and the influence of the discussant comments on the final papers was quite extraordinary, and thus we would like to thank everyone who attended the workshop, as well as NSF for their support.

1.1 What is Direct Compositionality?

In its simplest formulation, the hypothesis of direct compositionality can be summed up with the following slogan:

The syntax and the semantics work together in tandem.

At the very least, this slogan imposes a certain discipline on the syntax- semantics interface; it requires, for example, that for every syntactic operation there must be a corresponding semantic operation. Of course, this by itself may not seem like a requirement with considerable consequences, since the relevant semantic operation could be as trivial as the identity function on meanings. But indeed this does have a significant consequence: for it ensures that the input to every syntactic operation—or, put differently, every expression which is computed in the syntax—actually does have a meaning. And therein lies one of the major differences between direct compositionality and some other views of the organization of the grammar where interpretation is “postponed” until a later stage in the grammatical computation; we will return to this point below. Thus direct compositionality names one of the core principles of Montague’s Universal Grammar, and the fragment in Montague’s (1973) “The Proper Treatment of Quantification in Ordinary English” (PTQ) stands as one of the first and most influential directly compositional analyses of a significant portion of English grammar. Indeed, this remains one of the most influential semantic analyses of any sort.

Of course, direct compositionality is a type of compositionality, where (roughly) a theory of grammar is compositional if the meaning of an expression can be reliably computed from the meanings of its parts. Many discussions of compositionality concern themselves with the extent to which natural language is or is not compositional; the papers in this volume for the most part assume that natural language is predominantly or essentially compositional, and consider instead the following question: when natural language is compositional, is that compositionality direct or not?

As with any discussion of compositionality, there will always be some who question whether direct compositionality is “merely” a methodological preference, or whether there are genuine, testable empirical consequences that follow from the claim that natural languages are directly compositional. At the very least, direct compositionality makes concrete empirical predictions about which linguistic objects have meanings. To illustrate with a concrete example, consider the standard, non-directly compositional analysis of quantifier scope construal: a verb phrase such as saw everyone fails to have a semantic interpretation until it has been embedded within a large enough structure for the quantifier to raise and take scope (e.g., Someone saw everyone). On such an analysis, there is no semantic value to assign to the verb phrase saw everyone at the point in the derivation in which it is first formed by the syntax (or at any other point in the derivation, for that matter). A directly compositional analysis, by contrast, is forced to provide a semantic value for any expression that is recognized as a constituent in the syntax. Thus if there are good reasons to believe that saw everyone is a syntactic constituent, then a directly compositional analysis must provide it with a meaning.

Clearly, then, whatever the naturalness and appeal of direct compositionality, it cannot be taken for granted. By no means do all respectable approaches adhere to direct compositionality. In fact, as noted above, any formalism fails to be directly compositional that postpones interpretation until a level
Introduction: Direct Compositionality

of Logical Form. This includes, of course, any theory that relies on Quantifier Raising, which is by far the dominant paradigm in the field today. The papers in this volume explore the correctness of the hypothesis of direct compositionality, arguing in some cases against it, and in some cases trying to show that apparent challenges to the hypothesis can be met. Moreover, the question of whether or not (and to what extent) direct compositionality can be maintained is relevant to a wide range of formal frameworks. Indeed, the papers in this volume discuss this hypothesis in the context of type-logical grammars (Barker, Dowty), variable-free approaches (Barker, Bittner, Dowty, Jacobson, Shan), and combinatory grammars that rely heavily on type-shifting (Jacobson, Winter). The question of whether or not interpretation is directly readable off the "surface" level or involves instead a certain amount of hidden material and/or reconstruction is addressed from different perspectives in Bhatt and Pancheva, Caponigro and Heller, Romero, and Sharvit. The papers also address a variety of empirical phenomena; often extending the discussion to domains which have not been "standard fare" in previous discussions (see especially Bittner and Potts). Although each of these papers bears directly on the feasibility of direct compositionality, the arguments given by the authors in this volume do not always favor direct compositionality. Our hope, then, is that this volume will help provide a better understanding of the issues and the trade-offs, rather than a resolution to the question.

In the remainder of the introduction we will first map out the position of direct compositionality within the modern theoretical landscape, and then provide more detailed descriptions of the specific contributions of the papers included in the volume.

1.2 The Organization of the Grammar

One way to appreciate the constraints imposed by direct compositionality is by outlining some of the views which have been taken in the literature for the last thirty-five years or so about the way in which the syntax and semantics interact. In so doing we hope to illustrate a bit more what sorts of systems are directly compositional, what sorts are not, and the latitude concerning syntactic operations which is feasible while still maintaining a theory of direct compositionality. Some of this discussion is elaborated on in Jacobson (2002).

Jacobson (2002) defines the notion of "strong direct compositionality" which (as opposed to weaker versions of direct compositionality) is really a claim about what sorts of syntactic operations are allowed. This strong direct compositionality (like all varieties of direct compositionality) claims that each syntactic expression has a meaning, but also imposes the following constraint on the syntax: there can be no reference to syntactic (or semantic) structure internal to a constituent—and so from this it follows that the only syntactic combination operation allowed is concatenation. Type-shifting is allowed. An example would be any context-free syntax with rule-to-rule interpretation (such as the fragment in Gaifman et al. 1978); one could perhaps augment this or a similar system using Hendriks' (1993) type-shifting system for handling quantifiers.

The papers in this volume discuss this hypothesis in the context of type-logical grammars (Barker, Dowty), variable-free approaches (Barker, Bittner, Dowty, Jacobson, Shan), and combinatory grammars that rely heavily on type-shifting (Jacobson, Winter). The question of whether or not interpretation is directly readable off the "surface" level or involves instead a certain amount of hidden material and/or reconstruction is addressed from different perspectives in Bhatt and Pancheva, Caponigro and Heller, Romero, and Sharvit. The papers also address a variety of empirical phenomena; often extending the discussion to domains which have not been "standard fare" in previous discussions (see especially Bittner and Potts). Although each of these papers bears directly on the feasibility of direct compositionality, the arguments given by the authors in this volume do not always favor direct compositionality. Our hope, then, is that this volume will help provide a better understanding of the issues and the trade-offs, rather than a resolution to the question.

In the remainder of the introduction we will first map out the position of direct compositionality within the modern theoretical landscape, and then provide more detailed descriptions of the specific contributions of the papers included in the volume.
Introduction: Direct Compositionality

There is another kind of direct compositional theory which one can imagine and which is close to (although not exactly like) the classical theory of Generative Semantics (see, e.g., McCawley 1970; Lakoff 1971). In this view we have a set of rules building expressions and in tandem assigning them a meaning—but there are syntactic rules which are allowed to “adjust” the syntactic structure before pronunciation. (This of course means that the grammar will indeed need rules in the syntax referring to the internal structure of expressions because such rules are always stated on structures rather than just on strings: these rules are the classic “transformations” of classical transformational grammar.) While the rules adjusting the structure do not “change meaning”, we can nonetheless associate them with a corresponding semantic operation (the identity function). The important part is that each expression still receives a meaning, and the rules computing the initial structure of the transformational derivation are like the rules in a strong direct compositional theory: they “build” syntactic expressions and assign each a meaning. As mentioned above, this is similar to—although not exactly—the theory of Generative Semantics: the reason for the difference is that Generative Semantics did not (at least in general) actually posit an explicit model-theoretic interpretation for its “Logical Forms” (the representations computed by the initial syntactic rules). Had it done so, however, it would have been a direct compositional theory in this sense.

But now let us compare all those with a different kind of view about the organization of the grammar—the view that we can call “Surface-to-LF”. This is like the Generative-Semantics view taken above, except it is “backwards”: in the first phase, syntactic rules build an uninterpreted complex structure, in a second phase syntactic rules adjust that structure to produce a Logical Form, and only then does the semantics apply (model-theoretically, bottom-up) to interpret the subexpressions of LF. This is the view which has been taken in much of the semantics literature associated with “Extended Standard Theory”, Government-Binding Theory, and the Principles and Parameters Theory (see, e.g., Chomsky 1970; May 1977), and is the view in the system which is codified in Heim and Kratzer (1998). Unlike all of the views above, this is not directly compositional: here there are expressions referred to in the syntax and which form the input to many syntactic operations which have no meaning; they are interpreted only “later” in the grammatical computation. So take, for example, the case of in-situ quantification, in which a generalized quantifier can be inserted into direct object position of a transitive verb despite having an incompatible semantic type, in which case semantic interpretation must wait until a later operation of Quantifier Raising produces an interpretable LF.
many others). However, the relationship between direct compositionality and variable-free analyses is indirect, and the two are not logically linked. The fragment in, for example, Montague’s PTQ relies heavily on variables and assignment functions and yet is directly compositional; meanings are computed at each step relative to an assignment function but are still computed in tandem with the syntactic composition. Of course empirical facts may lead to the conclusion that direct compositional analyses can be facilitated in a variable-free account, but the hypothesis of direct compositionality does not a priori commit one to the rejection of variables and assignment functions.

1.3.3 Type-Shifting

Much work discussing direct compositionality makes heavy use of type-shifting operators. Type-shifting is a natural tool for building directly compositional analyses, since type-shifting operators are always presented as having a precisely defined semantic effect, often in conjunction with a change of syntactic category. Of course, type-shifting is also routinely used in analyses that are not directly compositional, and in these analyses there generally is no associated change of syntactic category. But again, while empirical facts might suggest that direct compositional analyses can be facilitated with the use of type-shifting, the two are logically independent. While it is not easy to find a directly compositional analysis that does not rely on type-shifting in one form or another, Montague’s PTQ (yet again) provides a key example. Although it has now become common to think of the principle ultrafilter meaning for a proper name (where $[[\text{John}]]_\text{} = \text{JP}(\text{John})$) as derived from the individual $j$ through a process of type raising (as in, e.g., Partee and Rooth 1983), this was not the strategy taken in PTQ. There, all noun phrases, including proper names, uniformly denote generalized quantifiers without any shifting back and forth. And we might further note that one can always trade in a “type-shift” rule for a “silent expression” in the syntax whose meaning is such that it performs the type-shift function (these have generally gone under the rubric of empty or silent operators), and so type-shifting operations can be recast in this way if desired. The issue of type-shifting is taken up in much more detail in Winter’s contribution to this volume, which discusses some complex and empirically interesting relationships between the syntax and the semantics of type-shifting.

1.4 Descriptions of Individual Papers

Each of the papers in this volume carefully explores the predictions of direct compositionality within a different empirical domain. These domains include
Optionality, Scope, and Licensing

ESSLLI 2007 CD Version

Raffaella Bernardi and Anna Szabolcsi
Faculty of Computer Science, Free University of Bozen-Bolzano, Department of Linguistics, New York University

27th of June 2007

Abstract

This paper uses a partially ordered set of syntactic categories to accommodate optionality and licensing in natural language syntax. A complex but well-studied data set pertaining to the syntax of quantifier scope and negative polarity licensing in Hungarian is used to illustrate the proposal. The presentation is geared towards both linguists and logicians. The main ideas can be implemented in different grammar formalisms; in this paper the partial ordering on categories is given by the derivability relation of a calculus with residuated and galois-connected unary operators.

1 Introduction

Among the basic issues that all syntactic theories have to deal with are the following:

1. Expressions often specify not only the broad categorial status of the expressions that they combine with, but also a particular subcategory. E.g., while will be hungry combines with any noun phrase subject, are hungry requires one in the plural.

2. Some expressions are optionally present, but have a fixed position and are not iterable. Numerals are an example. So three black dogs must be categorially distinct from black dogs. It is remarkable that despite this fact determiners like those apparently recognize that they are getting the desired complement whether it is of the form dogs or of the form three dogs.

3. Some expressions that have a sentential status are nevertheless ungrammatical and need a licensor for some of their components. Mary drank any more wine with a negative polarity item is an example. Drank has found its arguments, but the addition of a decreasing expression is still required, as in Not that Mary drank any more wine or whether Mary drank any more wine.

A common way to approach (1) and (2) is to use Typed Feature Structures or other constraint-based grammar formalisms (Kaplan and Bresnan (1982), Uszkoreit (1986), Carpenter (1992), Pollard and Sag (1994), Morrill (1994), Dorre and Manandhar (1997), Baldrige (2002), and others). The basic idea is that each syntactic category is a partially ordered set of subcategories (or, each expression is characterized by a feature structure and feature structures are partialy ordered sets), and the combination of expressions requires subsumption, rather than identity, between the pertinent categories.

Bernardi (2002) extends the same idea to the licensing problem in (3). Informally, assume the following partially ordered set; the category labels are ad hoc speaking names:

\[ \langle \{ \text{Complete Sentence}, \text{Incomplete Sentence}, \text{Good-enough Sentence} \}, \leq \rangle \]

with the following ordering relation: Good-enough \( S \) \( \leq \) Incomplete \( S \) and Good-enough \( S \) \( \leq \) Complete \( S \). Crucially, Incomplete \( S \) \( \not\leq \) Complete \( S \). The rest of the grammar will ensure that Mary drank a glass of wine is a Good-enough \( S \), and Mary drank any more wine is an Incomplete \( S \). Whether and not that select for an Incomplete \( S \) and yield a Complete \( S \). Notice that they are free to combine with either one of our two specimens, given the ordering Good-enough \( S \) \( \leq \) Incomplete \( S \). On the other hand, whereas the ordering relation tells us that Mary drank a glass of wine, by itself, is not only Good-enough \( S \) but also Complete, Mary drank any more wine must combine with whether or not that to be part of a Complete \( S \). Whether and not that act as licensors precisely because they can take a complement that is not Good-enough on its own and turn it into a Complete \( S \). - This type of account generalizes to other licensing relations.

Bernardi develops her proposal in a type-logical setting, relying specifically on innovations in Kurtonina and Moortgat (1995). In type-logical grammar, categories are labeled with logical formulae. Then, each partially ordered set of subcategories is a set of formulae with a derivability relation on it. Thus the exotic-looking (4) is traded for the more austere (5).

\[ \langle \{ \text{Complete \( S \)}, \text{Incomplete \( S \)}, \text{Good-enough \( S \)} \}, \leq \rangle \]

(4) \( \langle \{ \text{Complete Sentence}, \text{Incomplete Sentence}, \text{Good-enough Sentence} \}, \leq \rangle \)

(5) \( \langle \{ \psi_1, \psi_2, \psi_3 \}, \vdash \rangle \), where \( \vdash \) is the derivability relation of a particular calculus

With category labels as formulae, the ordering sheds its ad hoc character.

A technically novel feature of Bernardi’s use of this apparatus is to produce a "multi-layered" set of categories. To see how this is motivated by the linguistic problem at hand, consider the fact that Mary drank a glass of wine and Mary drank any more wine only differ in that the second one needs a licensor. In all other respects they are built in the same way, and moreover Mary drank a glass of wine happily occurs in all the larger environments that contain a licensor for Mary drank any more wine. This means that the whole partially ordered set of categories in the grammar has to be duplicated, so to speak, but with
a systematic subsumption relation between the categories of expressions without polarity items (the Good-enough ones) and those of comparable expressions with polarity items (the Incomplete ones). Natural languages exhibit a variety of different licensing relations, some of them local and some of them not, and so they necessitate the multiplication of smaller or larger partially ordered sets of categories. It is desirable, then, to have a systematic way to create multiple "layers" and subsumption relations between them. Representing syntactic categories as formulae of an appropriate calculus achieves this goal by deriving one layer from another.

This paper presents the above approach against the backdrop of a rather complex empirical domain, the highly constrained scope interaction between quantifiers and negative operators in Hungarian. A smaller set of English quantifier scope data was treated in this way in Bernardi (2002), based on Beghelli and Stowell’s (1997) observations. This is an empirically new domain as compared to those traditionally treated in the literature with partially ordered sets of categories. The switch from English to Hungarian here is motivated by the fact that Hungarian offers a richer set of data but, at the same time, a more transparent one. The surface left-to-right order of quantifier phrases in Hungarian largely mirrors their scope order and thus the language makes the syntax of scope directly observable. Another important property of Hungarian is that linear order is determined not by grammatical function but by quantifier class (group denoting, distributive, counting, negative concord, etc. quantifiers). In terms of Minimalism (Chomsky 1995), in our case study quantifier scoping represents the checking of interpretable features in overt syntax.

This paper has four goals.

(A) Present Bernardi’s theory in a way accessible to linguists whose home theory is not type-logical grammar but either some Typed Feature Structure grammar or Minimalism. This joint project has also led to some extensions of the logical aspects of Bernardi (2002); the extensions will be folded in.

(B) Argue in detail that the use of a partially ordered set of categories offers a natural solution to the problems of optionality and licensing. The optionality part of the argument will be familiar to TFS readers but novel with respect to Minimalism; the licensing part is generally novel.

(C) Illustrate and test the working of the theory with empirical data pertaining to Hungarian quantifiers, drawing from Szabolcsi (1997), Brody and Szabolcsi (2003), and other literature that is highly compatible with Beghelli and Stowell’s approach to quantifier scope in English.

(D) Illustrate how the same ordering among categories could be defined using a calculus with features and Boolean connectives.

2 The grammar

2.1 Proof theoretical approach

As was mentioned in the introduction, the central idea to be explored here is that the set of syntactic categories is partially ordered, as in Kaplan-Bresnan (1982) Uszkoreit (1986), Carpenter (1992), Pollard and Sag (1994), Morrill (1994), Dörre and Manandhar (1997), Baldridge (2002), and other work. In some empirical domains the ordering is straightforwardly determined by the nature of the features involved, e.g. Plural Noun Phrase ≤ Noun Phrase. If no such ordering is obvious, it can be stipulated and still have an advantage over theories that assume no such ordering. However, Bernardi (2002) showed that certain commonalities of the patterns can be elegantly captured if the ordering is given by the derivability relation of a particular calculus. So we present the whole proposal in a proof theoretical format, although its specific advantages will only come into play later.

The proof theoretical approach to syntax presents syntax as a calculus, where the syntactic category labels assigned to lexical expressions are the axioms and the syntactic category labels derived for complex expressions – sentences among them – are the theorems.1 Our case study focuses on the behavior of quantifiers and negation, so we do not need to talk about how simple sentences are built. We only work with two kinds of expressions: sentences and sentential operators. The discussion will be framed within a novel version of the Lambek calculus. Lambek’s (1958) idea was to take syntactic category labels to be formulae of a propositional calculus with just material implication →, notated in categories / \. The pertinent inference rules of this simple calculus are the ones corresponding to modus ponens (the elimination of \(\rightarrow\)) and conditionalization (the introduction of \(\rightarrow\)).

An expression of category \(y/x\) followed by an expression of category \(x\) forms an expression of category \(y\). Compare modus ponens:

\[
\begin{align*}
x & \rightarrow y \\
x & \vdash y
\end{align*}
\]

If an expression is of some category \(z\) such that when followed by another expression of category \(x\), they form an expression of category \(y\), then the category \(z\) derives \(y/x\). Compare conditionalization:

\[
\begin{align*}
x & \\
\text{if } z \text{ then } x & \vdash y
\end{align*}
\]

1Syntactic categories are sets of expressions: those expressions that belong to the given category. VP, c/v, etc. are category labels: names for such sets. This distinction is important to bear in mind when one talks about categories as formulae, although the literature is often sloppy about it and we will also take the liberty to sometimes use the term “category” to refer to a category label.
The category of sentences will be informally notated as $s$ and that of sentential operators as $s/s$. The argument category is under the slash; i.e. $s/s$ is $s_{\text{value}}/s_{\text{argument}}$. The following Hungarian examples show the working of negation, $\text{nem}$ as $s/s$:

\begin{itemize}
  \item $\text{nem ‘not’}$ is an expression of category $s/s$.
  \item $\text{é g a há z}$ \text{burns the house}
    \begin{itemize}
      \item ‘The house is on fire’
    \end{itemize}
  \item $\text{nem é g a há z}$ is an expression of category $s$.
    \begin{itemize}
      \item ‘The house is not on fire’
    \end{itemize}
\end{itemize}

(6) a. $\text{is hungry}$ requires one in the singular. This can be handled with the following assumptions:

\begin{itemize}
  \item $\text{she} \in \text{Singular Noun Phrase}$, $\text{they} \in \text{Plural Noun Phrase}$
  \item $\text{will be hungry}$ wants an argument of category Noun Phrase
  \item $\text{are hungry}$ wants an argument of category Plural Noun Phrase
  \item $\text{is hungry}$ wants an argument of category Singular Noun Phrase
  \item Singular Noun Phrase $\leq$ Noun Phrase
  \item Plural Noun Phrase $\leq$ Noun Phrase
\end{itemize}

Will $\text{be hungry}$ can take either $\text{she}$ or $\text{they}$ as an argument. While these expressions do not receive the category label Noun Phrase in the lexicon, the grammar tells us that their lexical category labels derive Noun Phrase. In general,

(8) An expression of category $A/C$ combines with an expression of category $B$ as an argument iff $B$ derives $C$.\footnote{From our perspective scope taking can be reduced to functional application. A quantifier phrase like every man that denotes a generalized quantifier of type $\langle e, t \rangle$ syntactically combines with its scope by Montague’s Quantifying-in rule or some reincarnation thereof. Its interaction with other operators is determined by what sentential (sub)category it can be quantified into (its argument category), and what sentential (sub)category it feeds to higher operators (its value category). The fact that it binds an individual variable, whereas not does not, is immaterial from this perspective. Therefore the ordering directly pertains to just the sentential categories and is inherited by the generalized quantifier categories. To make the discussion less tied to Categorial Grammar or to the irrelevant details of Hungarian, the main body of this paper abstracts away from how atomic sentences are built (although a sample derivation of a full sentence is provided in Appendix C). In this simplified context we schematically talk of the category of operators, including quantifier phrases as $s_{\text{val}}/s_{\text{arg}}$. As will be seen below, this is especially straightforward in Hungarian, where quantifier phrases in the preverbal field line up in their scopal order, rather than stay in subject, object, etc. position as in English. The generalization in (8) simply extends to quantification, as in (9).}

(7) If $A$, $B$ are formulae serving as category labels, and in the calculus $A$ derives $B$, then every expression that belongs to category $A$ also belongs to category $B$.

That is, the derivability relation between category labels corresponds to a subset relation between the sets of expressions bearing those labels. The fact that $p$ derives $\Box p$ will correspond to the fact that every expression that belongs to the (sub)category $p$ also belongs to the (sub)category $\Box p$.

\[
A \rightarrow B \iff \left\{\text{expressions labeled } A\right\} \subseteq \left\{\text{expressions labeled } B\right\}
\]

2.2 Functional Application, Scope, and Intervention

Recall problem (1) mentioned in the introduction: the verb phrase will be hungry combines with any noun phrase as its subject, but are hungry requires one in the plural, and is hungry requires one in the singular. This can be handled with the following assumptions:

\begin{itemize}
  \item $\text{she} \in \text{Singular Noun Phrase}$, $\text{they} \in \text{Plural Noun Phrase}$
  \item $\text{will be hungry}$ wants an argument of category Noun Phrase
  \item $\text{are hungry}$ wants an argument of category Plural Noun Phrase
  \item $\text{is hungry}$ wants an argument of category Singular Noun Phrase
  \item Singular Noun Phrase $\leq$ Noun Phrase
  \item Plural Noun Phrase $\leq$ Noun Phrase
\end{itemize}

\begin{itemize}
  \item $\text{she} \in \text{Singular Noun Phrase}$, $\text{they} \in \text{Plural Noun Phrase}$
  \item $\text{will be hungry}$ wants an argument of category Noun Phrase
  \item $\text{are hungry}$ wants an argument of category Plural Noun Phrase
  \item $\text{is hungry}$ wants an argument of category Singular Noun Phrase
  \item Singular Noun Phrase $\leq$ Noun Phrase
  \item Plural Noun Phrase $\leq$ Noun Phrase
\end{itemize}

Will be hungry can take either she or they as an argument. While these expressions do not receive the category label Noun Phrase in the lexicon, the grammar tells us that their lexical category labels derive Noun Phrase. In general,

(8) An expression of category $A/C$ combines with an expression of category $B$ as an argument iff $B$ derives $C$.\footnote{From our perspective scope taking can be reduced to functional application. A quantifier phrase like every man that denotes a generalized quantifier of type $\langle e, t \rangle$ syntactically combines with its scope by Montague’s Quantifying-in rule or some reincarnation thereof. Its interaction with other operators is determined by what sentential (sub)category it can be quantified into (its argument category), and what sentential (sub)category it feeds to higher operators (its value category). The fact that it binds an individual variable, whereas not does not, is immaterial from this perspective. Therefore the ordering directly pertains to just the sentential categories and is inherited by the generalized quantifier categories. To make the discussion less tied to Categorial Grammar or to the irrelevant details of Hungarian, the main body of this paper abstracts away from how atomic sentences are built (although a sample derivation of a full sentence is provided in Appendix C). In this simplified context we schematically talk of the category of operators, including quantifier phrases as $s_{\text{val}}/s_{\text{arg}}$. As will be seen below, this is especially straightforward in Hungarian, where quantifier phrases in the preverbal field line up in their scopal order, rather than stay in subject, object, etc. position as in English. The generalization in (8) simply extends to quantification, as in (9).}

(9) A quantifier phrase (operator) of category $s_{\text{val}}/s_{\text{arg}}$ takes immediate scope over a syntactic domain of category $s_j$ iff $s_j$ derives $s_{\text{arg}}$.\footnote{The reader interested in an in-depth formal presentation of the treatment of QPs in categorial type logic is referred to Moortgat (1997).}
Crucially to the solution of problems (2) and (3), recognizing the derivability (inclusion) relations between categories offers an account of when the intervention of some sentential operator $OP_k$ between $OP_a$ and $OP_e$ is optional or obligatory.\(^5\)

Given the total ordering and the category assignment as in (10), the left-to-right order of the operators follows:

$$\begin{align*}
OP_3 &> OP_2 > OP_1 \\
s3/s3 &> s2/s2 > s1/s1
\end{align*}$$

By transitivity, $s1 \rightarrow s3$, viz. $OP_1$ can also scope over $OP_1$ directly. $OP_2$ is optional.

$$\begin{align*}
OP_3 &> OP_1 \\
s3/s3 &> s1/s1
\end{align*}$$

If the ordering relation is not total but partial, as in Figure 1, then in (11) $OP_3$ may only scope over $OP_3$ if $OP_3$ intervenes and bridges between $OP_3$’s value category and $OP_3$’s argument category. For instance, given the derivability relations in Figure 1 and $OP_3$ and $OP_3$ of category $s4/s2$ and $s2/s$, respectively, $OP_3$ can precede $OP_3$ only if operator $OP_3$ of category $s4/s2$ mediates, because $s2 \not\rightarrow s4$.\(^6\)

![Figure 1: Partial order of sentential subcategories.](image)

Figure 1: Partial order of sentential subcategories.

Hence,

\(^5\)In what follows, $X > Y$ notates “$X$ precedes and/or scopes over $Y$”, $X \rightarrow Y$ notates “$X$ derives $Y$”. In this paper the $\leq$ notation is reserved for the pretheoretical, informal notion of an ordering relation.

\(^6\)The dot acts as a placeholder in the presence of syncategorematic operators.

The bridging between two categories that the ordering relation does not relate to each other is crucial for the account of many phenomena, among others, NPI-licensing.

### 2.3 Residuated Pairs of Connectives

Our framework is a Categorial Grammar (CG) known as Categorial Type Logic (CTL).\(^7\) It consists of (i) the logical rules of binary operators\(^8\) and (ii) the logical rules of unary operators.

#### 2.3.1 Residuated Binary Connectives

The rules of the binary operator / are the same as the introduction and elimination rules of the propositional calculus $\rightarrow$, see the standard deduction theorem:

$$\Gamma \vdash p \iff \Gamma \vdash q/p$$

In words: $\Gamma$ concatenated with $p$ belongs to the category $q$ if and only if $\Gamma$ belongs to the category $q/p$. The $\bullet$ indicates the concatenation of structures.

The relation above between the $\bullet$ and the / is known in algebra as the residuation principle. The $\bullet$ and the / form a residuated pair in the same way as addition and subtraction or multiplication and division do. Recall how one solves an algebraic equation like $3 \times x \leq 5$ by isolating the unknown $x$ using the law connecting $(\times, \div)$ and producing the solution $x \leq \frac{5}{3}$. The law connecting these two binary (residuated) operators says:

$$x \times y \leq z \iff x \leq \frac{z}{y}$$

In CTL, such a pair of operators is used to put together and take apart linguistic expressions as sketched in Section 2.1.

It follows from residuation that $A/C$ is order reversing (with respect to category selection) in its argument position ($C$), and order preserving in its value position ($A$). If $B$ derives $C$, then $A/C$ is also happy with $B$ as an argument; if $A$ derives $D$, then $A/B$ also counts as $D/B$. This is formally represented by the inferences below.

$$\begin{align*}
B &\rightarrow C \\
A/C &\rightarrow A/B \\
A &\rightarrow D \\
A/B &\rightarrow D/B
\end{align*}$$

\(^7\)Alternative names are Type Logical Grammar, see for instance Morrill (1994), and Multi-modal Categorial Grammar (Moortgat and Oehrle 1994).

\(^8\)The binary operators are $\backslash, \bullet, /$. For ease of exposition we will focus only on / and $\bullet$. 

2.3.2 Residuated Unary Operators

The residuated unary operators, to which we now turn, will serve to create a fine grained partial order of categories. We show that the partial order among the sentential subcategories required to control scope and word order can be encoded as the derivability relation driven by residuated unary operators.

Kurtonina and Moortgat (1995) further explored the space of the Lambek calculus by exploiting unary operators inspired by tense logic. The idea of this line of research is to take the minimum logic, i.e. a logic characterized by those properties that are at the core of any logic (namely, transitivity of the derivability relation and upward/downward monotonicity of operators) as a starting point to analyze linguistic universals, and then extend its language so as to increase its expressivity and analyze linguistic structures and cross-linguistic variation.

The past possibility and future necessity operators of tensed modal logic have just the core properties. That is, they obey the algebraic principle of residuation introduced above:

\[ \text{PastPoss} A \rightarrow B \text{ iff } A \rightarrow \text{FutNec} B \]

Following Kurtonina and Moortgat (1995), we fashion our residuated pair of unary operators after these and notate them as \( \Diamond \) and \( \Box \). These symbols are hijacked for typographical convenience and must not be confused with the standard modal operators, which form a pair of duals and not a pair of residuals.

Thus, in the notation to be used below:

\[ \Diamond A \rightarrow B \text{ iff } A \rightarrow \Box B \]

The properties below follow from (13); see details in Appendix A

1. \( \Diamond \Box A \rightarrow A \) \hspace{1cm} [Unit]
2. \( A \rightarrow \Box \Diamond A \) \hspace{1cm} [Co-unit]
3. \( \Diamond A \rightarrow \Diamond B, \text{ if } A \rightarrow B \) \hspace{1cm} [\( \Diamond \) upward monotonic]
4. \( \Box A \rightarrow \Box B, \text{ if } A \rightarrow B \) \hspace{1cm} [\( \Box \) upward monotonic]
5. \( \Diamond \Box A \rightarrow \Box \Diamond B, \text{ if } A \rightarrow B \) \hspace{1cm} [\( \Diamond \Box \) upward monotonic]
6. \( \Box \Diamond A \rightarrow \Diamond \Box B, \text{ if } A \rightarrow B \) \hspace{1cm} [\( \Box \Diamond \) upward monotonic]

In this paper, we use the \( \Diamond \) and \( \Box \) operators as decorations on sentential categories. The derivability relation among decorated categories defines a partial order. As in Bernardi (2002), that partial order will be used to express the fine grained partial ordering among sentential categories that is necessary to capture the differential scoping abilities of quantifier phrases. Section 4 will illustrate this with Hungarian material.

Figure 2 illustrates the derivability relations within one small set of decorated categories. It exhibits all the derivability relations that exist within the given set of categories (although of course these categories are derivable from infinitely many others and derive infinitely many others). The claim that all and only the indicated derivability relations hold within this set could be proven model theoretically. But given the simplicity of categories the reader can easily verify it by consulting the theorems above and Appendix A. The colors have no status in the calculus and only serve to help the reader with keeping track of how one category is derived from another.

\[ \text{Figure 2 HERE} \]

2.3.3 Multiple Modes for Unary Operators

The last feature of the calculus to be introduced here is the availability of multiple modes for the unary operators. There are various linguistic applications of multimodality in CTL, some of them quite different from our own application.\(^\text{12}\) Suppose we have just two modes, one notated with empty \( \Diamond \), \( \Box \) and the other with filled \( \bullet \), \( ■ \). The two modes will add further flexibility to the logic whose derivability relation formalizes the partial ordering of sentential categories.

The consequences of residuation listed above hold for unary operators of the same mode. Distinct modes do not mix, i.e. there is no law that derives anything from \( \Box \bullet A \). On the other hand, the same Co-unit property that gives \( \Diamond \Box s \rightarrow \Box \Diamond \Box s \) also derives \( \bullet \Box s \rightarrow \Box \bullet \Box s \). Likewise, the Unit property that gives \( \Diamond \Box \Box s \rightarrow \Box \Box s \) also produces \( \bullet \Box \Box s \rightarrow \Box \Box s \). This means that several alternative paths may be constructed from one element of the partially ordered set to another: one involving only operators in the empty mode, another involving both empty and filled ones, etc. The diagram below illustrates one way of adding operators in the filled mode to the set in Figure 2. The linguistic role of the two modes is discussed in Section 4, where the diagram will be repeated.

\[ \text{Figure 3 HERE} \]

\(^{12}\) See Heylen (1999) for a detailed study of the use of unary operators to encode feature structure information.
Figure 2: Derivability relations among a few operators

Figure 3: Derivability in the space of Figure 2 with two modes
3 Quantifier order and scope in Hungarian, and some questions that they raise

3.1 A bird's eye view

The syntax of scope in Hungarian will serve as our testing ground. Our interest is not in the Hungarian operators per se, but rather in the fact that (i) they illustrate a case where the surface syntactic distribution of expressions depends on their interpretable features, and (ii) they are numerous enough to give rise to rather complex interactions.

To a significant extent, the syntax of scope is the syntax of Hungarian: the left-to-right order of operators in the preverbal field unambiguously determines their scopal order. Another remarkable property is that the possible orders are determined by quantifier class and not by grammatical function. Thus, the examples in (14) illustrate the fact that a distributive universal must precede a counting quantifier with kevés 'few', irrespective of which is the subject and which is the direct object and, given their order, the former inescapably outscopes the latter.\(^{13}\)

\[\text{(14) a. Minden doktor kevés filmet láttott.} \]
\[\text{every doctor-nom few film-acc saw} \]
\[\text{`Every doctor saw few films', viz. every} \text{_{\text{subject}}>few_{\text{object}}}. \]

\[\text{b. Minden filmet kevés doktor láttott.} \]
\[\text{every film-acc few doctor-nom saw} \]
\[\text{`Few doctors saw every film', viz. every} \text{_{\text{object}}>few_{\text{subject}}}. \]

\[\text{c. *Kevés doktor minden filmet láttott.} \]
\[\text{few doctors-nom every film-acc saw} \]
\[\text{`Few doctors saw every film', viz. every} \text{_{\text{subject}}>few_{\text{object}}}. \]

A common way to capture these facts has been to assume that operators move into designated positions in the manner of wh-movement, and their left-to-right order translates into a quantifying-in hierarchy. This assumption differs from Fox-Reinhart style Interface Economy, according to which quantifier scope is assigned by Quantifier Raising, an adjunction operation that applies only when it makes a truth conditional difference (Fox 1999; Reinhart 2006). Notice, however, that both Fox and Reinhart concern themselves with a case where scope assignment has no effect on surface constituent order: covert scope shifting in English. The case of Hungarian is different: quantifier phrases in Hungarian occur in the positions to be discussed below irrespective of whether this has a disambiguating effect. Even if one were to ignore the fact that left-to-right order determines interpretive order, the syntax of Hungarian would have to account for the fact that certain word orders are grammatical and others are not.

The following diagram illustrates three of the relevant positions with their characteristic inhabitants. For space reasons only the determiners are included. Some though not all quantifier phrases may occur in more than one position and their interpretations vary accordingly. An example in (15) is sok 'many'. When sok ember 'many men' occurs in the "counter" position, which is the only possible position of hatnál több ember 'more than six men', it supports both distributive and collective readings, but when it occurs in the "distributive" position, which is the only possible position of minden ember 'every man', the collective interpretation is not available. Such semantic matters are discussed in detail in Szabolcsi (1997).

\[\text{(15) \quad }\text{ ReferentialIP}\]
\[\text{"topics"} \quad \text{\quad Dist(rictutive)IP}\]
\[\text{val-\text{\_}scene} \quad \text{\quad CountingIP} \quad \text{\quad AgrP}\]
\[\text{"distributives"} \quad \text{\quad "counters"} \quad \text{\quad hat/nok 'six/many'} \quad \text{\quad Vinite}\]
\[\text{kevés 'few'} \quad \text{\quad minden 'every'} \quad \text{\quad hatnál több 'more than six'} \]

The filling of each of these positions is optional; however, all the positions can be filled simultaneously. RefP and DistP are recursive (cf. the Kleene stars), subject to the same "left-to-right order determines scope" rule. Preverbal operators normally outscope all postverbal ones (those occur in the … part of (15)). Therefore, a counter gets a chance to outscope a distributive quantifier if the latter occupies a postverbal position.\(^{14}\)

Postverbal quantifier order is virtually free. É. Kiss (1998) and Brody and Szabolcsi (2003) argue however that the sequence of operator positions observed preverbally reiterates itself in the postverbal field. The impression of postverbal order freedom is due to the fact that of the inflectional heads that separate the operators, only the first (in this case, AGR) determines its own order.

\(^{13}\)We draw directly from the results of Szabolcsi (1997, 1981), Brody and Szabolcsi (2003), É. Kiss (1987, 1991, 1998, 2002), Puskas (2000), Horvath (2000, 2006), Hunyadi (1999), and Surányi (2003).\(^{14}\)Inverse scope, i.e. one that does not match left-to-right order and where, specifically, a postverbal operator outscopes preverbal ones, is possible in two cases: (i) with a postverbal specific indefinite, and (ii) with a postverbal distributive that bears primary stress. Neither of these is assumed to involve overt or covert operator movement and will not be further discussed in this paper. The wide existential scope of indefinites may be attributed to existential closure over choice functions à la Reinhart (1997). As regards primary stressed postverbal distributives, both É. Kiss (1996) and Brody and Szabolcsi (2003) argue in detail that they effectively occur in the highest DistP projection and their postverbal ordering is obtained using permutation rules that do not affect c-command and scope relations.
operator sequences – Agr(eement), T(ense), etc. – only the highest is visible: the one that hosts the finite verb. Therefore two adjacent operators in the postverbal field need not belong to the same operator sequence and need not conform to the sequence-internal hierarchy. The overt or covert inflectional heads play the kind of beneficial mediating role that was described in Section 2.2.15

3.2 Total order?

An important fact about the operators reviewed above is that they can all co-occur. Adding focus, negation, and question words to the mix raises new questions about how expressions, and their categories, can be ordered.

First consider focus. Hungarian is one of those languages that have a reflex of focussing in surface syntax. Counting quantifiers and foci (emphatic focus, identificational focus, and phrases modified by csak ‘only’) are complementary in the immediately preverbal position. As they never co-occur, no left-to-right ordering can be established between them:

\[(16) \text{topic} \succ \text{distributive} \succ \text{counter} \succ (\text{verb} \ldots) \succ \text{focus}\]

Next consider negation. The preverbal field may contain two distinct instances of sentential negation (nem), to be dubbed as hi-neg and lo-neg when the distinction is necessary. The two happily co-occur and, naturally, do not cancel out, when an appropriate third party intervenes. The postverbal field houses no negation. See Koopman and Szabolcsi (2000, Appendix B).

Even putting aside negative polarity items, operators come in different flavors as regards their ordering constraints with respect to negation. Negation may follow a focus or a counter, and it may precede a focus, though not a counter:

\[(17) \begin{align*}
& \text{(*hi-neg)} \succ \text{counter} \succ \text{lo-neg} \succ (\text{verb} \ldots) \\
& \text{hi-neg} \succ \text{focus} \succ \text{lo-neg} \succ (\text{verb} \ldots)
\end{align*}\]

Distributive universals cannot scope immediately above negation, nor for that matter immediately below it; 16

15The analysis of the postverbal field is a matter of some disagreement, see Surányi (2003). The postverbal facts will play little role in this paper: they are mentioned only to enable us to provide a concrete sample derivation in Appendix C.

16Examples with nem minden fiú ‘not every boy’ contain phrase internal negation and not a hi-neg preceding the quantifier minden fiú in one of its otherwise legitimate positions. The critical data that show this involve order interaction with verbal particles. The verbal particle fel-’up’, etc. precedes the verb unless the next element to the left is negation, or a focus, or a counter. With non-negated minden-phrases the only possibility is (i). However, nem minden-phrases require the verbal particle to follow the verb, as in (ii). Thus nem minden-phrases represent a separate quantifier class, cf. also (19). (i) Minden fiú fel-ébredt. every boy up-woke
(II) Nem minden fiú ébredt fel. not every boy woke up

17Hungarian is a so-called strict negative concord language. Negative concord items (NC) are interpreted as universals, following Szabolcsi (1981), Giannakidou (2000), Puskas (2000).
counters cannot (unless they have a contrastive component) is one argument for the distinct categories analysis. However, it could be accommodated in (22) by adding that Hi-Neg requires its complement to carry the feature [contrast], and not all [pred] phrases have [contrast]. *Ki ‘who’ will have [pred] but its PredP is specified not to be the complement of a head with [neg] or [dist].

Another supplementary assumption is that distributive universals, distributive existentials, and NC universals all have a [dist] feature and are thus headed for the specifier of a Dist head, but they are marked differently as to what features the complement of Dist should carry. NC universals require that the closest head below have a [neg] feature; distributive universals require that the same head not have [neg]; distributive existentials and expressions with [topic] are not marked in this regard. Not only nem ‘not’ has [neg], but also nem mindenki ‘not everyone’, and senki ‘no one, NC’ come with a [neg] feature that they transmit by specifier-head agreement. The treatment of nem mindenki itself remains difficult. One might say that it has [dist] and [neg] features and additionally requires that the complement of the Dist head have [neg] or [contrast] or [bare agr], where [bare agr] is an ad hoc feature to pick out the verb separated from its particle.

This will suffice to show both that the total order in (22) could be maintained and what kind of cost it would incur. A description using a total order of categories is possible, but the result does not look very Minimalist. Put in general terms, this description preserves the illusion of a total order by not assigning a status to the featural restrictions within the theory of syntax. This might be fine if all the restrictions follow from the semantics of the expressions involved. The restrictions on question words probably do, but it is not obvious that the same holds for all the other restrictions.

3.3 Optionality

The Hungarian data highlight another fundamental question. As was noted in 3.1, the presence of all the operators discussed in this section is optional. Consider:

(23) Tudom, hogy [ReP az emberek [AgSP láttak]].
    know-1sg that the men saw-3pl.1sg
    ‘I know that the men saw me’

(24) Tudom, hogy [DistP minden ember [AgSP láttak]].
    know-1sg that every man saw-3sg.1sg
    ‘I know that every man saw me’

(25) Tudom, hogy [AgSP látták].
    know-1sg that saw-2sg.1sg
    ‘I know that you saw me’

These examples raise the optionality problem (2) of Section 1. The complementizer head hogy ‘that’ is apparently equally happy to recognize ReP, DistP, and AgrSP as suitable arguments. Likewise, Ref selects for DistP, but it is equally happy with AgrSP, and so is Dist, which selects for PredP. How are the complement selection requirements of these heads (functional categories) satisfied?

The optionality problem is by no means specific for Hungarian; Hungarian just illustrates it on a large scale. Although it is no novelty in formalisms using partially ordered sets of categories or features, it does not seem to have received much attention in the Minimalist literature and we are not aware of a standard solution. In line with Cinque’s (1999) influential proposal that the sequence of functional categories is invariant and universal, one hypothesis might be that whereas the full sequence of categories in (22) is always present, the individual categories need not host lexical items in every sentence. This hypothesis would have been easy to accommodate in earlier, phrase structure rule based versions of generative syntax, but it is not in Minimalism, the mainstream format of the theory for over a decade. The problem is that in Minimalist Theory categorial structure is projected from the lexical items that make up the sentence: no lexical item, no category. So, one would need to postulate that for each optional head category there exists a “dummy lexical item”, which has no ability to attract a phrase to its specifier but suffices to project the phrasal category that satisfies the complement selection requirements of the head above it. Moreover, to accommodate the constraints discussed in connection with the total order (22), one would need to ensure that phrasal categories headed by dummies inherit the features of the next phrasal category below them that is headed by a real lexical item.

The conclusion is the same as that of the previous subsection: a solution involving an invariant sequence of categories is in principle possible, but it does not look very Minimalist.

3.4 Partial order and derivability/inclusion relations: Two birds with one stone

As was pointed out in sections 1 and 2, the optionality problem receives a natural solution if expressions are not thought to have a unique category label but derivability/inclusion relations among categories are recognized. If the grammar recognizes the DistP −→ ReP (DistP ⊆ ReP) relation, and the complementizer head hogy ‘that’ does not look for a complement specifically labeled as ReP but accepts any category that derives ReP, then the grammaticality of (24) no longer comes as a surprise; and similarly for the other examples.

Thus the solution to the optionality problem points to a partially, not totally, ordered set of categories. This suggests that the effort to create a total order in (22) and supplement it with an extra-theoretical device, a set of featural constraints as detailed above, is unnecessary. Those same constraints can be expressed as finer details of the partial ordering. The next section demonstrates how this works in CTL, and Section 7 recasts it using features.

The same conclusion that syntactic categories should be partially, rather than totally, ordered was reached in Nilsen (2002, 2004) within Minimalist syn-
tax, based on somewhat different data. Nilsen’s empirical arguments come from the distribution of adverbs in Norwegian. He observes, contra Cinque (1999), that Norwegian adverbs by default occur in any order; whatever ordering restrictions one finds follow from the fact that the individual adverbs are often ordered with respect to negation.

4 A mini-grammar of operators in Hungarian

Within the multi-modal categorial type logic introduced in Section 2, capturing the partial order of the most important Hungarian operators requires the use of the portion of the logical space exhibited in Figure 3.

For easier reference each sentential category is given a number. The category $s_1$ is assigned to sentences whose initial element is an inflected verb.

Prior to locating the Hungarian operators in this space, let us draw attention to the two modes in Figure 3. The basic mode is represented with empty boxes and diamonds. The filled boxes and diamonds can be seen to add an alternative dimension to some parts of the system; to highlight this, the sentential categories that involve filled modes have the numbers of the corresponding categories in the empty mode plus an asterisk. So, for example, parallel to $\Diamond \Box p = (s_1)$ is $\Box \Diamond p = (s_1^*)$ and parallel to $\Box \Diamond \Box p = (s_4)$ is $\Diamond \Box \Diamond p = (s_4^*)$. The two alternative dimensions merge where $s_n$ and $s_n^*$ derive the same category. $s_1$ and $s_1^*$ both derive $p = (s_2)$, and $s_4$ and $s_4^*$ both derive $\Diamond p = (s_3)$, etc.

The categories based on the filled mode will be used to capture the behavior of those operators – negative concord items – that must scope immediately above negation or another negative concord item. So $s_1$ is the category of basic affirmative sentences and $s_1^*$ the category of basic negative sentences. The argument category of lo-neg is $s_1$ and its value category is $s_1^*$, yielding the functor category $s_1^*/s_1$.

---

Figure 5 adds operator expressions to this diagram, but to reduce clutter, those categories that are not immediately relevant are trimmed off, and the decorated categories are removed. Operator expressions have the category $val/arg$ and are represented in Figure 5 as curved arrows pointing from the argument category to the value category. The curved arrows are labelled either with the informal names of the classes (topic, counting quantifier, focussed XP, hi-neg, lo-neg) or with a representative member (who, no one-NC, everyone, XP too, many people, not everyone).

---

These asterisks are not to be confused with the Kleene stars that indicate iteration.
Expressions that do not care about scoping directly above negation have their argument categories on the $s_1 - s_2 - s_3 - s_8 - s_9$ track; those that must not scope directly above negation on the $s_1 - s_4 - s_7$ track; and those that must scope directly above negation or another operator of their own kind (negative concord items) on the $s_1^* - s_4^* - s_7^*$ track.

To see how Figure 5 captures other data reviewed in subsection 3.1, recall that counters ($s_7/s_2$) and foci ($s_4/s_2$) do not co-occur and are therefore not ordered with respect to each other. Notice that neither $s_4$ nor $s_7$ derives $s_2$. The reader is invited to find other examples.

Whenever the value category of an expression derives the argument category of another, the predicted results are grammatical, although more than five or six operators preceding the verb may sound crowded. Consider just one example:

```
Kati a legtöbb napon mindenivel sok újságot
Kate on most days with everyone many pieces of news.acc
rosszindulatból nem közölt.out of malice not shared.3sg
```

‘On most days for every person there were many pieces of news such that it was out of malice that Kate did not share those pieces of news with that person’

Although this paper is concerned only with the operator categories, for concreteness Appendix C spells out the complete Natural Deduction style derivation of a simple sentence, including inflectional categories. This derivation will show how the optional operators co-exist with obligatory elements in a grammar. Some comments are added pertaining to innovations in the treatment of syntactic phenomena, but the syntactic framework is not introduced there. Readers not familiar with Moortgat (1997) may want to skip Appendix C.

5 The logic of licensing

5.1 Salvaging ungrammatical categories

We now turn to the licensing problem discussed in (3) in Section 1. Bernardi (2002) develops a proposal for negative polarity item (NPI) licensing,\(^{19}\) which we can take to be the representative of licensing in general. That is, from our perspective the following two structures are alike, and whatever we say about NPI-licensing carries over to the licensing of inversion, for example:

\(^{19}\)Negative polarity items are expressions like ever, any, and others that must occur within the immediate scope of a monotonically decreasing operator called the licensor. See further details in the next section.
That the NPI needs to be licensed means that a structure containing the NPI is ungrammatical unless it is within the scope of an appropriate operator. Notating the category of an NPI as \( \text{val}_{\text{NPI}}/\text{arg} \), \( \text{val}_{\text{NPI}} \) does not derive \( s \), the category of complete grammatical sentences: \( \text{val}_{\text{NPI}} \) is an “ungrammatical category”. The fact that being within the immediate scope of a licensor salvages the structure means that the licensor is capable of bringing it back to the grammatical fold: the value category of the licensor is a grammatical one. More surprising is the fact that the licensor can scope immediately above the structure containing the NPI: it shows that \( \text{val}_{\text{NPI}} \) does not derive \( s \) iff \( \text{arg}_{\text{lic}} \), the category of complete grammatical sentences: \( \text{val}_{\text{NPI}} \) is an “ungrammatical category” too. Example:

\[
\begin{align*}
\text{seldom} & > \text{anything} > \text{saw} \\
\text{val/arg}_{\text{lic}} & > \text{val}_{\text{NPI/arg}} > \text{arg}
\end{align*}
\]

On the other hand, the licensor does not require for there to be an NPI within its immediate scope. This means, in turn, that \( \text{arg}_{\text{lic}} \) is also derived by various grammatical categories:

\[
\begin{align*}
\text{seldom} & > \text{everything} > \text{saw} \\
\text{val/arg}_{\text{lic}} & > \text{val}_{\text{lic/arg}} > \text{arg}
\end{align*}
\]

So, we have:

\[
\begin{align*}
\text{val}_{\text{NPI}} \text{ (ungrammatical)} & \not\rightarrow s_n \text{ (grammatical)} \\
\text{arg}_{\text{lic}} \text{ (ungrammatical)} & \not\rightarrow s_n \text{ (grammatical)} \\
\text{(26)} & \text{val}_{\text{NPI}} \text{ (ungrammatical)} \rightarrow \text{arg}_{\text{lic}} \text{ (ungrammatical)} \\
\text{val}_{\text{salv}} \text{ (grammatical)} & \rightarrow \text{arg}_{\text{lic}} \text{ (ungrammatical)}
\end{align*}
\]

It is in principle possible to set up the partial order of syntactic categories in such a way that ungrammatical categories reside in a “blind alley” that satisfies the requirements in (26). For example, \( \square\square\square\square\square\square p \) is derivable from all the categories used in Figure 2 but does not derive any one of them. \( \square\square\square\square\square\square p \) might be designated as “the ungrammatical category”. Given however the variety of licensing relations the grammar has to accommodate, this solution would probably be ad hoc and unable to capture finer patterns.

5.2 The logic of ungrammatical categories

5.2.1 Galois-connected Unary Operators

Bernardi (2002) proposes a systematic way to encode the kind of derivability relations described in (26) using unary Galois operators. These were first introduced into CTL in Areces and Bernardi (2001) inspired by Dunn (1991), Goré (1998). These authors show that the realm of minimum logic (i.e. the logic characterized by just the core properties of the transitivity of derivability and the monotonicity of the logical operators) has space for operators that reverse the derivability relation among formulae. Recall

\[
\Diamond A \rightarrow B \iff A \rightarrow \Box B
\]

Let \( \text{o} \) and \( \text{o} \) be two unary operators. They are said to be Galois-connected if they obey the definition below.

\[
B \rightarrow \text{o} A \iff A \rightarrow B^0
\]

These two operators behave exactly like \( \Diamond \) and \( \Box \), except that they are downward monotonic, cf. the fact that \( B \) occurs on the righthand side of the arrow in \( \Diamond A \rightarrow B \) but on the lefthand side in \( B \rightarrow A \). The algebraic analogy now involves reciprocals: the greater a number, the smaller its reciprocal:\(^{20}\)

\[
x \times y \leq 2 \iff x \leq \frac{2}{y}
\]

As in the case of \( \Diamond \) and \( \Box \), the properties regarding the composition and the monotonicity behavior of the Galois operators follow, namely:

1. \( A \rightarrow \text{o}(A^0) \)
2. \( A \rightarrow (\text{o}A)^0 \)
3. \( \text{o}A \rightarrow \text{o}B, \text{if } B \rightarrow A \) \( \text{[o. downward monotonic]} \)
4. \( A^0 \rightarrow B^0, \text{if } B \rightarrow A \) \( \text{[o. downward monotonic]} \)
5. \( \text{o}(A^0) \rightarrow \text{o}(B^0), \text{if } A \rightarrow B \) \( \text{[o(o)] upward monotonic} \)
6. \( \text{o}(A)^0 \rightarrow \text{o}(B)^0, \text{if } A \rightarrow B \) \( \text{[o(o)] upward monotonic} \)

Notice that since the composition of two downward monotonic operators is upward monotonic, \( \text{o}(A^0) \) and \( (\text{o}A)^0 \) are upward monotonic in \( A \). In what follows we will only use them in pairs, i.e. as (composite) upward monotonic operators.

Double-galois operators can be used to create additional layers of the poset given by \( \Box \) and \( \Diamond \). As (27) indicates, each \( s_n \) derives \( \text{o}(s_n^0) \), and if \( s_n \rightarrow s_k \), \( \text{o}(s_n^0) \rightarrow \text{o}(s_k^0) \). This means that the derivability relations within each double-galois layer are the same as those within the \( \Box \) and \( \Diamond \) layer of the poset. However,\(^{20}\)

\[
x \leq \frac{2}{y} \iff x \leq \frac{2}{y} \text{ let } 0 = \frac{2}{y} \text{ in the place of the unidirectional reciprocal } 1 \text{ we obtain } 0 = 0^0
\]

23

24
the paths are unidirectional: double-galois operators can only be added, not
removed. This means that grammatical categories derive “ungrammatical” ones,
but no “ungrammatical” category derives a grammatical one. This is precisely
what ungrammaticality is.\footnote{The pairs \( \cdot \) and \( \cdot \) are closure operators, therefore the iteration of the same pairs
of Galois produces equalities, viz. \((\cdot \cdot A)^0 \to (\cdot A)^0 \) and similarly for the other pair. On
the other hand, the iteration of different pairs, i.e. \( \cdot \) followed by \( \cdot \) \( \cdot \) and conversely \( \cdot \)
followed by \( \cdot \) produces inequalities, \((\cdot \cdot A)^0 \to (\cdot A)^0 \) but \( \cdot \) \( \cdot \) \( \cdot \)^0 and
similarly for the other combination (see more details in Appendix B). Turning back to our
application, the iterations of different pairs of Galois gives us the possibility to express many
“ungrammatical” sentential layers.}

\[
\begin{array}{c|c|c}
\text{grammatical} & \text{ungrammatical} & \text{ungrammatical} \\
\hline
s_k & \to & 0_1(s_k^0) \\
\uparrow & & \to \\
\hline
s_n & \to & 0_1(s_n^0) \\
\end{array}
\]

An advantage of using the double-galois operators to encode ungrammaticality is that we now have a systematic solution, rather than an ad hoc “blind alley”
for ungrammatical categories.

5.2.2 Additional Layers With Multiple Modes

A similar effect could be achieved with different modes instead of double-galois
operators. Instead of using \( A \to (\cdot A) \), one could create, for every grammatical-
category \( A \), a corresponding NPI-containing category, using the Co-Unit
property with some designated modality:

\[
A \to \Box_n \Diamond_n A
\]

If one needs more ungrammatical layers, then instead of iterating different
galois-connected pairs, iterations of \( \Box \) in different modes can be used.

There are two reasons why using double-galois operators is neater. The weaker
one is that it does not require introducing newer and newer modes. A stronger
reason is that the galois-connected operators only have the (analog of) the
Co-Unit property, whereas the residuated ones have both Co-Unit and Unit.

Therefore they produce many more derivations and the logical space becomes
much richer. In this case, increase in richness may be undesirable. If however
the number of categories in a given linguistic application is relatively small,
sticking with just residuated operators is logically and conceptually simpler.

6 Negative polarity item licensing

6.1 The monotonicity of licensing

This section will examine whether and how Bernardi’s theory of licensing can be
implemented in a realistic setting. But an empirical property of NPI-licensing
has to be introduced first. Different negative polarity items require different
licensors. Zwarts (1983) proposed that the relevant distinctions can be made in
terms of the “negative strength” of the licensors, characterizable with how
many of the de Morgan implications each bears out.

\[
f \text{ is anti-morphic (AM) } \iff f(a \lor b) = fa \land fb \text{ and } f(a \land b) = fa \lor fb
\]

\[
e.g., \text{ not}
\]

\[
f \text{ is anti-additive (AA) } \iff f(a \lor b) = fa \land fb
\]

\[
e.g., \text{ never, nobody}
\]

\[
f \text{ is decreasing (DE) } \iff f(a \lor b) = fa \land fb
\]

\[
e.g., \text{ seldom, at most five men}
\]

Thus we have the following subset relations:

\[
\text{anti-morphic } \subseteq \text{ anti-additive } \subseteq \text{ decreasing}
\]

Van der Wouden (1997) provides a detailed discussion of the Dutch NPI-licensing
data in these terms. To use examples from other languages, Num (1994) argues
that the Korean exceptive \textit{pakkey} ‘only’ is an NPI that requires an antimorphic
licensor. \textit{English in weeks} requires an antiadditive one, and \textit{ever} is satisfied
with one that is (roughly) decreasing:

a. We haven’t been there in weeks.

b. Nobody has been there in weeks.

c. *At most five men have ever been there.

These properties play a role in other licensing relations as well. Roughly de-
creasing adjuncts undergo negative inversion in English (Büring 2004):

\[
\text{Under no / few / *some circumstances would I do this.}
\]

These data sets exhibit what we may call “the monotonicity of licensing”:

Monotonicity of Licensing:

1. A weak NPI is licensed by an operator that is decreasing
   or stronger.

2. A medium NPI is licensed by an operator that is anti-additive
   or stronger.

3. Negative inversion involves adjuncts that are decreasing
   or stronger.

We expect the syntax of licensing to conform to this generalization (where it
indeed holds). How could this be done? One possibility is for nobody, for
instance, to be tagged separately as decreasing and as anti-additive. But one
hopes that it is not necessary to resort to such brute force methods, and the
monotonicity of licensing can be captured in the form of derivability (inclusion)
relations.
At first blush one might think that this requires incorporating the inclusion relations in (28) into the syntax, but that is not the case. It suffices if the following derivability relations hold between the categories:

- arg. of strong, antimorphic licensor \(\rightarrow\) val. of strong licensee
- arg. of medium, antiadditive licensor \(\rightarrow\) val. of medium licensee
- arg. of weak, decreasing licensor \(\rightarrow\) val. of weak licensee

It is easy to see that if these relations hold, a weak licensee like ever for example can be licensed by any of the three kinds of licensors — without the syntax incorporating any derivability relations between the categories of the licensors.

6.2 Is it logically viable?

Does our calculus in general and the logical space explored in Figure 2 in particular make it possible to pick categories in the desired way? The following assignment of value categories to licensees and argument categories to licensors will do. No arrow between two categories means no derivability.

(30) arg. of strong lic-or \(\rightarrow\) val. of strong lic-ee
arg. of medium lic-or \(\rightarrow\) val. of medium lic-ee
arg. of weak lic-or \(\rightarrow\) val. of weak lic-ee

Figure 6 below shows that this is indeed viable. Figure 6 exhibits double-galois categories and their derivability relations. Recall from Section 5 that \(\delta(s^4_0) \rightarrow \delta(s^4_0)\) iff \(s_a \rightarrow s_b\). Therefore the patterns of the derivability in Figure 6 are familiar; they are exactly the same as the ones in galois-free Figure 2. Because no double-galois category derives a galois-free category, relations between the galois-free (“grammatical”) categories that are not part of the diagram cannot cause trouble. The derivability relations relevant in (30) are highlighted with double lines in Figure 6. It is easy to see that all the relations required in (30) hold. On the other hand, \(\delta(s^4_0), \delta(s^2_1),\) and \(\delta(s^3_0)\) are independent.

The diagram also contains curved arrows corresponding to the categories of linguistic expressions. The argument categories of licensees (with dotted lines) are replaced by bullets, since they are irrelevant from the present perspective and will vary with the licensees under consideration. The strong, medium and weak licensors are supposed to be antimorphic, antiadditive, and decreasing.

22In subsection 6.4 we come back to the question whether incorporating (28) into the syntax would be possible at all.

23The downward monotonic nature of both the licensors of NPI and the galois operators is a pure coincidence. Notice that we use galois operators always in a pair, i.e. as upward monotonic operators, and moreover, the same application of galois operators could be used to model other sorts of licensing relations that do not involve downward monotonicity of the licensors.

Figure 6: Licensors and licensees with derivability relations among “ungrammatical” categories

functors, respectively, which, following Section 5, point from double-galois (ungrammatical) to galois-free (grammatical) categories. What their concrete value categories are is irrelevant from the general logical perspective, and so Figure 6 indicates them with empty circles. They are however absolutely relevant from an empirical perspective, to which we now turn; the reader is invited to fill in the circles in due course.

6.3 Is it empirically viable?

What we have seen demonstrates that it is possible in our calculus to assign categories to licensees and licensors in the manner envisaged in Section 5. The empirical question is whether natural language expressions can be matched up with these possibilities. In this paper we only discuss the empirical properties of licensors in detail. We simply assume that the licensees can be assigned to categories in accordance with Figure 6.

We take the Hungarian operator poset in Figure 3 as a point of departure. A quick glance at Figure 3 reveals that Hungarian has suitable decreasing oper-
ators. Almost all the merely decreasing quantifiers in Hungarian are counters, assigned to s7/s2 in Figure 3; this is now revised to s7/0(s22). The revision does not affect the word order behavior of counters, since all and only those galois-free categories that derive s2 derive 0(s22).²⁴

Nem mindenki ‘not everyone’ is decreasing but its word order behavior slightly differs from that of counters; its category in Figure 3 is s4*/s3. If nem mindenki is a good NPI-licensor, then its category should be revised to s4*/0(s30). It turns out that negated universals are cross-linguistically poor licensors of even weak NPIs, compare:

(31) *Not everyone saw anything / has ever been there.

Why this is so is something of a mystery. One possibility is to attribute the unacceptable ness of (31) to the intervention of everyone between not and anything/every, cf. ‘I don’t think that everyone saw anything’ (Linebarger 1987). If however there is reason to analyze not everyone as one complex quantifier, then the intervention account becomes less obvious. Indeed, the complex quantifier analysis was motivated for Hungarian in Section 3.2. Since the judgment in (31) is replicated in Hungarian, there is no reason to assign the licensor category s4*/0(s3) to nem mindenki.

Hungarian is a strict negative concord language and as such it has no antiadditive quantifier phrases comparable to English no one. For the purposes of the present investigation we may however contemplate a closely related imaginary language Hungarian’ that has no strict negative concord but has an antiadditive. This imaginary item is comparable to no one in its scope behavior: it does not scope immediately above negation but can be immediately outscoped by a decreasing counter, cf.

*No one didn’t laugh.

Few men saw no one.

These properties are guaranteed by assigning it to the category s2/s5, to be revised as s2/0(s50) because it is an NPI-licensor.

Hungarian has even two anti-morphic operators: lo-neg and hi-neg. Are they both strong licensors? If not, which of the two is? It turns out that the choice of lo-neg is simply incompatible with our most basic assumptions. If it were a strong licensor, its category would be s1*/0(s10). But 0(s10) is the bottom element of our small set of categories, and indeed the “center” of the whole (infinite) set of categories defined by our calculus. If the value category of strong licensors (or medium licensees, for that matter) derived 0(s10), then it would derive the argument categories of all licensors. That move would wipe out all the strong/weak distinctions we are trying to accommodate. Therefore lo-neg is not in the game. Fortunately, we can resort to the hi-neg version of nem, previously assigned to category s4*/s4. This is now revised as s4*/0(s40). ⁰(s40)

²⁴Although both ‘more than six men’ and ‘few men’ are counting quantifiers, their categories are now distinguished: ‘more than six men’ is s7/s2 but ‘few men’ is s7/0(s22). Thus while their word order behavior is otherwise the same, only the latter is a licensor.

is the right value category for strong licensees, because it does not derive the argument categories of either weak or medium licensors, and of course it derives itself in the capacity of being the argument category of the strong licensors.

The assumption that hi-neg is a licensor but lo-neg is not is empirically less strange than it may initially sound. In (32), where the finite verb is preceded by just one negation and by no focus or counter, this negation could be an instance of either lo-neg or hi-neg. The category of the inflected verb, s1 derives the argument categories of both s1 and s4.

(32) Nem hiszenn, hogy valaki is hallotta volna a hirt, not think.1sg that someone even heard aux the news ‘I don’t think that anyone heard the news’

Since (32) contains the NPI valaki is ‘someone even’, we will simply take its nem to be hi-neg. The one case where hi-neg and lo-neg are distinguishable is where a focus or counter precedes the negation. Our analysis makes the prediction that (33), which cannot but involve lo-neg, is unacceptable. As linguists often say, the judgment is subtle, but the example is certainly less natural than (32):

(33) *Én nem hiszenn, hogy valaki is hallotta volna a hirt. I not think.1sg that someone even heard aux the news ‘It is me who doesn’t think that anyone heard the news’

Hi-neg licenses an NPI only in the absence of an intervening focus, cf. (34).

Since Linebarger (1987) NPI-licensing has been known to be sensitive to operator intervention. Whatever technique is employed to capture this, it will rule out (34):

(34) *Nem én hiszenn, hogy valaki is hallotta volna a hirt.
not I think.1sg that someone even heard aux the news ‘It is not me who thinks that anyone heard the news’

All in all, it is not unreasonable to assume that in this licensing domain hi-neg, but not lo-neg, represents the strong, antimorphic licensor and, given the fact that lo-neg is at the bottom of our category set, this is indeed the only option.

To summarize, the following licensors fit the recipe and work for the reincarnation of Hungarian dubbed Hungarian’s:

Strong NPI-licensor: s4*/0(s40) – example: hi-neg nem

Medium NPI-licensor: s2/0(s50) – example: imaginary ‘no one’

Weak NPI-licensor: s7/0(s22) – example: any decreasing counter

Notice that in this theory not only NPI-licensing is a licensing relation. Any structure that must be immediately outscoped by a particular kind of operator is a “licensor” – one whose value category is an “ungrammatical category”, wherefore its superstructure can only derive s9, the category of grammatical sentences if it is brought back to the “grammatical plane”. Does our theory predict that no licensee might call for, or allow for, lo-neg as a licensor in
Hungarian (or in Hungarian′)? It does not. Recall from Section 5.2.1 that infinitely many distinct “ungrammatical planes” can be formed by adding new pairs of galois-operators: the iteration of different pairs, i.e. \((\mathcal{Q})\) followed by \((\mathcal{Q})\) and conversely \((\mathcal{Q})\) followed by \((\mathcal{Q})\) produces inequalities, \((\mathcal{Q}) \rightarrow \mathcal{Q}((\mathcal{Q})((\mathcal{Q}((\mathcal{Q})))))\) and similarly for the other combination.

The conclusion is that semantic inclusion does not amount to syntactic inclusion, which Serbo-Croatian and Hungarian show is not the case.

The natural explanation of the mismatch between semantic properties and word order behavior is that each expression has many semantic properties, whereas our syntax builds all word order properties into the syntactic category of the expression. (This is indeed the basic idea of categorial grammar. If I know your category, I know how you behave.) But then we cannot expect one particular semantic property to correspond to a syntactic category. Our syntax differs from the Minimalist syntax employed in Stabler (1997), for example, where each lexical item is a bundle of syntactic features, including [determiner], [decreasing], [singular], etc. Stabler’s idea is to couple that syntax with a Natural Logic, whose inference schemata are anchored to some of the features, e.g. [decreasing]. Stabler’s framework would probably lend itself more easily to studying whether a thorough-going match between genuine semantic properties and syntactic behavior can be found.

### 6.4 Semantics in the syntax? Is licensing truly monotonic?

In (28) it was observed that licensors exhibit a semantic inclusion relation: anti-morphic \(\subseteq\) anti-additive \(\subseteq\) decreasing. Perhaps the most appealing way to accommodate the monotonicity of licensing would be to import the corresponding derivability relations between argument and value categories into the syntax.

Would that be possible? We have already seen some empirical reasons why it would not be. First and foremost, the semantic approach would force us to treat hi-neg and lo-neg alike. Or, if they can be semantically distinguished, lo-neg might end up as “the” anti-morphic operator. But lo-neg is at the bottom of both the semantic inclusion hierarchy and the syntactic category hierarchy. Therefore, as was observed above, if the value category of strong licensors derives the argument category of lo-neg, it would inescapably derive the argument categories of medium and weak licensors as well, and all the licensing distinctions would be lost.

Secondly, notice that for the sake of the argument we considered a Hungarian′ which is not a strict negative concord language (and thus has anti-additive generalized quantifiers) and whose NPIs are exactly like NPIs in English or Dutch. These two related properties of Hungarian′ do not hold of plain Hungarian. Progovac (1994) observed that the distribution of English anything is covered by two complementary items in Serbo-Croatian: ništa in the context of clause-mate negation and ista elsewhere. Ništa is a strict negative concord item in our sense, and ista a NPI. But the licensing of ista is non-monotonic: while it is licensed by clause-mate ‘few men’, it is not licensed by clause-mate ‘not’. The same holds for Hungarian: senki is the equivalent of ništa and valoški is of ista. (The latter would have the value category \((\mathcal{Q})((\mathcal{Q}))\), not \((\mathcal{Q})((\mathcal{Q}))\) when its licensor is clause-mate.) If derivability relations corresponding to semantic inclusion were part of the syntax, licensing should always be monotonic, which Serbo-Croatian and Hungarian show is not the case.\(^{25}\)

The conclusion is that semantic inclusion does not amount to syntactic inclusion. It is not true that a semantically stronger operator can do everything in syntax that a semantically weaker one can. It may have restrictions of its own that the weaker one lacks. Therefore, simply importing semantic inclusion relations into the syntax is not possible.

The natural explanation of the mismatch between semantic properties and word order behavior is that each expression has many semantic properties, whereas our syntax builds all word order properties into the syntactic category of the expression. (This is indeed the basic idea of categorial grammar. If I know your category, I know how you behave.) But then we cannot expect one particular semantic property to correspond to a syntactic category. Our syntax differs from the Minimalist syntax employed in Stabler (1997), for example, where each lexical item is a bundle of syntactic features, including [determiner], [decreasing], [singular], etc. Stabler’s idea is to couple that syntax with a Natural Logic, whose inference schemata are anchored to some of the features, e.g. [decreasing]. Stabler’s framework would probably lend itself more easily to studying whether a thorough-going match between genuine semantic properties and syntactic behavior can be found.

### 7 Summary and Comparisons

This paper has argued that using a partially ordered set of categories offers a unified theory for solving the problem of complement selection in the presence of optional categories and accommodating licensing relations. As was mentioned at the outset, the partial ordering on the set of categories could always be stipulated. If instead the category labels are logical formulae then the ordering is given by the derivability relation of the logic. This fact has at least two important advantages. One is that the logic will predict what categories or feature structures can combine; this is how Johnson (1991), Johnson and Bayer (1995), and Blackburn and Spaan (1993) use their logics. Another is that the logic will allow one to create systematic relationships between certain, smaller or larger, sets of categories. This use of the logic is more novel, to our knowledge.

(i) The categories s1, s2, and s3 are members of the same basic layer and are ordered as s1 \(\rightarrow\) s2 \(\rightarrow\) s3. This models the situation where expressions of category s1 or s2 can satisfy a higher head that selects for a complement of category s3, i.e. where s2 and s3 are optional.

\(^{25}\)Empirically even the English data are more complicated, see De Decker et al. (2005).
Such a layer, or parts of such a layer, can be multiplied by the use of different modes. The categories $s_1$ and $s_1^*$ belong to two distinct modes and are not ordered with respect to each other. However, just as $s_1 \rightarrow s_4 \rightarrow s_3$, $s_1^* \rightarrow s_4^* \rightarrow s_3$. Therefore there are two minimally distinct ways to derive $s_3$. This models the situation where $s_1$ and $s_1^*$ differ from each other in one respect and some other categories are sensitive to the distinction; in all other respects however $s_1$ and $s_1^*$ as well as those other categories behave identically. We used two modes to capture some quantifiers’ constraints with respect to negation in their immediate scope.

The basic layer (together with its distinct modes) is fully replicated by arbitrarily many other layers unidirectionally derived from it: $s_1' \rightarrow s_2' \rightarrow s_3', s_1'' \rightarrow s_2'' \rightarrow s_3''$, etc. Fully replicated means that the exact same derivability relations obtain in each layer: $s_1 \rightarrow s_2$ iff $s_1' \rightarrow s_2'$, and unidirectionality derived means that $s_1 \rightarrow s_1''$ but never the other way around. Given this unidirectionality, an expression whose category belongs to one of the “primed” layers can only be part of a grammatical sentence (whose category is on the basic layer) if a wider scoping operator maps it back to the basic layer. This models the situation where an otherwise well-formed expression is “ungrammatical” in that it requires licensing by a particular wider scoping operator; each “primed” layer corresponds to one kind of licensing need. We used such an ungrammatical layer to assign categories to expressions containing an unlicensed NPI.

The fact that the different modes and layers have identical internal derivability relations ensures that “other things” are always kept equal.

The grammar outlined this paper was formulated using a version of the Lambek calculus. However, the ideas are independent both of the Lambek calculus and of our particular additions. The same ideas pertaining to the role of partial ordering can be formulated in theories that do not use these particular techniques. In the subsections below we sketch how the same data involving Hungarian quantifiers could be described in two alternative ways.

### 7.1 Partial Orders With Conjunctive Formulae

A distinctive property of the proposal explored in this paper is that the ordering relation on the basic layer was given by the derivability relation between propositions decorated with the residuated $\square$ and $\lozenge$, and the ordering relation between this layer and the “ungrammatical” layers was given by the derivability relation between the former propositions and ones additionally decorated with the galois operators $\circ$ and $\circ'$. Naturally, our actual grammar of Hungarian uses only a finite and small subset of the formulae in this calculus. The same small poset could be defined in alternative ways, for example, using formulae with conjunction. The poset in Figure 7 is isomorphic to the one exhibited in Figure 3. The partial ordering is given by $p \land q \rightarrow p$ (in the notation of this diagram, $p.q \rightarrow p$).
The question is whether a reasonable linguistic interpretation can be given for the atomic propositions in Figure 7. The answer is Yes. We continue to assume that the categories of Hungarian operators are as the curved arrows in Figures 5 and 6 indicate. In the following interpretation, \( A \) = “A topic can immediately precede me” means that a category with label \( A \) derives the argument category of those Hungarian expressions that we called topics. \(^{26}\)

\[
\begin{align*}
A &= \text{A topic can immediately precede me} \\
B &= \text{A distributive existential can immediately precede me} \\
C &= \text{A negated universal can immediately precede me} \\
D &= \text{A negative concord item can immediately precede me} \\
E &= \text{A distributive universal can immediately precede me} \\
F &= \text{A counter or a focus can immediately precede me} \\
I &= \text{Lo-negation can immediately precede me} \\
J &= \text{I do not need a NPI licensor}
\end{align*}
\]

For example,

\( s_4 = A.B.C.E.J \) = A topic, a distributive existential, a negated universal, and a distributive universal each can immediately precede me, and I do not need a NPI licensor.

\( s_4^* = A.B.C.D.J \) = A topic, a distributive existential, a negated universal, and a negative concord item each can immediately precede me, and I do not need a NPI licensor.

\( s_5^* \) = A.B.C.E = A topic, a distributive existential, a negated universal, and a distributive universal each can immediately precede me.

To wit, \( s_4 \) derives the argument categories of topics (\( s_9 \)), distributive existentials (\( s_8 \)), negated universals (\( s_3 \)), and distributive universals (\( s_7 \)). So the propositions \( A \) through \( I \) express exactly the kind of information that we used to model the behavior of Hungarian operators earlier in this paper. Even the generalizations that emerge are the same. Members of the \( s_1 - s_4 - s_7 \) track share \( E = “A distributive universal can immediately precede me”; members of the \( s_1^* - s_4^* - s_7^* \) track share \( D = “A negative concord item can immediately precede me”, and the conjunct that only \( s_1 \) has is \( I = “Lo-negation can immediately precede me”\).

A novelty is the fact that each element of the poset has a \( J \) conjunct. This serves to replicate the effect of the double-galois operators. In addition to whatever categories are ordered above it in this set, each grammatical category derives an ungrammatical one with the same label minus the \( J \) conjunct. Compare \( s_4 \) and \( s_5^* \) above. \( p \land r \rightarrow q \) iff \( p \land r \rightarrow q \land r \), and grammatical categories unidirectionally derive ungrammatical ones: \( p \land r \rightarrow p \), but not the other way around.

The linguistic difference between implementing Bernardi’s theory of licensing with double-galois operators and with conjunction is that in the double-galois case the grammatical categories are “unmarked” and the ungrammatical ones are “marked”, whereas in the conjunction case each grammatical category is marked for what licensing needs it does not have. Ungrammatical categories have those needs in view of the fact that they are not specified as not having them. This awkward or even undesirable feature of the present, simple implementation may well be avoidable using a different Boolean construct. In any case, any number of independent ungrammatical layers can be defined by using logically independent propositions of the "I do not need ... " kind. Or, if there is a derivability relation between the ungrammatical layers, conjunctions of such propositions could be used.

Apart from the “markedness” point, there is little empirical difference between the calculi using unary operators and conjunctions. In both cases the empirical claims are embodied in the categories assigned to the operators, not in the labeling of the argument and the value categories. One might say that the conjunctive formulae are more explicit than the ones with unary operators, but their labels do not in any way explain why exactly those expressions can immediately precede something of the given category.

Logically speaking the conjunctive formulae are simpler. On the other hand, as Kurtonina and Moortgat (1995) point out, \( \Box \) and \( \Diamond \) are natural extensions of the Lambek calculus with residuated binary connectives. The significance of this observation may be dependent on how insightfully the Curry-Howard correspondence extends to the unary operators. This is a subject of ongoing research, some of which is discussed in Bernardi and Moortgat’s ESSLLI 2007 course.

### 7.2 An Attribute Value Matrix Approach

Yet another, quite different approach might be to revisit the characterization of Hungarian operators in Section 3.2. At that point we argued that introducing features like \([\text{pred}]\) and \([\text{contrast}]\) preserves only the illusion of having totally ordered functional categories in Minimalism. But such features might be put to use in a different framework. The following list that merely recaps the characterization of some of the operators discussed there could be recast in an AVM format.

---

\(^{26}\)s0, s10*, s5, s5*, and s6 were not part of the actual Hungarian category inventory, so we do not attempt to give \( G \) and \( H \) a realistic interpretation.
This approach might eventually build a bridge to the compositional semantics of the operators.

Acknowledgments. We are grateful to Lucas Champollion and Chris Barker for comments on an earlier version of this manuscript, and to Øystein Nilsen, Eytan Zweig, Michael Moortgat, Kit Fine, Julia Horvath, Mark Baltin, and Ed Stabler for extensive discussions about various parts of the project, and to three reviewers for very helpful comments pointing to related work. Part of this research was presented at the workshop on “Proof Theory at the Syntax/Semantics Interface” (LSA Linguistic Institute, MIT/Harvard, July 8-11, 2005) and at a seminar team-taught by Bernardi, Nilsen, and Szabolcsi at New York University in Spring 2005; we thank the participants of both for their input. This project was partially sponsored by New York University’s International Visitors Program in 2005.

References


shop on "Polarity from Different Perspectives" NYU, http://www.nyu.edu/gsas/dept/lingu/events/polarity/posters.htm.


Appendix A: Properties of Unary Residuated Operators

The proofs of the properties of residuated unary operators are given below. All the derivability relations among two formulae decorated with ♦ and □ are due to these properties. The arrows in the Figure ?? are the results of different orders of application of these properties.

(a) Unit: ♦□A → A
   i. □A → □A [Axiom]
   ii. □□A → A [Residuation]

(b) Co-unit: A → □♦A
   i. ◊A → ◊A [Axiom]
   ii. A → □♦A [Residuation]

(c) Monotonicity of □
   i. ◊A → A [by Unit]
   ii. A → B [Hypothesis]
   iii. ◊A → B [from i. and ii. by trans]
   iv. □A → □B [from iii. by Residuation]

(d) similarly for the Monotonicity of ◊

Not all iterations of unary operators patterns produce formulae that are not equivalent to simpler ones. In particular, the iteration of □◊ is unproductive, i.e. we obtain equalities of formulae: □◊□◊A → □◊A; similarly for ◊□, ◊□A → ◊□◊A.

On the other hand, if we compose □◊ with the other pair ◊□ we obtain inequalities, viz. ◊□A → ◊□◊□A and similarly for ◊□A → ◊□□◊A. But neither □◊□□A nor ◊□□◊A derive ◊□A, though they derive ◊□A. Other productive patterns of unary operators are what we call center embeddings. In particular, if we plug the pair ◊□ into the middle of □◊ we obtain an inequality among the formulae, namely ◊□□◊A → ◊□A but not the other way around.

Similarly, if we plug □◊ into the middle of □◊ we obtain a new formula, viz. □◊A → □◊◊◊A. In the following, we highlight the embedded pairs by underlining them.

Unproductive Iteration Unproductive iterations are due to the fact that both ◊□- and □◊- are closure operators.

(a) □◊A → □◊◊◊A
   i. ◊A → ◊A [Axiom]
Productive Iterations

Productive iterations are obtained in two ways,
(I) by combining different pairs of unary operators, namely ♦□ with □♦ and conversely, and (II) by center embeddings.²⁷ The derivations are spelled out below.

(I)

i. A → A
   ♦□A → ♦□A [by Mon. of □ and Mon. of ♦]
   □♦A → □♦□♦A [by Co-Unit]
ii. A → A
   □A → □◊◻A [Co-Unit]
   ♦◊◻A → ♦□□♦A [Mon. of ♦ and Mon. of □]

(II) (a) ♦◊◻A → ◊◻A

   i. ◊◻A → ◊◻A [Axiom]
   ♦◊◻A → ◊◻A [from i. by Unit]
   ♦◊◻A → ♦◊◻A [from ii. by Mon. of ♦]
   whereas, ♦◊◻A → ♦◊◻A

   (b) □♦A → □◊◻A

   i. ♦A → ◊A [Axiom]
   ♦A → □◊◻A [from i. by Co-unit]
   □◊◻A → □◊◻A [from ii. by Mon. of □]
   whereas, □◊◻A → □◊◻A

The following diagram illustrates the derivability relations within one small set of decorated categories. It exhibits all the derivability relations that exist within the given set of categories (although of course these categories are derivable from infinitely many others and derive infinitely many others). The

²⁷These patterns have been pointed out to us by Eytan Zweig during the visit of the first author to the NYU Linguistics Department.
The second figure is minimally different. It illustrates the effect of using slightly different steps, resulting in the replacement of $s10$ with $s11$. The little circles indicate that derivability arrows are crossing and there is no category label omitted there.
Appendix B: Properties of the Galois Operators

Similar observations hold for the Galois operators. In this case as well we have to consider the two different pairs, namely, $\langle \cdot, 0 \rangle$ and $\langle 0, \cdot \rangle$. For the sake of transparency we notate the second pair as $\langle 0, \cdot \rangle$. See Goré (1998) for the modal theoretical interpretation of these operators, and Areces et al. (2003) for the proof of soundness and completeness of the full Lambek calculus with Galois and Residuated unary operators.

As in the case of the residuated operators, iteration yields an equality, $(\langle 0, \cdot \rangle) 0 \longrightarrow (\langle 0, \cdot \rangle)$ and $\langle 0, \cdot \rangle$. On the other hand, the composition of different pairs produces inequalities, namely $(\langle 0, \cdot \rangle) 0 \longrightarrow (\langle 0, \cdot \rangle)$, $(\langle 0, \cdot \rangle) \longrightarrow (\langle 0, \cdot \rangle)$, and the same holds for the formula $\langle 0, \cdot \rangle$.

Furthermore, in this case as well productive patterns are obtained by means of center embeddings. We can embed $\langle 0, \cdot \rangle$ within another pair of the same sort $\langle 0, \cdot \rangle$ obtaining $\langle 0, \cdot \rangle(\langle 0, \cdot \rangle)$ and similarly for the other pair.

(a) Co-unit': $A \longrightarrow (\langle 0, \cdot \rangle)$
   i. $0_A \longrightarrow 0_A$ [Axiom]
   ii. $A \longrightarrow (\langle 0, \cdot \rangle)$ [Galois]

(b) similarly for the other pair

(c) Monotonicity of $\langle 0, \cdot \rangle$
   i. $A \longrightarrow (\langle 0, \cdot \rangle)$ [Co-unit']
   ii. $B \longrightarrow A$ [Axiom]
   iii. $B \longrightarrow (\langle 0, \cdot \rangle)$ [from i. and ii. by trans.]
   iv. $0_A \longrightarrow B$ [from iii. by Galois def.]

(d) similarly for $\langle 0, \cdot \rangle$.

Unproductive Iterations

(a) $(\langle 0, \cdot \rangle) 0 \longrightarrow (\langle 0, \cdot \rangle) 0$
   i. $0_A \longrightarrow 0_A$ [Axiom]
   ii. $0_A \longrightarrow 0 (\langle 0, \cdot \rangle)$ [from i. by Co-Unit']
   iii. $(\langle 0, \cdot \rangle) 0 \longrightarrow (\langle 0, \cdot \rangle) 0$ [from ii. by Mon. of $\langle 0, \cdot \rangle$
   i'. $(\langle 0, \cdot \rangle) 0 \longrightarrow (\langle 0, \cdot \rangle) 0$ [by Co-unit']

(b) similarly for the other pair.

Productive Iterations

(I) $(\langle 0, \cdot \rangle) 0 \longrightarrow \langle \langle 0, \cdot \rangle \rangle$ simply by Co-unit. Whereas, $\langle \langle 0, \cdot \rangle \rangle \not\longrightarrow (\langle 0, \cdot \rangle)$

(II) (a) i. $A 0 \longrightarrow A^0$ [Axiom]
   ii. $A^0 \longrightarrow (\langle 0, \cdot \rangle)$ [from i. by Co-Unit']
   iii. $\langle 0, \cdot \rangle(\langle 0, \cdot \rangle) \longrightarrow A^0$ [from ii. by Mon. of $\langle 0, \cdot \rangle$]

(b) similarly with the other pair.

For the application in this paper it is important to pay particular attention to the following difference between Galois-connected and residuated operators: while the pair of residuated operators $\langle 0, \cdot \rangle$ can disappear from a formula by means of the Unit $\not\longrightarrow A$, there is not such possibility for the pairs of Galois, there is neither $(\langle 0, \cdot \rangle)$ nor $\langle 0, \cdot \rangle$ (see the derivation of the Unit above 7.2.)

This fact is relevant for us, since we use pairs of Galois operators to mark “ungrammatical” expressions, and of course, they should not have the power of becoming grammatical by themselves, but rather only when a proper operators (a licensor) take scope over them.
Appendix C: A sample Natural Deduction derivation

(36) Kati nem látott mindenkit.
Kate not saw 3sg everyone-acc
‘Kate did not see everyone’

The analytical assumptions in Figure 10 follow strictly what is argued for in Brody and Szabolcsi (2003) and merely recapture it in a different framework. These assumptions are as follows. Inflectional heads are obligatory; operator expressions are optional. Figure 10 contains three inflectional heads, C(complementizer), Agr(ement), and T(ense). Morphology spells out the finite verb in Agr but the verb does not move there in syntax. Negation is specified to occur only in the operator sequence of Agr; a name and a distributive universal may occur in any of the operator sequences. Although within a single sequence the universal is ordered before negation, negation is capable of scoping over it in (36) because the universal occurs in the operator sequence of T. The intervening Agr head mediates between the two sequences. The topic Kati occurs in the Agr-sequence. The topic and the universal bind traces of category dp.

Undecorated s serves as the category of uninflected sentences. The obligatory-ness of inflectional heads is captured by assigning them categories decorated with an indexed box (see Moortgat (1999) for a detailed description of this use of unary operators). T(ense) for example has the category □s(Agr(s1/s)), □E(s) moves the decoration over to T in the form of ( . . . )T, and the structural rules abbreviated as [Pxx] put it back to the whole chunk containing T, right before Agr should enter the picture. Agr now has the category □T/A(s1/□T/s9), which crucially differs from that of T in that the argument it seeks is not uninflected S but a sentence already containing T. [□I] allows (everyone o (T o (see o (□dp)))) to be recognized as such. The same holds for C requiring an argument that contains Agr. C closes off the clause.

The value categories of all operators derive s9. Both Agr and C have s9 for their argument categories. This allows a full set of operator expressions – or no operator expression at all – to occur right below C and Agr. The categories of operators that freely occur in any sequence (i.e. either preverbally or postverbally) are not tagged for inflectional heads. [□I] is interpreted as λ-abstraction and thus allows the operators to bind their dp traces. Negation however occurs only in the preverbal field. To ensure this its argument category is decorated with □A. The □A decoration on its whole functor category plays the same role as it does with Agr.
Type raising, continuations, and classical logic

Philippe de Groote
Inria-Lorraine

Abstract. There is a striking analogy between type raising, as introduced by Montague (1973), and the notion of continuation that has been developed in programming language theory in order to give compositional semantics to control operators (Stratchey and Wadsworth, 1974). In fact, this analogy is such that it is possible to see Montague’s semantics as a continuation based semantics.

On the other hand, the notion of continuation allows classical logic to be given a Curry-Howard interpretation (Griffin 1990). In particular, the double negation law \((A \rightarrow \perp) \rightarrow \perp\) is provided with a computational content, which may be used to give a type logical interpretation of type lowering.

Putting the pieces of the picture together, it is possible to use “classical extensions” of the \(\lambda\)-calculus in order to express the semantic components of the lexical entries of Morrill’s (1994) type logical grammars. This solution offers the advantage of not burdening the syntax by enforcing type raising to the worst case.

1 Type raising and continuations

Montague (1973) introduced type raising as a way of providing a compositional semantics to constructs that may give rise to scope ambiguities. Such constructs (typically, quantifiers) have semantic scopes that may be wider than their apparent syntactic scopes. Around the same time, computer scientists were trying to provide a compositional semantics to full jumps (i.e., ‘goto’ statements), which led to the discovery of continuations (Stratchey and Wadsworth, 1974).

Both problems are similar, and both solutions present striking similitudes. Montague’s type raising is based on Leibniz’s principle, which consists of identifying an entity with the set of its properties. Consequently, the type of entities \(e\) is replaced by \((e \rightarrow t) \rightarrow t\), where \(t\) is the type of propositions. In programming language theory, a continuation semantics (as opposed to a direct semantics) consists in providing the semantic function with the continuation of the program as an explicit parameter. Let \(P\) be a program, let \(\llbracket - \rrbracket\) be the semantic function, and let \(s\) be some initial state. If we consider programs as state transformers, a direct semantics is such that \(\llbracket P \rrbracket s \in \text{State}\). On the other hand, a continuation semantics gives \(\llbracket P \rrbracket s \in (\text{State} \rightarrow \text{State}) \rightarrow \text{State}\). In fact, in both cases (type raising and continuation semantics), a type \(A\) is replaced by a type \((A \rightarrow O) \rightarrow O\), where \(O\) is the type of observable entities or facts.

2 Negative translations and classical logic

In the realm of the \(\lambda\)-calculus, the notion of continuation gave rise to the so-called CPS-transformations (Plotkin 1975). These are continuation-based syntactic transformations of the \(\lambda\)-terms that allow given evaluation strategies (typically, call-by-name or call-by-value) to be simulated.
For instance, Plotkin’s call-by-value CPS-transformation is as follows:

\[ \sigma = \lambda k. k c; \]
\[ \tau = \lambda k. k x; \]
\[ \lambda x. M = \lambda k. k (\lambda x. \overline{M}); \]
\[ \overline{MN} = \lambda k. \overline{M} (\lambda m. \overline{N} (\lambda n. m n k)) \]

Now, compare the following naive type logical grammar, where the lexical items are assigned a direct interpretation:

<table>
<thead>
<tr>
<th>Lexical Item</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>NP</td>
</tr>
<tr>
<td>Mary</td>
<td>NP</td>
</tr>
<tr>
<td>loves</td>
<td>(NP \ S) / NP</td>
</tr>
</tbody>
</table>

together with the grammar, where the lexical items are assigned a Montague-like interpretation:

<table>
<thead>
<tr>
<th>Lexical Item</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>(\lambda k. k J) : NP</td>
</tr>
<tr>
<td>Mary</td>
<td>(\lambda k. k M) : NP</td>
</tr>
<tr>
<td>loves</td>
<td>(\lambda f. \lambda y. f (\lambda x. g (\lambda y. \text{LOVE} \ y \ x))) : (NP \ S) / NP</td>
</tr>
</tbody>
</table>

Again, the analogy between continuation and type raising is striking. The Montague-like interpretation may almost be seen as the call-by-value CPS-transform of the direct interpretation. This opens a new line of research that has been advocated in Barker’s recent work (2000, 2001).

When applying a CPS-transformation to a typed lambda-term, it induced another transformation at the type level (Meyer and Wand, 1985). For instance, the above CPS-transformation induces the following type transformation:

\[ \overline{\sigma} = (\alpha^* \to \bot) \to \bot, \text{ where:} \]
\[ \alpha^* = a, \text{ for } a \text{ atomic;} \]
\[ (\alpha \to \beta)^* = \alpha^* \to \beta. \]

Griffin (1990) observed that these type transformations amount to double negative translations of classical logic into minimalist logic, and that it allows classical logic to be provided with a formulae-as-type interpretation. In this setting, the double negation law \((A \to \bot) \to \bot \to A\), which corresponds to type lowering, is given a computational content by considering the absurd type \(\bot\) to be the type of observable entities (this is radically different from the usual interpretation of \(\bot\) as the empty type).

3 The \(\lambda\mu\)-calculus

Griffin’s discovery gave rise to several extensions of the \(\lambda\)-calculus, which aim at adapting the Curry-Howard isomorphism to the case of classical logic. The \(\lambda\mu\)-calculus (Parigot 1992) is such a system.

The \(\lambda\mu\)-calculus is a strict extension of the \(\lambda\)-calculus. Its syntax is provided with a second alphabet of variables (\(\alpha, \beta, \gamma, \ldots\) — called the \(\mu\)-variables), and two additional constructs: \(\mu\)-abstraction \((\mu \alpha. \ t)\), and naming \((\alpha t)\).

These constructs obey the following typing rules:

\[
\begin{align*}
  \alpha : \neg A & \quad t : A \\
  \alpha t : \bot
\end{align*}
\]

\[
\begin{align*}
  \alpha : \neg A & \quad t : \bot \\
  \mu \alpha. t : A
\end{align*}
\]

Besides \(\beta\)-reduction:
\((\beta) \ (\lambda \alpha. t) t \rightarrow t[x := u]\)

a notion of \(\mu\)-reduction is defined:

\[(\mu) \ (\mu \alpha. u) v \rightarrow \mu \beta. u[\alpha t_i := \beta (t_i v)]\]

where \(u[\alpha t_i := \beta (t_i v)]\) stands for the term \(u\) where each subterm of the form \(\alpha t_i\) has been replaced by \(\beta (t_i v)\). It corresponds to the following proof-theoretic reduction:

\[
\frac{\alpha : \neg(A \rightarrow B) \quad t_i : A \rightarrow B}{\alpha t_i : \bot} \quad \frac{\beta : \neg B \quad t_i v : B}{\beta (t_i v) : \bot}
\]

\[
\frac{\mu \alpha. u : A \rightarrow B \quad v : A}{(\mu \alpha. u) v : B} \rightarrow \quad \frac{\mu \beta. u[\alpha t_i := \beta (t_i v)] : \bot}{\mu \beta. u[\alpha t_i := \beta (t_i v)] : B}
\]

As well-known, classical logic is not naturally confluent. Consequently, there exist variants of the \(\lambda \mu\)-calculus that do not satisfy the Church-Rosser property (Parigot 2000). This is the case if we also consider the symmetric of the \(\mu\)-reduction rule:

\[(\mu') \ v (\mu \alpha. u) \rightarrow \mu \beta. u[\alpha t_i := \beta (v t_i)]\]

Finally, for the purpose of the example given in the next section, we also add the following simplification rules:

\[(\sigma) \ \mu \alpha. u \rightarrow u[\alpha t_i := t_i]\]

which may be applied only to terms of type \(\bot\).

4 Semantic recipes as \(\lambda \mu\)-terms

Dealing with a calculus that do not satisfy the Church-Rosser property is not a defect in the case of natural language semantics. Indeed, the fact that a same term may have several different normal forms allows one to deal with semantic ambiguities.

If we consider the sentential category \(S\) (or, semantically, Montague’s type \(t\)) to be our domain of observable facts, the following typing judgement is derivable:

\[
\text{PERSON} : e \rightarrow t \quad x : e \quad \alpha : e \rightarrow t \quad x : e
\]

\[
(\text{PERSON} x) : t \quad (\alpha x) : t
\]

\[
\frac{(\text{PERSON} x) \supset (\alpha x) : t}{\forall x. (\text{PERSON} x) \supset (\alpha x) : t}
\]

\[
\mu \alpha. \forall x. (\text{PERSON} x) \supset (\alpha x) : e
\]
This allows the following type logical lexical entries to be defined:

\[
\text{everybody} \quad - \quad \mu \alpha. \forall x. (\text{PERSON} x) \supset (\alpha x) : NP
\]
\[
\text{somebody} \quad - \quad \mu \alpha. \exists x. (\text{PERSON} x) \land (\alpha x) : NP
\]
\[
\text{loves} \quad - \quad \lambda x. \lambda y. \text{LOVE} y x : (NP \setminus S) / NP
\]

Then, the sentence

\[\text{everybody loves somebody}\]

has only one parsing, to which is associated the following semantic reading:

\[
(\lambda x. \lambda y. \text{LOVE} y x) (\mu \alpha. \exists x. (\text{PERSON} x) \land (\alpha x)) (\mu \alpha. \forall x. (\text{PERSON} x) \supset (\alpha x))
\]

This \( \lambda \mu \)-term may be considered as an underspecified representation. Indeed, its possible reductions yield two different normal forms:

\[
(\lambda x. \lambda y. \text{LOVE} y x) (\mu \alpha. \exists x. (\text{PERSON} x) \land (\alpha x)) (\mu \alpha. \forall x. (\text{PERSON} x) \supset (\alpha x))
\]

\[
\rightarrow (\lambda y. \text{LOVE} y (\mu \alpha. \exists x. (\text{PERSON} x) \land (\alpha x))) (\mu \alpha. \forall x. (\text{PERSON} x) \supset (\alpha x)) \quad (\beta)
\]

\[
\rightarrow \text{LOVE} (\mu \alpha. \forall x. (\text{PERSON} x) \supset (\alpha x)) (\mu \alpha. \exists x. (\text{PERSON} x) \land (\alpha x)) \quad (\beta)
\]

\[
\rightarrow (\mu \beta. \forall x. (\text{PERSON} x) \supset (\beta (\text{LOVE} x) (\mu \alpha. \exists y. (\text{PERSON} y) \land (\alpha y)))) \quad (\mu)
\]

\[
\rightarrow \forall x. (\text{PERSON} x) \supset (\beta (\text{LOVE} x (\mu \alpha. \exists y. (\text{PERSON} y) \land (\alpha y)))) \quad (\sigma)
\]

\[
\rightarrow \forall x. (\text{PERSON} x) \supset (\exists y. (\text{PERSON} y) \land (\text{LOVE} y x)) \quad (\sigma)
\]

\[
(\lambda x. \lambda y. \text{LOVE} y x) (\mu \alpha. \exists x. (\text{PERSON} x) \land (\alpha x)) (\mu \alpha. \forall x. (\text{PERSON} x) \supset (\alpha x))
\]

\[
\rightarrow (\lambda y. \text{LOVE} y (\mu \alpha. \exists x. (\text{PERSON} x) \land (\alpha x))) (\mu \alpha. \forall x. (\text{PERSON} x) \supset (\alpha x)) \quad (\beta)
\]

\[
\rightarrow \text{LOVE} (\mu \alpha. \forall x. (\text{PERSON} x) \supset (\alpha x)) (\mu \alpha. \exists x. (\text{PERSON} x) \land (\alpha x)) \quad (\beta)
\]

\[
\rightarrow (\mu \beta. \forall x. (\text{PERSON} x) \supset (\beta (\text{LOVE} x) (\mu \alpha. \exists y. (\text{PERSON} y) \land (\alpha y)))) \quad (\mu)
\]

\[
\rightarrow \forall x. (\text{PERSON} x) \supset (\exists y. (\text{PERSON} y) \land (\text{LOVE} y x)) \quad (\sigma)
\]

\[
\rightarrow \exists y. (\text{PERSON} y) \land (\forall x. (\text{PERSON} x) \supset (\beta (\text{LOVE} y x))) \quad (\sigma)
\]

These correspond to subject and object wide scope readings, respectively.

5 conclusions

We have argued that Montague’s type raising is a particular case of continuation. Consequently, continuation based formalisms, which have been developed in the context of programming language theory, may be used to deal with the sort of semantic ambiguities for which Montague invented type raising. Parigot’s \( \lambda \mu \)-calculus is such a formalism, and we have shown how it may be used to cope with quantifier scope ambiguities. We claim that the \( \lambda \mu \)-calculus is particularly suitable for expressing compositional semantics of natural languages. For instance, it allows Cooper’s (1983) storage to be given a type logical foundation. In fact, it allows a lot of dynamic constructs to be defined, which is of particular interest for discourse representation.
references


Abstract
This paper will appear, in a slightly shortened form, as an in-depth article (# 231) in the Encyclopedia of Cognitive Science, Nature Publishing Group, Macmillan Publishers Ltd. For alerts on the project’s progress, visit www.cognitivescience.net.

Keywords
Categories, types, processing, parsing, deduction.

Article definition
Categorial grammar: a lexicalized grammar formalism based on logical type-theory. A categorial lexicon assigns one or more types to the atomic elements of a language; the assembly of form and meaning is accounted for in terms of the rules of inference for these types seen as formulas of a grammar logic. Cross-linguistic variation results from extending the invariant core of the grammar logic with facilities for structural reasoning.

Contents
1 Introduction
2 Form: grammatical invariants and structural variation
   2.1 The base logic
   2.2 The structural module
   2.3 Generative capacity and computational complexity
   2.4 Language learning
3 Meaning assembly: the Curry-Howard correspondence
   3.1 Model-theoretic semantics, type theory and the lambda calculus
   3.2 Formulas-as-types, proofs as programs
   3.3 The syntax/semantics interface
   3.4 Processing issues
4 Exploration
   4.1 Variants and alternatives
   4.2 Further reading
1 Introduction

Categorial grammar, a linguistic framework with firm roots in type theory and constructive logic, is well represented in the logical and mathematical literature. This article puts the emphasis more on the categorial modelling of the cognitive abilities underlying the acquisition, use and understanding of natural language. The sections below address two central questions. First of all, what are the invariants of grammatical composition, and how do they capture the uniformities of the form/meaning correspondence across languages? Second, how can we reconcile the idea of grammatical invariants with structural variation in the realization of the form/meaning correspondence?

The slogan 'parasing as deduction' concisely expresses the categorial perspective on these questions. A grammar, essentially, is given by an assignment of types to the elementary units in the lexicon. The type-forming operations have the status of logical connectives: determining whether an expression is well-formed amounts to presenting a derivation, or proof, in the logic for these connectives. Natural language expressions are input with a form and a meaning dimension. The categorial type language, consequently, is model-theoretically interpreted with respect to these two dimensions, and a derivation encodes an effective procedure for building up the structural organization of an expression, and for associating this structure with a recipe for meaning assembly.

The article is organized as follows. In §2, we focus on the form dimension of expressions. We identify the logical constants of the computational system, and study how the base logic for these constants can be extended with facilities for structural reasoning. In §3, we see how the logical rules of inference for the type-forming operations can be read as instructions for meaning assembly, and how the structural rules determine which components of an expression can enter into the assembly process. The final section provides some background information and pointers to current areas of research.

2 Form: grammatical invariants and structural variation

2.1 The base logic

Natural language expressions are structured objects that come with a linear order and a hierarchical grouping. In categorial grammar, the traditional parts of speech assume the form of type formulas. The structure of these types mirrors the composition of the expressions they categorize. The set of type formulas Type is obtained as the closure of a small set Atom of basic types under a number of type-forming operations. Individual categorial grammars will differ with respect to the type-forming operations they employ. For the present purposes, the following clauses will be representative.

(1) (ATOMS) Atom is a subset of Type;

(UNARY) if A is a formula in Type, then □A and △A are too;

(BINARY) if A and B are formulas in Type, then A • B, A/B and A\B are too.

Basic types play a role similar to that of major constituents in phrase-structure grammar: they categorize expressions one can think of as ‘complete’. Examples could be the type np for proper names, s for sentences, n for common noun phrases. Languages can differ as to which basic type distinctions they make. The unary and binary operations provide a vocabulary to categorize expressions in terms of their constituent parts. Informally, a formula A • B categorizes an expression that can be decomposed into a constituent of type A followed by a constituent of type B. An expression with a fraction type A/B (or B\A) is incomplete: it combines with an expression of type B on its right (or left, respectively) into an expression of type A. The unary type-forming operations are more recent additions to the categorial vocabulary. They can be thought of as features: an expression of type □A issues a request for a feature to be checked; such an expression can be used as a regular A as soon as the □ feature is eliminated. The operation △ provides the means to perform the required feature-checking.

Frame semantics. To make this informal description precise, Došen (1992) and Kurtonina (1995) make use of frame-based models familiar from possible-world semantics for modal logics. For the categorial type language, a frame is a tuple (W, R\s, R\d). W is a non-empty set, the set of expressions, R\s and R\d are binary and ternary relations over W, interpreting the unary and binary type-forming operations, respectively. One can think of R\s as the ‘Merge’ relation: R\sxy holds in case x is the composition of the parts y and z. Similarly, R\dxy holds if the feature-checking relation connects y to x. One obtains a model by adding a valuation V assigning subsets of W to the atomic formulas. For complex types, the valuation respects the conditions below.

\[ x \in V(\triangle A) \quad \text{iff there exists a } y \text{ such that } R\s xy \text{ and } y \in V(A) \]
\[ x \in V(\square A) \quad \text{iff for all } y, R\d yz \text{ implies } y \in V(A) \]
\[ x \in V(A \bullet B) \quad \text{iff there are } y \text{ and } z \text{ such that } y \in V(A), z \in V(B) \text{ and } R\s yz \]
\[ x \in V(C/B) \quad \text{iff for all } y \text{ and } z, \text{ if } y \in V(B) \text{ and } R\d zy \text{ then } z \in V(C) \]
\[ x \in V(A/C) \quad \text{iff for all } y \text{ and } z, \text{ if } y \in V(A) \text{ and } R\d zy, \text{ then } z \in V(C) \]

2.2 Soundness and completeness

Type computations, soundness and completeness. On the proof-theoretic level, we are interested in a deductive system to perform type computations (type B is derivable from type A). We want this system to be faithful to the interpretation of the type-forming operations, in the following sense:

\[ \text{SOUNDERNESS AND COMPLETENESS} \]
\[ A \rightarrow B \text{ is provable iff } V(A) \subseteq V(B), \text{ for every frame } V \text{ and valuation } V. \]

An axiomatization satisfying the soundness and completeness requirements starts with an identity axiom \( A \rightarrow A \), and an inference rule allowing one to conclude \( A \rightarrow C \) from premises \( A \rightarrow B \) and \( B \rightarrow C \). Semantically, these express the reflexivity and transitivity of the derivability relation. In addition, one has the inference rules in (4) establishing the relationship between the interpretation of \( \triangle \) and \( \square \), and between \( \bullet \) and left and right division \( \div \) and \( \div \). The patterns in (4) turn \( (\triangle, \square, (\bullet, /) \) and \( (\bullet, \div) \) into what are known as residuated pairs in algebra, or adjoint functors in category theory.

\[ \text{(4)axioms)} \]
\[ (R0) \quad A \rightarrow B \quad \text{if and only if } A \rightarrow \square B \]
\[ (R1) \quad A \rightarrow B \rightarrow C \quad \text{if and only if } A \rightarrow C/B \]
\[ (R2) \quad A \rightarrow B \rightarrow C \quad \text{if and only if } B \rightarrow A/C \]

Sample theorems. Let us look at some elementary theorems of the grammatical base logic. From the identity axiom, one obtains the Application schema of (5b) in one step, using the \( \triangle \) and \( \square \) to express the combinatory role.

\[ \text{(5)theorems)} \]
\[ a. \quad A/B \rightarrow A \rightarrow B \] (Ax)
\[ b. \quad (A/B) \rightarrow A \rightarrow B \rightarrow (R2 \rightarrow) \]
\[ c. \quad A \rightarrow B \rightarrow (A/B) \rightarrow (R1 \rightarrow) \]
\[ A \rightarrow (B/A) \rightarrow (R2 \rightarrow) \]

The Application schemata are no doubt the most familiar laws of categorial combinatory. The original categorial grammars of Ajdukiewicz and Bar-Hillel in fact were restricted to Application. Using the Application schemata, one can ‘lexicize’ the rules of a context-free phrase structure grammar. Take the productions \( S \rightarrow NP \rightarrow VP \) and \( VP \rightarrow TV \rightarrow NP \) for the derivation of a Subject-Transitive Verb-Object (SVO) pattern. In categorial terms, one types the Transitive Verb as \( (np)(x) / np \), thus projecting the SVO pattern in two Application steps: rightward application consumes the Object, leftward application the Subject. The auxiliary label VP disappears; the complex type np\( x \) expresses its combinatory role.

Instances of Lifting would be type transitions from np (the type assigned to simple proper names) to \( s(np)(x) \) or \( (np)(x) / np \)\( (np)(x) \). These lifted types are appropriate for noun phrases.
with a distribution restricted to the subject position, in the case of \(s/(np)s\), or the direct object position, in the case of \((np)s/(np)s\). What the derivability arrow says here is that any expression that is assigned the type \(np\) will be able to occur in subject or object position, but that there can be expressions with a restricted subject or object distribution, expressed through the higher order types. One can think of case-marked pronouns, as Lambek (1958) already pointed out. With \(s/(np)s\) as the lexical type assignment for ‘he/’she’, but \((np)s/(np)s\) for ‘him/’her’, we correctly rule out ‘him irritates she’ while allowing ‘he irritates her’.

Elementary theorems for the unary type-forming operations are established in (6).

\[
\begin{align*}
\Delta A & \rightarrow \Delta A \quad (\text{Ax}) \\
\Delta A & \rightarrow A \quad (Ax) \\
\Delta A & \rightarrow \Delta A \quad (\text{R}0 \rightarrow) \\
A & \rightarrow \Delta AD \quad (\text{R}0 \rightarrow)
\end{align*}
\]

An illustration of the added expressivity of the unary operators can be found in Bernardi (2002), where they are used to control the distribution of polarity sensitive items. Consider the contrast between ‘Nobody left yet’ with the negative polarity item ‘yet’ and ‘Somebody left yet’. In a type language with just the binary type-forming operations, both ‘somebody’ and ‘nobody’ would receive the subject type \(s/(np)s\), and ‘yet’ the modifier type \((np)s/(np)s\). Such type assignment is too crude to block the ungrammatical ‘Somebody left yet’. In the extended type language, the negative polarity trigger ‘nobody’ can be assigned the type \(s/(np)s\), whereas ‘somebody’ keeps the undecorated type \(s/(np)s\). By typing the negative polarity item ‘yet’ as \((np)s/(np)s\) one expresses the fact that it requires a trigger such as ‘nobody’ to check the \(\Delta \) decoration in its numerator subtype. For the derivation of the simple sentence ‘nobody left yet’ (with no polarity item to be checked), we rely on the fact that in the base logic, we have \(s/(np)s \rightarrow s/(np)s\), i.e. the \(\Delta \) decoration on argument subtypes can be simplified away, allowing the combination (in terms of the Application schema) of ‘nobody’ with a simple verb phrase ‘left’ of type \(np\).

Monotonicity properties. Apart from these theorems, the base logic has (7) as derived rules of inference. With respect to the derivability relation, the operations \(\varnothing\) and \(\Delta\) are order-preserving (isotone). The \(\bullet\) operation is order-preserving in its two arguments; the division operations \(/\) and \(\backslash\) are order-preserving in their numerator, and order-reversing (antitone) in their denominator argument.

\[
A \rightarrow B \quad \text{implies} \quad \varnothing A \rightarrow \varnothing B \\
A \bullet C \rightarrow B \bullet C \\
(7)
\]

From a combinatorial point of view, these rules produce an infinite number of type transformations from some small inventory of ‘primitive’ ones. Consider the Lifting schema. From it, one obtains the transformations known as Value Raising (for example, lifting a determiner type \(np\) to \((s/(np)s)/(np)s\)) and Argument Lowering (for example, lowering a third-order verb phrase type \((s/(np)s)/(np)s\) to first-order \(np\)).

Alternative presentations, Natural Deduction. The categorial base logic allows many alternative axiomatizations, each serving its own function. The essential point is that the different presentations must find their justification in the modeltheoretic interpretation of the connectives, i.e. one has to prove they are equivalent syntaxes for performing valid type computations. In the Gentzen sequent calculus, one replaces the arrows \(A \rightarrow B\) by statements \(\Gamma \Rightarrow \Delta (\text{structure } \Gamma \text{ is of type } B).\) The antecedent \(\Gamma\) is built out of formulas by means of the structure-building operations \(\langle \rangle\) and \((\cdot)\). The purpose of this presentation is to show that the transitivity rule (the Cut rule) can be eliminated. Every logical rule of inference in the Gentzen calculus introduces a connective either in the antecedent or in the succedent, so that backward-chaining, cut-free proof search immediately yields a decision procedure for categorial derivability, as shown in (Lambek 1958) for the binary and (Moortgat 1996) for the unary connectives.

The derivational format of Combinatory Categorial Grammar (CCG, (Steedman 2000b) and references cited there) is a Hilbert-style presentation. Functional Application here is taken as the basic, primitive schema for type combination. To the Application schema are added extra schemata, such as Lifting, the combinator \(T\). The CCG format of derivations is related to the Gentzen style as the combinator presentation of intuitionistic logic is to its Gentzen presentation. The recursive generalization of the primitive type transformations under monotonicity is important for such ‘combinatory’ presentations of categorial derivability: without this generalization, one loses completeness.

In a third format, Natural Deduction (ND), every type-forming connective has an introduction and an elimination rule. As a result, ND doesn’t have the pleasant proof search properties of the Gentzen calculus, but it is a perspicuous presentation of a derivation once it has been found. For this reason, ND is often used in linguistic discussion of categorial analyses. Also, ND is the most transparent format to associate meaning assembly with a derivation, as we will see in §3. We present the ND rules for the base logic below, using the Gentzen sequent style, which is explicit about the structural configuration of the antecedent assumptions.

\[
\begin{align*}
\Gamma \vdash \varnothing A & \quad (\varnothing E) \\
\Gamma \vdash A & \quad (\varnothing I)
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash \Delta & \quad (\Delta I) \\
\Gamma \vdash [A] & \quad (\Delta E)
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash B \vdash A & \quad (\varnothing B) \\
\Gamma \vdash \Delta & \quad B \vdash A \\
\Gamma \vdash A/B & \quad (\varnothing B)
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash A \rightarrow [A] & \quad (\varnothing A) \\
\Gamma \vdash [A] & \quad (\varnothing B)
\end{align*}
\]

\[
\begin{align*}
|\begin{array}{c}
\Gamma \vdash A \\
\Delta \vdash B
\end{array}| & \quad (\varnothing A/B) \\
\Gamma \vdash A \rightarrow B & \quad (\varnothing A/B)
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash A & \quad (\varnothing A/B) \\
\Gamma \vdash [A] & \quad C
\end{align*}
\]

Figure 1: Natural deduction. Notation: \(\Gamma \vdash A\) for the deduction of a conclusion \(A\) from a configuration of assumptions \(\Gamma\). Axioms: \(A \vdash A\). Antecedent structures are built from formulas with the structure-building operations \(\langle \rangle\) and \((\cdot)\). These are the structural counterparts of \(\varnothing\) and \(\bullet\), respectively, as the \(\varnothing\), \(\bullet\) introduction rules show.

Multimodal generalization. One can straightforwardly generalize the base logic to a system where one has not just one single merge and feature checking relation, but families of them. In modal logic terms, this means moving from a unimodal to a multimodal system, with frames \(\langle W, \{R^1, \ldots, R^n\}\rangle\) where the different relations are kept apart by indexing them with a composition mode label. Similarly, in the formula language, we index the connectives for these composition modes. The concept of multiple composition modes is not unfamiliar. For the binary operations, one can think of a distinction between the structure of words (morphology) and the structure of phrases (syntax): one can give a categorial analysis of morphology and syntax in terms of /\(\varnothing\), \(\bullet\) versus \(\varnothing\), \(\bullet\). For the unary connectives \(\varnothing\), \(\Delta\), multimodality makes it possible to distinguish a number of named features in the grammar, so that they can play different roles in controlling composition.
The multimodal perspective turns out to be particularly useful once we move beyond the base logic and consider its structural extensions, where one can have interaction between different binary composition modes (between morphology and syntax, in the case of complement inheritance, for example), and between specific unary control features and binary composition operations. Such interaction principles are discussed below.

2.2 The structural module

The laws of the base logic do not depend on specific structural properties of the ‘Merge’ and feature-checking relations: the completeness theorem (5) does not impose any restrictions on the interpretation of $R_{\text{M}}$ and $R_{\text{C}}$. In this sense, the base logic can be said to capture the invariances of grammatical composition. Although the base logic already has a rich deductive structure, the system also has its limitations. If an expression can occur in different structural configurations, one would like to relate these configurations. In the base logic, this cannot be done: type assignment is structurally rigid, in the sense that different structural environments will lead to different type assignments. To overcome the problem of structural rigidity, one extends the base logic with facilities for structural reasoning. Technically, such facilities have the status of non-logical axioms, or postulates. They can be introduced in a global, or in a controlled fashion. We discuss these in turn.

Global structural rules. The postulates in (8) create a hierarchy of categorial systems: adding structural options, flexibility of type combination increases, but structural determination deteriorates.

\[
\begin{align*}
& (A \bullet B) \bullet C \rightarrow A \bullet (B \bullet C) \quad \mathcal{A}_1 \\
& (A \bullet B) \bullet C \rightarrow (A \bullet C) \bullet B \quad \mathcal{A}_2
\end{align*}
\]

(8)

The rebracketing postulates $\mathcal{A}_1$ and $\mathcal{A}_2$ added to the /, \ fragment of the base logic, produce the system known as L, the associative calculus of (Lambek 1958). The /, \ fragment of the base logic itself is known as NL: in (Lambek 1961) this systems was obtained by dropping the associativity postulates from $\mathcal{L}$. Characteristic theorems of $\mathcal{L}$ are the type transitions in (9): the Geach laws $G_0$, $G_2$, and the functional composition schemata (known as combinator $B$ in CCG) of which $B_0$, $B_2$ are the simplest forms.

\[
\begin{align*}
& G_0: A/B \rightarrow (A/C)/(B/C) \quad B_0: (A/B)/(B/C) \rightarrow A/C \quad (C/B) \bullet (B/A) = C \quad B_2
\end{align*}
\]

(9)

Adding the commutativity postulate to $\mathcal{L}$ produces $\mathcal{L}P$ (Lambek calculus with permutation), a system coinciding with the multiplicative fragment of linear logic, which has a commutative product operation matched by a single linear implication. The distinction between left-incompleteness and right-incompleteness collapses in the presence of $C$.

Extending the base logic with facilities for structural reasoning has consequences for the interpretation of the type-forming operations, discussed in (Doi 1992; Kurtonina 1995). An interpretation with respect to arbitrary frames, obviously, is not available any more. Instead, each postulate introduces a corresponding frame constraint restricting the interpretation of the Merge relation $R_{\text{M}}$ and completeness is stated with respect to frames respecting the relevant constraints.

A Commutativity postulate, for example, would impose the semantic constraint that for all $x, y, z \in W$, $R_{\text{Cxy}}R_{\text{Cyz}}$ implies $R_{\text{Czy}}R_{\text{Cyx}}$. Similarly for the other postulates discussed. In the presence of such semantic constraints, it will often be the case that one can specialize the abstract relational interpretation to more concrete models. A good example is the system L with its associative composition relation $R_{\text{C}}$. In this case, one can read $R_{\text{Cxy}}$ as concatenation, i.e. $x = y = z$. Pentus (1994) proves that L indeed is complete with respect to this concatenation interpretation.

Controlled structural reasoning There are many natural language phenomena that seem to require some of the flexibility offered by the postulates (8). Cases of non-constituent coordination can be naturally handled with the possibilities for type-combination that follow from the rebracketing postulates. Displacement phenomena are ubiquitous in natural language, and seem to require some form of commutativity. At the same time, it is clear that in a global form, these structural options overgenerate. Commutativity would entail that well-formedness is preserved under arbitrary changes in word order; free rebracketing makes constituent structure irrelevant for determining grammaticality.

To obtain controlled structural extensions of the base logic, various strategies have been pursued. In the rule-based approach of Combinatory Categorial Grammar, one augments the Application/Lifting basis with structural combinators which, in an unconstrained form, would be overgenerating. One then imposes type-restrictions on these extra combinators. In addition, the set of rule schemata (combinators) is kept finite, so that one can avoid the consequences of the recursive generalization of rules under monotonocity. The alternative is to exploit the intrinsic logical instruments for structural resource management offered by richer type systems with unary control features and multimodal interaction principles. To compare these two strategies, consider the following cases of extraction.

\[
\begin{align*}
& \text{a. what Alice found} \\
& \text{b. what Alice found there}
\end{align*}
\]

Figure 2: Wh-extraction: combinator-style derivation. The clause body ‘Alice found there’ is assigned type s/np by means of the backwards crossed composition combinator $B_{1,s}$. The rule can apply because the cancelled $(np/s)$ satisfies the type-restriction on $B_{1,s}$.

In CCG, the peripheral case of extraction (10a) are derived from an assignment wh/$(s/np)$ to the wh-pronoun by lifting the type for ‘Alice’ to $(s/(np/s))$ which is then composed with the transitive verb type $(np/s)/np$ for ‘found’ by means of $B_0$. To obtain the non-peripheral case of extraction in (10b), one needs the combinator $B_{1,x}$, a form of composition which depends on the commutativity postulate. To avoid collapse into LP, one imposes a side-condition on the rule, restricting the middle term $B$ to certain verbal categories, in this case $(np/s)$.

\[
\begin{align*}
& B_{1,s} \quad (B/C) \bullet (B/A) \rightarrow A/C \quad \text{where } B \text{ is a predicate category}
\end{align*}
\]

(11)

The $\otimes$ connectives make it possible to avoid extra-logical type-restrictions. The postulates $P1/P2$ below implement a controlled form of rebracketing and reordering for formulas carrying the $\otimes$ control feature, as shown in (Moortgat 1999). With a lexical type assignment $wh/(s/\otimes np)$ to the wh-pronoun, one obtains peripheral and medial extraction from right branches. Under this analysis, one does not attribute any associativity/commutativity to the operation itself; displacement effects arise through the interaction of the Merge operation with a gap hypothesis carrying the licensing $\otimes$ feature. A derivation is given in Figure 3.

\[
\begin{align*}
& P1 \quad (A \bullet B) \bullet (\otimes C) \rightarrow (A \bullet (\otimes C)) \bullet B \\
& P2 \quad (A \bullet B) \bullet (\otimes C) \rightarrow A \bullet (B \bullet (\otimes C))
\end{align*}
\]

(12)
2.4 Language learning

Kanazawa (1998) has studied formal learning theory for categorial grammar within Gold’s paradigm of identification in the limit on the basis of positive data. The focus is on classical categorial grammars, using only the Application rules, and on combinatory extensions with extra rule schemata. On the input side, Kanazawa considers both learning from strings, and from function-argument structures. On the output side, the class of rigid grammars (where the grammar assigns a unique type to each word) is compared with the class of $k$-valued grammars (where at most $k$ types are assigned to a lexical item). It is a matter of dispute whether Gold’s very abstract formulation of the learning problem is directly relevant for first language acquisition. An alternative purely inductive approach, learning a subclass of the shallow context-free languages, is presented in (Adrians 2002).

The discussion in the previous section suggests some directions for further research in this area. First of all, one would like to obtain learnability results for classes of Lambek-style categorial grammars, where the learner has access to both the Elimination rules and the Introduction rules for the type-forming operators. Secondly, one would like to go beyond systems with a hard-wired structural component, in order to investigate the learnability effects of different choices of structural packages, in combination with an invariant base logic. The work of Fock (2001) is promising in this respect: she mixes unification/substitution with Lambek-style deduction, suggesting modulation of learnability questions in terms of different structural postulates. Finally, the role of semantic information in learning needs further investigation. The challenge here is to find a level of informativity that would be realistic in the setting of first language acquisition.

3 Meaning assembly: the Curry-Howard correspondence

Categorial grammar adheres to the truth-conditional theory of semantics: the interpretation process establishes a systematic relationship between linguistic expressions and states of affairs in the world in such a way that specifying the meaning of a sentence comes down to giving its truth conditions. As in the previous section, model theory provides the tools to carry out this program. For semantic interpretation this involves the construction of a set-theoretic model of ‘the world’ in terms of objects and configurations of such objects; these set-theoretic constructs then serve as the semantic values of natural language expressions.

The integrated treatment of syntax and semantics, which is now seen as the most attractive aspect of categorial grammar, is of relatively recent origin. The original Lambek systems (Lambek 1958; Lambek 1961) were presented as syntactic type calculi. The correctness of Chomsky’s conjecture that context-free equivalence extends to the Lambek calculus was finally established in (Pentus 1993). This result does not have a direct corollary for polynomial parsability, because the construction of a context-free grammar from an I grammar is of exponential complexity.

For the structural extensions of the base logic discussed in §2.2, the challenge is to identify appropriate constraints: it is clear that arbitrary combinator extensions, or structural rule packages, lead to excessive expressivity. But Vijay-Shanker and Weir (1994) show that an appropriately restricted version of CCG is weakly equivalent to the linear indexed grammars, hence polynomially parsable. In a similar spirit, Moortgat (2002) shows how with appropriate restrictions on lexical assemblages and structural postulates, one can carve out a class of multimodal categorial grammars equivalent with Lexicalized Tree Adjoining Grammars and inheriting the polynomial parsability of these systems. The general theory of $\tilde{\Omega}$ as a control operators has been investigated in (Kurotina and Moortgat 1997). These authors establish a number of embedding theorems showing that the full logical space between the base logic and LP can be navigated in terms of the control connectives, both in the ‘licensing’ direction illustrated above (allowing structural inferences that would be unavailable without the control features) and in the ‘constraining’ sense (blocking structural options that would be licit in the absence of the control features).

More important than weak generative capacity are issues of strong capacity, which in the categorial tradition would mean the proof structures (or their lambda terms, discussed in §3) that produce a certain string. In this area, Tiede (2001) has obtained interesting results, showing that while the Lambek systems (NJL are weakly CF, their expressivity in terms of strong capacity goes beyond that of CF grammars.

Figure 3: Wh-extraction: $\tilde{\Omega}$ control. The type-assignment to the relativizer ‘what’ expresses the fact that the relative clause body is a sentence built with the help of a ‘gap’ hypothesis of type $\square \langle \text{np} \rangle$. The feature-marked hypothesis has to be withdrawn at the right periphery, but it is not selected in that position. It is related to the non-peripheral direct object position within the relative clause body by virtue of the postulates P1 and P2. Once it has found the direct object position, the licensing feature $\tilde{\Omega}$ has done its work and can be cleaned up by the law $\square \text{np} \rightarrow \text{np}$.

The ‘gap’ hypothesis is then used as a regular direct object with respect to the selecting verb ‘found’.

3.1 An introduction to categorical realizability

The view on grammatical invariants and structural variation invites a comparison between the categorial landscape and the Chomskyan hierarchy. For a recent survey, see (Bazukowski 1997). The discovery in the Eighties of dependency patterns that cannot be adequately captured by context-free grammars has led to an interest in ‘mildly context-sensitive’ formalisms, i.e. systems with an expressivity beyond context-free, but sufficiently restricted to have polynomial parsing algorithms. The classical Ajdukiewicz-Bar-Hillel grammars have long been known to be weakly equivalent to context-free grammars, hence to be too poor to serve as models of Universal Grammar. The same is true for the base logic described in §2.1 (7). The correctness of Chomsky’s conjecture that context-free equivalence extends to the Lambek calculus was finally established in (Pentus 1993). This result does not have a direct corollary for polynomial parsability, because the construction of a context-free grammar from an I grammar is of exponential complexity.

The general theory of $\tilde{\Omega}$ as a control operators has been investigated in (Kurotina and Moortgat 1997). These authors establish a number of embedding theorems showing that the full logical space between the base logic and LP can be navigated in terms of the control connectives, both in the ‘licensing’ direction illustrated above (allowing structural inferences that would be unavailable without the control features) and in the ‘constraining’ sense (blocking structural options that would be licit in the absence of the control features).

More important than weak generative capacity are issues of strong capacity, which in the categorial tradition would mean the proof structures (or their lambda terms, discussed in §3) that produce a certain string. In this area, Tiede (2001) has obtained interesting results, showing that while the Lambek systems (NJL are weakly CF, their expressivity in terms of strong capacity goes beyond that of CF grammars.

Figure 3: Wh-extraction: $\tilde{\Omega}$ control. The type-assignment to the relativizer ‘what’ expresses the fact that the relative clause body is a sentence built with the help of a ‘gap’ hypothesis of type $\square \langle \text{np} \rangle$. The feature-marked hypothesis has to be withdrawn at the right periphery, but it is not selected in that position. It is related to the non-peripheral direct object position within the relative clause body by virtue of the postulates P1 and P2. Once it has found the direct object position, the licensing feature $\tilde{\Omega}$ has done its work and can be cleaned up by the law $\square \text{np} \rightarrow \text{np}$.

The ‘gap’ hypothesis is then used as a regular direct object with respect to the selecting verb ‘found’.

2.3 Generative capacity and computational complexity

The modular view on grammatical invariants and structural variation invites a comparison between the categorial landscape and the Chomskyan hierarchy. For a recent survey, see (Bazukowski 1997). The discovery in the Eighties of dependency patterns that cannot be adequately captured by context-free grammars has led to an interest in ‘mildly context-sensitive’ formalisms, i.e. systems with an expressivity beyond context-free, but sufficiently restricted to have polynomial parsing algorithms. The classical Ajdukiewicz-Bar-Hillel grammars have long been known to be weakly equivalent to context-free grammars, hence to be too poor to serve as models of Universal Grammar. The same is true for the base logic described in §2.1 (7). The correctness of Chomsky’s conjecture that context-free equivalence extends to the Lambek calculus was finally established in (Pentus 1993). This result does not have a direct corollary for polynomial parsability, because the construction of a context-free grammar from an I grammar is of exponential complexity.

The structural extensions of the base logic discussed in §2.2, the challenge is to identify appropriate constraints: it is clear that arbitrary combinator extensions, or structural rule packages, lead to excessive expressivity. But Vijay-Shanker and Weir (1994) show that an appropriately restricted version of CCG is weakly equivalent to the linear indexed grammars, hence polynomially parsable. In a similar spirit, Moortgat (2002) shows how with appropriate restrictions on lexical assemblages and structural postulates, one can carve out a class of multimodal categorial grammars equivalent with Lexicalized Tree Adjoining Grammars and inheriting the polynomial parsability of these systems. The general theory of $\tilde{\Omega}$ as a control operators has been investigated in (Kurotina and Moortgat 1997). These authors establish a number of embedding theorems showing that the full logical space between the base logic and LP can be navigated in terms of the control connectives, both in the ‘licensing’ direction illustrated above (allowing structural inferences that would be unavailable without the control features) and in the ‘constraining’ sense (blocking structural options that would be licit in the absence of the control features).

More important than weak generative capacity are issues of strong capacity, which in the categorial tradition would mean the proof structures (or their lambda terms, discussed in §3) that produce a certain string. In this area, Tiede (2001) has obtained interesting results, showing that while the Lambek systems (NJL are weakly CF, their expressivity in terms of strong capacity goes beyond that of CF grammars.

2.4 Language learning

Kanazawa (1998) has studied formal learning theory for categorial grammar within Gold’s paradigm of identification in the limit on the basis of positive data. The focus is on classical categorial grammars, using only the Application rules, and on combinatory extensions with extra rule schemata. On the input side, Kanazawa considers both learning from strings, and from function-argument structures. On the output side, the class of rigid grammars (where the grammar assigns a unique type to each word) is compared with the class of $k$-valued grammars (where at most $k$ types are assigned to a lexical item). It is a matter of dispute whether Gold’s very abstract formulation of the learning problem is directly relevant for first language acquisition. An alternative purely inductive approach, learning a subclass of the shallow context-free languages, is presented in (Adrians 2002).

The discussion in the previous section suggests some directions for further research in this area. First of all, one would like to obtain learnability results for classes of Lambek-style categorial grammars, where the learner has access to both the Elimination rules and the Introduction rules for the type-forming operators. Secondly, one would like to go beyond systems with a hard-wired structural component, in order to investigate the learnability effects of different choices of structural packages, in combination with an invariant base logic. The work of Fock (2001) is promising in this respect: she mixes unification/substitution with Lambek-style deduction, suggesting modulation of learnability questions in terms of different structural postulates. Finally, the role of semantic information in learning needs further investigation. The challenge here is to find a level of informativity that would be realistic in the setting of first language acquisition.
3.1 Model-theoretic semantics, type theory and the lambda calculus

For semantic interpretation, we associate every type \( A \) with a semantic domain \( D_A \). Expressions of type \( A \) find their denotations in \( D_A \). Semantic domains can be set up in two ways: directly on the basis of the types as discussed in the previous section, or indirectly, via a mapping from syntactic to semantic types. The indirect option is attractive for a number of reasons. On the level of atomic types, one may want to make different basic distinctions depending on whether one uses syntactic or semantic criteria. For complex types, a map from syntactic to semantic types makes it possible to forget information that is relevant only for the way expressions are to be configured in the form dimension. Finally, the semantic type system naturally fits the language of the typed lambda calculus, which we can then use, together with its standard interpretation, to specify the instructions for meaning assembly.

Semantic and syntactic types. For a simple extensional interpretation, the set of atomic semantic types \( \text{SemAtom} \) could consist of types \( e \) and \( t \), with \( D_t \) the domain of discourse (a non-empty set of entities, objects), and \( D_e = \{0, 1\} \), the set of truth values. The full set of semantic types \( \text{SemType} \) is then obtained by closing \( \text{SemAtom} \) under the rule that if \( A \) and \( B \) are in \( \text{SemType} \), then \( A \to B \) is also. \( D_{A \to B} \), the semantic domain for a functional type \( A \to B \), is the set of functions from \( D_A \) to \( D_B \). The mapping from syntactic to semantic types \( (\cdot)^* \) could now stipulate for basic syntactic types that \( n^* = e \), \( s^* = t \), and \( n^* = (e \to t) \). Sentences, in this way, denote truth values; \( (\cdot)^* \) is the mapping from individuals; common nouns functions from individuals to truth values. For complex syntactic types, we set \((A/B)^* = (B/A)^* = B^* \to A^* \). On the level of semantic types, the directionality of the slash connective is no longer taken into account. The distinction between numerator and denominator — domain and range of the interpreting functions — is kept. Notice that both verb phrases with syntactic type \( n p^* \) and common nouns are mapped to the semantic type \( e \to t \).

The language of the simply typed lambda calculus. In §3.2, we will present a procedure to associate a derivation \( A_1, \ldots, A_n \to B \) with a term \( t \) of type \( B \) representing a recipe for meaning assembly with parameters \( x_1, \ldots, x_k \) for the lexical assumptions \( A_1, \ldots, A_n \). To prepare the ground, we build up the set of meaningful expressions (terms) of semantic type \( A \), starting from a denumerably infinite set of variables for each type. For each expression \( t \) of type \( A \), we specify its interpretation \( [t]^h \) relative to an assignment function \( g \) which assigns to each variable of type \( A \) a member of \( D_A \).

Variables. Let \( x \) be a variable of type \( A \). Then \( x \) is a term of type \( A \). Interpretation: \([x]^h = g(x)\).

Application. Let \( t \) and \( u \) be terms of type \( A \to B \) and \( A \) respectively. Then \( (t \ u) \) is a term of type \( B \). Interpretation: \([t \ u]^h = [t]^h \cdot [u]^h\), i.e. the value one obtains when applying the function \([t]^h\) to \([u]^h\).

Abstraction. Let \( x \) be a variable of type \( A \) and \( t \) a term of type \( B \). Then \( \lambda x.t \) is a term of type \( A \to B \). Interpretation: \([\lambda x.t]^h b = [t]^h \), such that \( b \) is a member of \( D_A \). The assignment function \( g \) is the assignment that is exactly like \( g \) except for the possible difference that it assigns the object \( k \) to the variable \( x \).

Given this interpretation, certain equalities hold between terms. One can see them as syntactic simplifications, replacing a more complex term (the redex) by a simpler one with the same interpretation (the contractum).

\[
(\lambda x.t) \ u \sim_{\beta} t[u/x] \quad \text{provided } u \text{ is free for } x \text{ in } t
\]

\[
\lambda x.(t \ x) \sim_{\beta} t \quad \text{provided } x \text{ is not free in } t
\]

3.2 Formulas-as-types, proofs as programs

Curry's basic insight was that one can see the functional types of type theory as logical implications, giving rise to a one-to-one correspondence between typed lambda terms and natural deduction proofs in positive intuitionistic logic. A natural deduction presentation for \( \rightarrow \) starts from identity axioms \( A \vdash A \) and has the introduction and elimination rules below, where \( \Gamma, \Delta \) represent finite lists of formulas, and where \( \Gamma \vdash A \) results from dropping, some or all occurrences of \( A \) from \( \Gamma \).

\[
\frac{}{\Gamma, A \vdash B} \quad \frac{}{\Gamma, \Delta \vdash B} \quad \frac{}{\Gamma, \Delta \vdash B} \quad \frac{}{\Gamma, \Delta \vdash B} \quad \frac{}{\Gamma, \Delta \vdash B}
\]

Let us write \( \Gamma(t) \) for the string of types of free occurrences of variables in a term \( t \). Each term \( t \) of type \( A \) now encodes a natural deduction proof of the sequent \( \Gamma(t) \vdash A \). The Variable clause in the definition of well-formed terms corresponds to the axiom sequent, the Application clause to \( \rightarrow \) Elimination, and the Abstraction clause to \( \rightarrow \) Introduction, where the dropped \( A \) assumption corresponds to the variable bound by the lambda abstractor. In the opposite direction, every natural deduction proof is encoded by a lambda term. The normalization of natural deduction proofs corresponds to the \( \beta/\eta \) reductions of terms.

Translating Curry's formulas-as-types’ idea to the categorial type logics we are discussing, we have to take the differences between intuitionistic logic and the grammatical resource logic into account. Below we repeat the natural deduction presentation of the base logic, now taking term-decorated formulas as basic declarative units. Judgements take the form of sequents \( \Gamma \vdash t : A \).

The antecedent \( \Gamma \) is a structure with labels \( x_1 : A_1, \ldots, x_n : A_n \). The \( x_i \) are unique variables of type \( A_i^* \), where \( (\cdot)^* \) is the mapping from syntactic to semantic types. The succedent is a term \( t \) of type \( A^* \) with exactly the free variables \( x_1, \ldots, x_n \), representing a program which gives inputs \( k_1, \ldots, k_n \) produces \( t \) under the assignment that maps the variables \( x_i \) to the objects \( k_i \). The \( x_i \) in other words are the parameters of the meaning assembly procedure. A derivation starts from axioms \( x : A \vdash x : A \). The Elimination and Introduction rules have a version for the right and the left implication. On the meaning assembly level, this syntactic difference is ironed out, as we already saw that \((A/B)^* = (B/A)^*\). As a consequence, we don’t have the isomorphic (one-to-one) correspondence between terms and proofs of Curry’s original program. But we do read off meaning assembly from the categorical derivation.

\[
([t]^h) \quad \frac{\Gamma \vdash t : A}{\Gamma \vdash f \circ t : A/B} \quad \frac{\Gamma \vdash f : A/B}{\Gamma \vdash \Delta \vdash (f u) : A}/[E]
\]

\[
([t]^h) \quad \frac{\Gamma \vdash u : B}{\Gamma \vdash \Delta \vdash (f u) : A}/[E]
\]

Figure 4. Natural Deduction rules: term labeling.

A second difference between the programs/computations that can be obtained in intuitionistic implicational logic, and the recipes for meaning assembly associated with categorical derivations has to do with the resource management of assumptions in a derivation. The formulation of the \( \rightarrow \) introduction rule makes it clear that in intuitionistic logic, the number of occurrences of assumptions (the ‘multiplicity’ of the logical resources) is not critical. One can make this style of resource management explicit in the form of structural rules of Contraction and Weakening, allowing for the duplication and waste of resources.

\[
\Gamma, A, A \vdash B \quad \frac{}{\Gamma, A \vdash B} \quad \frac{}{\Gamma, A \vdash B} \quad \frac{}{\Gamma, A \vdash B}
\]

In contrast, the categorical type logics are resource sensitive systems where each assumption has to be used exactly once. At the level of LP, we have the following correspondence between resource constraints and restrictions on the lambda terms coding derivations.
1. no empty antecedents: each subterm contains a free variable;
2. no Weakening: each λ operator binds a variable free in its scope;
3. no Contraction: each λ operator binds at most one occurrence of a variable in its scope.

Moving from LP to the grammatical base logic imposes even tighter restrictions on binding: in the absence of Associativity and Commutativity, the slash introduction rules responsible for the λ operator can only reach the immediate daughters of a structural domain.

3.3 The syntax/semantics interface

Applied to the composition of natural language meaning, the ‘proofs-as-programs’ approach has some interesting consequences for the syntax/semantics interface.

A first point to notice is the strictly modular treatment of derivational versus lexical semantics. The proof term that is read off a derivation is a uniform instruction for meaning assembly that fully abstracts from the contribution of the particular lexical items on which it is built. As a result, no assumptions about lexical semantics can be built into the meaning assembly process as represented by a derivation. We illustrate the interplay between lexical and derivational semantics in Figures 5 and 6. Whereas the proof term in Fig 5 is a faithful encoding of the derivation (modulo directionality and structural operations), the term one obtains in Fig 6 after substitution of lexical meaning programs and β simplification has lost the transparency with respect to the derivation.

![Figure 5: Computation of the proof term for the pattern 'Noun that Subj Transitive-Verb'. Leaps are labeled with variables. The derivation produces a meaning recipe with parameters for the lexical meaning programs. The recipe can be applied to any particular choice of lexical items fitting the type requirements: ‘biscuit that Alice ate’, ‘book that Carroll wrote’, etc. The second feature is the limited semantic expressivity of a structure-sensitive type logic: many forms of meaning assembly that can be straightforwardly expressed in the language of the lambda calculus cannot be obtained as Curry-Howard images of the introduction/elimination inferences of the categorical base logic. To resolve the tension between structure-sensitivity and semantic expressivity, categorial grammars can exploit a combination of two strategies. Structural reasoning (in terms of combinators or structural postulates) makes it possible to explicitly determine which positions are accessible for semantic manipulation (binding). The example of controlled sub-extraction in Figure 3 is an illustration. Secondly, lexical meaning programs do not have to obey the resource constraints of the derivational semantics. Specifically, we do not impose the single-bind condition on lexical meanings (although the ban on vacuous abstraction does make sense, also in the lexicon.) An example of multiple binding is the lexical lambda term for the relative pronoun ‘that’ in Figure 6, a program which computes property interaction. Another example would be a reflexive pronoun like himself. With a type \(\langle(np(s))/np\rangle/(np(s))\), it consumes its transitive verb argument in a...
3.4 Processing issues

The interpretation procedure discussed above is essentially dynamic: interpretations are assembled on the fly in the course of the derivation process, rather than being computed post hoc from a given static structure. This has led to a distinctly 'categorical’ view on processing issues.

Incrementality, information structure. The flexible notion of derivational constituency en-
gendered by type-changing principles makes left-to-right parsing directly compatible with incre-
mental interpretation. The resulting categorial modeling of natural language processing has been
worked out in (Steedman 2000a). This work shows that derivational constituency is guided by
prosodic articulation (intonation contour). To do justice to this dimension of grammatical organi-
zation, one needs a richer notion of semantic interpretation, accommodating notions of focus and
information structure. Steedman’s proposals are formulated in the CCG style; Hendrigh (1999)
analyses information packaging and intonation contour in multimodal type-logical terms.  

Proof nets. A novel computational view on natural language processing derives from the proof-
net approach. Proof nets were originally developed in the context of Linear Logic, where they
elegantly capture the essence of resource-sensitive derivations in graph-theoretical terms. Moot
and Puite (2002) refine the proof net techniques for use with the grammatical type logics discussed
in this article, where apart from resource multiplicity also structural patterns have to be taken
into account.  

Johnson (1998) and Morrill (2000) have pointed out that proof nets offer an attractive perspec-
tive on performance phenomena. A net can be built in a left-to-right incremental fashion by
establishing possible linkings between the input/output connectors of lexical items as they are
presented in real time. This suggests a simple complexity measure on a traversal, given by the
number of unresolved dependencies between literals. This complexity measure on incremental
proof net construction makes the right predictions about a number of well-known processing is-
sues, such as the difficulty of center embedding, garden path effects, attachment preferences, and
preferred scope constraints in ambiguous constructions. An illustration is presented in Figure 7.

4 Exploration

4.1 Variants and alternatives

Pregroup grammars. An interesting variation on the categorial theme has been developed by
Jim Lambek in a number of recent papers (Lambek 1998; Lambek 2001). The approach makes use
of pregroups, algebraic structures closely related to the residuation-based models for the origi-
nal categorial type systems discussed below. A pregroup is a partially ordered monoid in which each element a
has a left and a right adjoint, aL, aR, satisfying aL a → 1 → aR a and aR a → 1 → aL a, respectively.
Type assignment takes the form of associating a word with one or more elements from the free
pregroup generated by a partially ordered set of basic types. For the connection with categorial
type formulas, one can use the translations a/b = aR b and b/a = aL b. Parsing, in the pregroup
setting, is extremely straightforward. Lambek (1999) proves that one only has to perform the
contractions replacing aL a and aR a by the multiplicative unit 1. This is essentially a check for
well-bracketing—an operation that can be entrusted to a pushdown automaton. The expansions
1 → aL a and 1 → aR a are needed to prove equations like (aL b) → bR aL. We have used the latter to
obtain the pregroup version of the higher-order relative pronoun type (n/a)/(s/np) in the example
below.

<table>
<thead>
<tr>
<th>CATEGORY TYPES</th>
<th>book that</th>
<th>Carroll wrote</th>
</tr>
</thead>
<tbody>
<tr>
<td>n aL (n)/(a)/(s/np)</td>
<td>np (np)/(s)/(np)</td>
<td></td>
</tr>
</tbody>
</table>

| PREGROUP ASSIGNMENT | n aL n aL s | s | up | np | np |

Comparing the pregroup approach with the original categorial type system, one notices that
the pregroup notation has associativity built in. This has pleasant consequences. In the standard
Lambek calculus, the choice between (np)/(s)/(np) and (np)/(s)/(np) as the lexical type assignment for a
transitive verb is in a certain sense arbitrary, given the fact that the associativity postulates make
these types interderivable. The pregroup-category format removes this notational overspecification:
the two types translate to (np)/(s)/(np). In general, every sequent derivable in the Lambek calculus
will be derivable in the corresponding pregroup. The converse is not true: the pregroup image of
the types (a•b)/(c) and (a•b)/(c), for example, is a•b, but these two types are not interderivable in L.

With respect to generative capacity, Buszkowski (2001) shows that the pregroup grammars
are equivalent to context-free grammars. They share, in other words, the expressive limitations of
the original categorial grammars. To overcome these limitations in the analyses of German word
order and Romance clitics referred to above, the authors rely on a combination of metaregular and
derivational constraints.

Minimalist grammars. Whereas the Chomskyan tradition of generative grammar and the
categorial tradition have been moving in separate orbits for a long time, there are surprising
convergences between resource-sensitive logics and Chomsky’s recent ‘Minimalist Program’ when
this is made mathematically precise, as in the algebraic formulation of (Stabler 1997; Stabler 1999).
A minimalist grammar, in this format, consists of a lexicon of type assignments, closed under the
structure-building operations Merge and Move. Type declarations are built up out of two sets of
features with matching input/output polarities: category features and control features. The
former govern the Merge operation, in which one easily recognizes the Modus Ponens/Application
rule of categorial deduction. The control features explicitly license structural reasoning (Move),
much like the unary multiplicatives O, O. The Stabler grammars have been shown to be weakly
equivalent to Multiple Context Free Grammars, hence to fall within the class of mild context-
sensitive formalisms. Comparing them with categorial logics, one notices that the minimalist category concept is
effectively first-order: no use is made of hypothetical reasoning with respect to Merge. The
restriction to Modus Ponens doesn’t seem to be an essential limitation of the minimalist design,
however. It would be interesting to extend minimalist grammars with facilities for hypothetical
reasoning, which, as we have seen above, plays such a central role in the meaning assembly process.

4.2 Further reading

The Supplementary References provide material for further exploration. We present brief guide-
lines below.

The history of categorial grammar is generally traced back to the work of Ajdukiewicz in
the Thirties (Ajdukiewicz 1935), which was later taken up by Bar-Hillel in the Fifties (Bar-Hillel
1954). Jim Lambek’s early papers (Lambek 1958; Lambek 1961), virtually unnoticed at the
time, have proved to be of central importance for the development of the field. In these papers,
the type-forming operations are for the first time treated as logical connectives; logical proof theory
takes the place of the stipulated rule schemata of the earlier systems. The seminal 1958 paper is
available electronically through JSTOR, and reprinted in (Buszkowski et al. 1998), a collection
which contains more of the early papers.

In the Eighties, the shift towards ‘lexicized’ grammar formalisms brings a revival of interest in
categorial grammar, which is recognized as the lexicialized framework par excellence. The
proceedings of the 1985 Tucson conference (Oehrle et al. 1988) give a good picture of the types
of categorial research in this period, both within the rule-based and within the logical traditions.
Van Beynnon’s contribution to this volume has been instrumental in introducing Lambek’s logical
approach to the linguistic community.

The advent of linear logic (Girard 1987), and the wave of research on ‘substructural’ styles of
inference with controlled options for resource management rather than hard-wired global choices,
have been important factors for the recent development of categorial grammar.
Action (van Benthem 1995) is a detailed study of the relations between categorial derivations, type theory and lambda calculus, and of the place of categorial grammars within the general landscape of resource-sensitive logics. Substructural Logics (Restall 2000) is an accessible textbook on this subject, doing justice both to Linear Logic and to its many predecessors in modal logic. The connections between linear logic, categorial grammar, and computational formulations of minimalist grammars are explored in a special issue of Language and Computation (Retoré and Stabler 2002). Proofs and types (Girard, Lafont, and Taylor 1988) is a good source for the Curry-Howard interpretation.

Apart from the chapter on categorial type logics (Moortgat 1997), which is the primary source for this article, the Handbook of Logic and Language (van Benthem and ter Meulen 1997) contains a number of further in-depth chapters that can be consulted for the connections between categorial type systems and mathematical linguistics and proof theory, formal learning theory, type theory, and Montague Grammar.

There is a choice of monographs and collections illustrating the different styles of current categorial research. Steedman’s recent books Surface Structure and Interpretation and The Syntactic Process (Steedman 1996; Steedman 2000b) well represent the agenda of Combinatory Categorial Grammar. For the deductive approach, the reader can turn to Type Logical Grammar (Morrill 1994), which offers a rich fragment of syntactic and semantic phenomena in the grammar of English, using a variety of type-forming operations (Boolean, quantificational) in addition to the composition operators discussed here. Type Logical Semantics (Carpenter 1998) is a general introduction to natural language semantics studied from the type-logical perspective; this book includes a detailed discussion of quantifier scope ambiguities as a case study. The collection (Kruifj and Oehrle 2002) reflects current categorial views on anaphora and binding.

A versatile computational tool for categorial exploration is Richard Moot’s grammar development environment GRAIL. The kernel of this system is a general type-theorem prover based on proof nets and structural graph rewriting. The user interacts with the kernel via a graphical user interface, which provides control over the lexicon and the structural module, and which gives access to a full-fledged proofnet based debugger. The system is publicly available at http://www.let.uu.nl/~Richard.Moot/personal/grail.html. A number of sample fragments can be accessed online at http://www.grail.let.uu.nl/tour.pdf.

**Text References**


Supplementary References


Figure 7: A proof net for the sentence ‘everyone loves somebody’. Formula decomposition trees with polarized vertices (black: input; white: output). Solid (dotted) edges for input (output) slashes. A linking of leaves with opposite polarities is well-formed if it produces a graph which is connected, acyclic (for each removal of a dotted edge from a pair), and planar. The net is constructed in a left-to-right incremental fashion. Processing complexity is measured in terms of the number of unresolved dependencies. The subject wide-scope reading for ‘everyone loves somebody’ (maximum of unresolved dependencies: 3) is preferred over the object wide-scope reading (maximum of unresolved dependencies: 4). Sources: Johnson (1998), Morrill (2000).
Semantics plays a role in grammar in at least three guises. (A) Linguists seek to account for speakers’ knowledge of what linguistic expressions mean. This goal is typically achieved by assigning a model theoretic interpretation in a compositional fashion. For example, \textit{No whale flies} is true if and only if the intersection of the sets of whales and fliers is empty in the model. (B) Linguists seek to account for the ability of speakers to make various inferences based on semantic knowledge. For example, \textit{No whale flies} entails \textit{No blue whale flies} and \textit{No whale flies high}. (C) The well-formedness of a variety of syntactic constructions depends on morpho-syntactic features with a semantic flavor. For example, \textit{Under no circumstances would a whale fly} is grammatical, whereas \textit{Under some circumstances would a whale fly} is not, corresponding to the downward vs. upward monotonic features of the preposed phrases.

It is usually assumed that once a compositional model theoretic interpretation is assigned to all expressions, its fruits can be freely enjoyed by inferencing and syntax. What place might proof theory have in this picture? This paper attempts to raise questions rather than offer a thesis.

1. Model theory and proof theory

Two approaches to semantics are the model theoretic and the proof theoretic ones. Using a familiar example, consider the model theoretic and the proof theoretic

---

1 An earlier version was presented at the 2005 Semantics Workshop at Rutgers University. I thank the commentators, Matthew Stone and Ken Shan, as well as the participants for criticism. Mark Steedman, Johan van Benthem, Chris Barker, Ed Stabler, Zoltan Szabo, Jason Stanley, and Barry Schein have kindly discussed these matters with me; this text does not yet do justice to all their suggestions.

2 There may be a discrepancy in the use of the term “semantics” between formal semanticists and philosophers. The former do not concern themselves with questions of reference and, in general, the relationship between expressions and the world out there; the entities in the models are abstract and linguistically/mathematically motivated.
faces of propositional logic. The interpretation of the connectives $\land, \lor$, and $\neg$ in terms of truth tables is the simplest kind of model theoretic semantics. It also determines relations between formulae. For example, $\neg p$ is a logical consequence of $\neg(p \lor q)$ because all the ways of assigning values to $p$ and $q$ that make $\neg(p \lor q)$ true also make $\neg p$ true. Thus the inference is said to be semantically valid (notated as $|=\$). Compare this with how the propositional calculus approaches the same inference. It offers a set of transition steps, the de Morgan law $\neg(p \lor q) = \neg p \land \neg q$ among them, with which the string of symbols $\neg p$ can be derived from the string $\neg(p \lor q)$. This demonstrates the so-called syntactic validity of the inference (notated as $|-\$). A calculus is sound if whatever it derives is true in the intended models ($\phi |- \psi$ only if $\phi |= \psi$); complete if it can derive whatever is true in the models ($\phi |- \psi$ whenever $\phi |= \psi$); and decidable if an algorithm can effectively determine whether $\phi |- \psi$ holds. The propositional calculus is sound, complete, and decidable; the first order predicate calculus is sound and complete. More complex calculi have at most generalized completeness.

As the term ‘syntactic validity’ indicates, proof theory involves symbol manipulation. Nevertheless, given soundness and (some interesting degree of) completeness, a calculus deserves the name ‘proof theoretic semantics’ in that it cashes out model theoretic semantic relations in its own syntactic terms, rather than concerning itself with the plain well-formedness of expressions, e.g., whether $(p \land)$ is well-formed.

On this view, model theory has primacy over proof theory. A language may be defined or described perfectly well without providing a calculus and thus, a logic for it, but (on this view) a calculus is of distinctly limited interest without a class of models with respect to which it is known to be sound and (to some interesting degree) complete.

It seems fair to say that mainstream formal semantics as practiced by linguists is exclusively model theoretic. As I understand it, the main goal is to elucidate the meanings of expressions in a compositional fashion, and to do that in a way that offers an insight into natural language metaphysics (Bach 1989) and uncovers universals of the syntax/semantics interface. Non-linguists sometimes regard the compositional interpretation of natural language expressions either as impossible or just a simple exercise. In contrast, linguists have come to think of it as a huge but rewarding empirical enterprise. The fact that the insights accumulated over the past

---

3 I am not quite sure of the position of the formal pragmatics work on implicatures, inspired by Grice. As far as I can see, the techniques are clearly model theoretic but the goals may or may not be.

4 Conservativity, a property of determiners or, more generally, of expressions denoting relations between sets, may be the best studied universal. Det is conservative iff Det(A)(B) = Det(A)(A $\cap$ B).
decades have been obtained by investigating denotation conditions. The immense role in linguists’ acceptance of and adherence to the model theoretic approach.

The view that model theory is not only necessary but also the primary source of insights is not the linguist’s invention. For example, in the Gamut textbook’s chapter on “Arguments and Inferences”, the authors--here, probably, Johan van Benthem and/or Dick de Jongh—put forward that “[S]emantic methods tend to give one a better understanding” but, they go on, “[they are] based on universal quantification over that mysterious totality, the class of all models (there are infinitely many models, and models themselves can be infinitely large). The notion of meaning that we use in the syntactic approach is more instrumental: the meaning of some part of the sentence lies in the conclusions which, because precisely that part appears at precisely that place, can be drawn from the sentence […]”.

This observation is nothing new, of course, but the idea that our way of doing semantics is both insightful and computationally (psychologically) unrealistic has failed to intrigue formal semanticists into action. Why? There are various, to my mind respectable, possibilities. (i) Given that the field is young and still in the process of identifying the main facts it should account for, we are going for the insight as opposed to the potential of computational/psychological reality. (ii) We don’t care about psychological reality and only study language in the abstract. (iii) We do care about potential psychological reality but are content to separate the elucidation of meaning (model theory) from the account of inferencing (proof theory). – But if the machineries of model theory and proof theory are sufficiently different, option (iii) may end up with a picture where speakers cannot know what sentences mean, so to speak, only how to draw inferences from them. Is that the correct picture? Perhaps we could have our cake and eat it too. Or have an altogether better cake if we cared to modify the recipe. It seems that a better understanding of the choice we made and of other choices that we might make would be useful. The present paper wishes to highlight this need and to elicit comments from linguists and from the neighboring fields.

In my initial attempt to reduce my ignorance, I seem to have identified three

---

5 I am carefully avoiding the term ‘truth (conditions)’, because only a fraction of natural language sentences and subsentential expressions can be said to be true or false to begin with, and also because it is immaterial from this perspective whether sentences are associated with truth values, situations, events, or something else.

6 Clearly, we are talking about proof theory offering the abstract possibility of psychological reality; there is no claim to the effect that people have, say, Natural Deduction machines in their heads.
interestingly different proof theoretic perspectives on semantics.\footnote{Proof theoretic approaches to natural language using categorial type logics have a rich tradition (Moortgat 1997, 2002; Oehrle 2003). But as far as I can see, the main focus has been on the syntax of natural languages. The central interpretive concern is limited to the Curry-Howard correspondence between formulae and lambda terms, i.e. to the interpretive effects of syntactic assembly. Aspects of meaning that go beyond type specification are not studied with any systematicity. Thus the actual results so far bear only on some of the issues formal semanticists tend to be interested in. Bernardi 2002, Bernardi and Szabolcsi 2005 are among the exceptions.}

2. Severing denotation conditions from infinity

Infinity is one typical complaint against model theoretic semantics. Indeed, finite systems may not literally house and manipulate an infinite set of infinite models, but this need not be a knockdown argument against the model theoretic approach. Infinity is necessary to capture the uncertainty as to what model, and what part of that model, we are talking about, but capturing this uncertainty need not belong to the object language. At the object language level infinity might be traded for partiality, retaining the strategy of assigning denotation conditions to expressions. Barwise and Perry 1983, Muskens 1995, and Kamp and Reyle 1993, 1996 come to mind. Denotation conditions, not infinity, are at the heart of the linguist’s attraction to model theory; the two could be disentangled.\footnote{Chomsky has often declared that semantics has no place within grammar. One might read this as a rejection of infinite models and as a commitment to proof theory, but most likely what Chomsky strives to reject is incorporating (links to) the real world, i.e. semantics in the philosopher’s sense. If so, then what formal semanticists do with abstract models may be fine with him and he would regard it as part of syntax. So, the mentalistic view of language may entail an answer to the model theory vs. proof theory dilemma, but it is not my impression that Chomsky’s actual statements are about this issue.}

A very interesting proposal of this sort is van Lambalgen and Hamm 2005. Here an event calculus is combined with minimal models in which events that the scenario of the given activity does not require to occur are assumed not to occur and enlargement of the model leads to nonmonotonic progression. What makes the proposal especially interesting is the fact that it puts to linguistic use a program for semantics where the sense of an expression is the algorithm that allows one to compute the denotation of the expression (Moschovakis 1994, 2006). Van Lambalgen and Hamm submit that only a computational notion of meaning is compatible with the results of psycholinguistics, but (drawing from Kamp’s and Steedman’s work on tense) the representations their theory computes are not alien to the denotational semantic intuition linguists have found insightful to work with.

In this connection we may also mention that the extensive literature on the
psychology of reasoning features two main approaches: mental models and mental logic. This may sound like a distinction corresponding to model theoretic versus proof theoretic semantics, but Johnson-Laird’s mental models are equally about building representations. Bonatti 1994 offers a good comparative evaluation.

3. Meaning via proofs

A second approach rejects the notion that inferences should play second fiddle to denotations. As Kahle and Schröder-Heister (2006) put it, “Proof-theoretic semantics proceeds the other way round, assigning proofs or deductions an autonomous semantic role from the very onset, rather than explaining this role in terms of truth transmission. In proof-theoretic semantics, proofs are not merely treated as syntactic objects as in Hilbert’s formalist philosophy of mathematics, but as entities in terms of which meaning and logical consequence can be explained.” See Prawitz (2006) on Gentzen, Dummett, and his own views. In a similar spirit, Moss (2005) wonders, “If one is seriously interested in entailment, why not study it axiomatically instead of building models? In particular, if one has a complete proof system, why not declare it to be the semantics? Indeed, why should semantics be founded on model theory rather than proof theory?”

Given the absence of pertinent literature, I am not in a position to judge how a semantics founded on proof theory would fare for natural language. In addition to a possibly major conceptual shift, I suspect that it may involve shifts in the detailed intuitions captured. To use a simple example, consider the model theoretic and the natural deduction treatments of the propositional connectives. The two ways of explicating conjunction and disjunction amount to the same thing indeed: if you know the one you can immediately guess the other. Not so with classical negation. The model theoretic definition is in one step: \( \neg p \) is true if and only if \( p \) is not true. In contrast, natural deduction obtains the same result in three steps. First, elimination and introduction rules for \( \neg \) yield a notion of negation as in minimal logic. Then the rule Ex Falso Sequitur Quodlibet is added to obtain intuitionistic negation, and finally Double Negation Cancellation to obtain classical negation. While it may be a matter of debate which explication is more insightful, it seems clear that the two are intuitively not the same, even though eventually they deliver the same result. See Hintikka 2002 for the possible linguistic relevance of this.

4. Natural Logic and semantically flavored syntactic features

A third relevant approach, Natural Logic, bears out the slogan that proof theory “syntacticianizes semantics”, not only in the sense that it manipulates representations,
but also in the sense that it lives off of the actual syntactic representations of expressions. It uses linguistic structures, as opposed to models or an auxiliary logical language, as the vehicle of inference. The literature contains a collection of small subsystems that are individually sound and complete in terms of the standard models. The techniques are fairly diverse. Johan van Benthem’s Monotonicity Calculus, explored further by Victor Sánchez-Valencia, tags all items for monotonicity and for polarity position, and computes the increasing/decreasing inferential status of any expression in tandem with the categorial grammatical derivation. Larry Moss presents a syllogistic logic with quantifiers, notably including most, which is not first order definable. Yoad Winter handles inferences with restrictive modification, monotonicity, and quantifier scope, exploiting insights from generalized quantifier theory.

If proof theory syntacticizes semantics, it may be of particular interest to pay attention to semantic properties that natural language already singles out as syntactically relevant. I will dub these semantically flavored syntactic features. Some fairly standard examples are [wh] (i.e. interrogative), [topic], [focus], [negative], [agent], [number], [telic], [evidential], and so on. It turns out that such features are quite pervasive, and generative syntax uses them as conditions for syntactic operations (“merge” and “move”). The so-called negative inversion construction of English is an example.

(1) Under no/few circumstances would a whale fly.
(2) *Under some/most circumstances would a whale fly.

(1) is acceptable, (2) is not, although it does not seem incoherent. The generalization is that the initial position accepts an adjunct only if it is (roughly) decreasing. One way to implement this is to assume that whenever the decision to fill this position arises in a syntactic derivation, the compositional model theoretic interpretation of the adjunct is inspected for decreasingness. This is what semanticists would do by default. Another implementation is to assume that certain adjuncts have a purely syntactic feature [de]; the set of expressions with [de] may substantially overlap or even coincide with those whose denotations are decreasing, but this fact has no place in the theory. This is what syntacticians would do by default. In contrast, Stabler (1997) proposes that [de] is a properly syntactic feature but, in addition to

---

9 That is, strings together with their syntactic and possibly intonational analyses. This is a straightforward response to the “misleading form” objection to Natural Logic.

10 This seems to be the same property as the one involved in the licensing of negative polarity items like ever. Given A≤B, a function f is monotonically increasing (upward entailing) iff f(A)≤f(B), and decreasing (downward entailing) iff f(A)≥f(B). See Ladusaw 1980, von Fintel 1999.
licensing syntactic operations like negative inversion, it features in the proof theoretic component and speakers use it to draw inferences. Natural language syntax can be sensitive to semantic properties precisely because its semantics is proof theoretical in nature, and those particular properties play a role in this proof theory. If this view is correct, we may say that syntax is a “window on the format of semantics”. (This formulation is somewhat stronger than Stabler’s but in keeping with Stabler’s views and intentions.)

A more conservative view maybe a hybrid one. Here natural language would have, in addition to a full model theoretic semantics, a partial proof theoretic one, which provides shortcuts in the cases of some shallow semantic features. For example, Geurts and van der Slik (2005) observe that monotonicity properties are shallow. Even though speakers often disagree about the precise truth conditions of donkey-sentences (is (3) true or false if farmers have more than one donkey each and do not treat all their donkeys alike?), they are quick to recognize that (3) entails (4) and is entailed by (5):

(3) Every farmer who owns a donkey beats it.
(4) Every farmer who owns a male donkey beats it.
(5) Every farmer who owns a donkey beats it with a stick.

This indicates that monotonicity inferences do not mobilize the whole model theoretic semantics of the sentences. We may now hypothesize that semantically flavored syntactic features are shallow ones. This would preserve Stabler’s insight without committing us to handle all complexities of linguistic meaning in a proof theoretic fashion.

5. Some general questions

So, some general questions arise, for the global approaches as well as for the particular variants.

Proponents of proof theoretic methods seem confident that only their approach, not the model theoretic one, can be integrated with the rest of cognitive science. Is that correct? If yes, what is the crux of the matter -- finite representations or inferential character? Is cognitive science possible without relating to the world outside?

Do model theoretic and proof theoretic semantics differ as to what general conception of language they fit with? Would there be gains or losses in domains not considered above?

11 Referring to mathematical results that the set of first order quantifiers is not identifiable in the limit from examples, Stabler 2005 points out that they might be learnable given inferential evidence.
What are the prospects of extending the proof theoretic approach to intensional phenomena, presupposition, and implicatures?

What kind of compositionality would proof theoretic approaches afford?

Although it has sometimes been suggested that any effective procedure that computes meanings will do, I believe that there is an important consideration that suggests that we must be more particular. Whatever one might think of the specific theories generative grammar has come up with over the past decades, I believe it has been demonstrated beyond any reasonable doubt that natural languages, while superficially wildly different, exhibit very detailed and thoroughgoing structural similarities; in other words, that “universal grammar” is not merely a wishful thought.

Therefore no theory incapable of accounting for the unity behind the superficial variation stands a chance to be an even remotely valid theory of natural language.

Now, cross-linguistic variation in syntax is to some extent paralleled by cross-linguistic variation in interpretation. Here are two simple examples.

(i) Given the right predicate, bare plurals in English and German may have an existential or a generic reading:

(6) Professors are sick.
   `There are professors stricken with illness'
   `Professors in general are disgusting'

But it is well-known that in many other languages, Romance languages among them, one or both interpretations may be unavailable.

(ii) The interaction of negation with disjunction and conjunction in English and German straightforwardly bears out the de Morgan laws:

(7) John didn’t study flute or accounting.
   primary reading: `neither'
(8) John didn’t study flute and accounting.
   primary reading: `not both'

In many other languages, Russian, Italian, Japanese, and Hungarian among them, the above interpretations are missing. The literal counterparts of (7) mean exclusively `One or the other he didn’t study’ and the literal counterparts of (8) mean exclusively `He studied neither one’.

Given such variation, it does not suffice to provide some effective procedure that delivers the correct interpretations for the constructions of the individual languages; what is needed is a compositional analysis that accounts for exactly how
languages differ. Without that, human languages will appear to be incommensurable.

References


Bernardi, Raffaella and Anna Szabolcsi (2007), Optionality, scope, and licensing. ESSLLI 2007 CD.


12 Chierchia 1998 and Szabolcs and Haddican 2004 are such attempts.


Stabler, Edward (2005), Natural Logic in linguistic theory. Workshop on *Proof Theory at the Syntax/Semantics Interface*. http://wintermute.linguistics.ucla.edu/prooftheory/


