

A Natural Logic Inference System

Yaroslav Fyodorov Yoad Winter Nissim Francez
Technion – Israel Institute of Technology
yaroslav, winter, francez@cs.technion.ac.il

Abstract This paper develops a version of Natural Logic – an inference system that works directly on natural language syntactic representations, with no intermediate translation to logical formulae. Following work by Sánchez (1991), we develop a small fragment that computes semantic order relations between derivation trees in Categorical Grammar. Unlike previous works, the proposed system has the following new characteristics: (i) It uses orderings between derivation trees as purely syntactic units, derivable by a formal calculus. (ii) The system is extended for conjunctive phenomena like coordination and relative clauses. This allows a simple account of non-monotonic expressions that are reducible to conjunctions of monotonic ones. (iii) A preliminary proof search algorithm based on a tree generating regular system is developed for Sánchez’ smaller fragment of Natural Logic.

1 Introduction

Model-theoretic semantic theories of natural language assume that most linguistic expressions – or even all of them – represent objects in partially ordered domains so that meanings of expressions of the same category are naturally comparable. Formal semantics treats order relations between expressions of complex categories as compositionally derived from orders between expressions of simpler categories, according to the structural rules of a given grammar and certain semantic properties of words. For instance, the denotation of the nominal expression *tall student* is semantically ”smaller than” the denotation of the noun *student* in every model. This simple ordering, together with the ”order reversing” meaning of the determiner *no*, is responsible for the fact that the noun phrase *no tall student* is semantically ”greater than” the noun phrase *no student* in every model. At the top level, these order statements result in a semantic ordering of natural language sentences. In an adequate semantic theory, this ordering corresponds to an intuitively valid *entailment* relation between sentences. For instance, the aforementioned order statements, together with the other elements in the sentence, are responsible for the valid conclusion of the sentence *John saw no tall student* from the premise *John saw no student*.

Fruitful as this view on inference in natural language is, its formulation within model-theoretic semantics is not immediately helpful for the design of computationally significant inference systems. The project on which this paper reports aims to develop an inference system for natural language using basic insights on order relations coming directly from model-theoretic semantics, but using only proof-theoretical symbolic manipulation of natural language syntactic representations with no appeal to models. On the other hand, the direct relationships between structures and meanings in model-theoretic semantics allows us to dispense with the translation of natural language syntactic representations into a level of logical representation. Rather, inferences are computed directly from semantically annotated syntactic derivations of expressions.

In Sánchez (1991), a similar conception of *Natural Logic* is used to describe certain semantic properties of natural language expressions. Sánchez proposes a mechanism

that decorates categorial grammar proofs of natural language expressions using signs that indicate the *monotonicity* properties of these expressions – whether they are “order preserving” or “order reversing”. Sánchez shows that this monotonicity marking can be used to account for non-trivial inferences in natural language. However, while Sánchez’ annotation of proof trees is rigorously defined, the derivation of the order statements that use them is not fully formalized. A similar limitation exists in more recent extensions of Sánchez’ work, as in Bernardi (1999). Therefore, Sánchez’ system and its current decedents do not fully derive inferences between natural language sentences. The first step we take in this paper is to use Sánchez’ treatment of monotonicity for defining order statements as purely syntactic relations between derivation trees in categorial grammar, using a formal calculus we call the *order calculus*. As a logic this “natural logic” is different than traditional conceptions of logic as a closure of a set of atomic formulas under certain operators. Here each formula is a pair of derivation trees in Categorial Grammar and therefore the whole power of CG is needed to describe formulas. Another non-obvious question about Sánchez’ system is whether this treatment of monotonicity can extend to cover inferences with non-monotonic expressions. We show that at least for a (large) subset of non-monotonic items, such a treatment is possible using a novel treatment of *coordination* in natural logic that relies on the semantic fact that items like *and* and *or* are greatest lower bound/least upper bound operators respectively with respect to the order relations in the categories they apply to. This immediately captures entailments with so called *continuous* non-monotonic expressions, which as proven in Thijsse (1983) are equivalent to conjunctions of monotonic expressions. Another question about natural logic is whether variants of this system are decidable. We propose a positive answer to this question by developing a proof search algorithm for our formulation of Sánchez’ system. The algorithm is based on a simple procedure for identifying trees generated by *regular systems* (Brainerd (1969)).

Section 2 briefly discusses previous works on inference in natural language and their relations with the present enterprise. Section 3 develops a categorial inference system formalizing Sánchez’ work and extends it to treat coordination. Section 4 gives a small lexicon for the system that is used for illustrating its application for some inferences with natural language sentences. Section 5 develops the proposed preliminary proof search algorithm and sketches briefly its correctness proof. A first version of a working prototype for computing inferences in natural logic has been implemented using Bob Carpenter’s TLG categorial grammar parser, and a fuller implementation is currently under development.

2 Previous work

Purdy (1991) proposes a sound and complete inference system that is based on operations with n -ary relations. The expressive power of this system lies between the predicate calculus without identity and the predicate calculus with identity. Purdy then shows how sentences in a simple fragment of English can be easily translated into this formalism. It is not clear to us, however, to what extent the structure of Purdy’s fragment resembles the structure of English. Most notably, the English noun phrase is missing from this fragment. This is because the analysis of structures of the form

Determiner - Noun - N-ary Predicate requires a representation for each of the three elements in Purdy's system. If the determiner is to combine with the noun first, as in most English grammars, then Purdy's system would require translation procedures that are not simpler than those needed into the notation of the standard predicate calculus. Consequently, it is hard to see how Purdy's system can deal with common phenomena like NP coordination in an elegant way.

McAllester and Givan (1992) propose an inference system that is sound and complete, and moreover decidable in polynomial time. The system is based on so-called *class expressions* – expressions that denote sets. A determiner like *every* can combine with two class expressions to form an atomic formula. Alternatively, a determiner, a class expression and a binary relation can form together another class expression. These operations are quite parallel to natural language structures and furthermore, the time complexity of the system is rather low. However, the proposed logical system is pretty weak. For instance, since negation operates only on atomic formulae, a determiner like *no* can be handled only when it appears in subject position (e.g. *no student smiled*) but not in object position (e.g. *John likes no student*). In addition, the inference rules in the system are specialized for deduction in the proposed fragment of predicate calculus, and it seems hard to extend them for richer constructions.

We think that because of these limitations of inference systems for natural language, it may be worthwhile to study systems like Sanchez', which are in better correspondence to natural language syntax, and whose inference rules are more directly related to modeltheoretic semantics of natural language. Such systems may be applicable for a larger variety of fragments. The price to be paid for the larger syntactic flexibility is that completeness results seem harder to obtain. In the next sections we introduce a version of natural logic and study its possible implementations.

3 The Categorical Inference System

In this section we introduce the two calculi that will be used to compute inferences with (non-)monotonic expressions. A categorical calculus, a variant of the simple AB-calculus, is responsible for assigning derivation trees decorated with monotonicity marking to expressions of natural language. An *order calculus* is responsible for deriving boolean order statements on such derivation trees. Especially, this calculus derives order relations between sentences, which correspond to semantic entailment relations.¹

We first define the *decorated categories* of the grammar. These categories are based on a small set of *primitive categories* and its standard extension to the set of categories in applicative categorial grammar, with the additional feature that decorated categories can have signs marking certain semantic properties (like monotonic, restrictive, or conjunctive behavior of expressions that derive them). A standard correspondence between decorated categories and functional semantic types is defined. This is needed not only for the standard definition of a type-theoretical semantics, but furthermore for the specification of which expressions (possibly of different syntactic

¹Here and henceforth, when we talk about entailment relations, we also consider cases where there are two or more premises.

categories) are of the same semantic type and hence should be comparable in the order calculus. The subset of categories that is of special interest in our inference system is the set of *boolean categories*, those categories that correspond to boolean types. For this subset order relations are easiest to define. We follow Keenan and Faltz (1985) and eliminate non-boolean categories from the set of decorated categories. The definitions below introduce types and categories in the system. Here an henceforth, by *types* we refer to semantic types and by *categories* to syntactic categories.

Definition 1 (types) Let \mathbf{T}^0 be some finite set of primitive types that includes t (for ‘truth value’). The set of types is the smallest set \mathbf{T} that satisfies:

1. $\mathbf{T}^0 \subseteq \mathbf{T}$,
2. If $\tau \in \mathbf{T}$ and $\sigma \in \mathbf{T}$ then also $(\tau\sigma) \in \mathbf{T}$.

Definition 2 (boolean types) Let $\mathbf{T}_b^0 \subseteq \mathbf{T}^0$ be some finite set of primitive boolean types s.t. $t \in \mathbf{T}_b^0$. The set of boolean types is the smallest set $\mathbf{T}_b \subseteq \mathbf{T}$ that satisfies:

1. $\mathbf{T}_b^0 \subseteq \mathbf{T}_b$,
2. If $\tau \in \mathbf{T}$ and $\sigma \in \mathbf{T}_b$ then also $(\tau\sigma) \in \mathbf{T}_b$.

The set of (undecorated) categories is standardly defined as in applicative categorial grammar (in “result on left” style slash format), with a mapping **type** from categories to types. We assume that the set of primitive categories includes at least two designated categories, s (for sentences) and \bar{s} (a category whose use will become clear as we go along).

Definition 3 (categories) Let \mathbf{CAT}^0 be a finite set of primitive categories s.t. $\{s, \bar{s}\} \subseteq \mathbf{CAT}^0$. Let $\mathbf{type}^0 : \mathbf{CAT}^0 \rightarrow \mathbf{T}$ be a typing function for this set s.t. $\mathbf{type}^0(s) = \mathbf{type}^0(\bar{s}) = t$. The set of categories is defined, together with its typing function **type** extending \mathbf{type}^0 , as the smallest set \mathbf{CAT} that satisfies:

1. $\mathbf{CAT}^0 \subseteq \mathbf{CAT}$. For any $A \in \mathbf{CAT}^0 : \mathbf{type}^0(A) = \mathbf{type}(A)$.
2. If $A \in \mathbf{CAT}$ and $B \in \mathbf{CAT}$ then $A/B \in \mathbf{CAT}$ and $A \setminus B \in \mathbf{CAT}$, and $\mathbf{type}(A/B) = \mathbf{type}(A \setminus B) = (\mathbf{type}(B) \mathbf{type}(A))$.

Definition 4 (boolean categories) The set \mathbf{CAT}_b of boolean categories is defined by $\{A \in \mathbf{CAT} : \mathbf{type}(A) \in \mathbf{T}_b\}$ - the set of categories whose type is boolean.

We use annotations on elements from the set of boolean categories to encode three kinds of semantic properties: *monotonicity*, *restrictive modification* and *conjunction/disjunction*. The annotation uses a semantic feature $F \in \{+, -, R, C, D\}$ on the main (back)slash constructor of the category, for denoting upward (+) monotonicity, downward (−) monotonicity, Restrictive modification, Conjunction and Disjunction, respectively.² The annotation F on a category A/FB that is assigned to an expression α means that α is upward/downward monotone, restrictive etc.

²In fact, these properties are not mutually exclusive, nor do they completely follow from one another. For instance, a restrictive modifier may or may not be upward monotone, but a conjunction is always upward monotone on both its arguments (see fact 1 below). A more comprehensive treatment would allow decorating a category by a *set* of semantic features, or even a *feature structure* of such features.

Monotonicity: A category A/B or $A \setminus B$ where both A and B are boolean can be decorated by a monotonicity $+/-$ sign, for upward/downward monotonicity. For instance, assigning the decorated category $(s/^+(s \setminus np))/-n$ to the determiner *every* specifies the (correct) assumption that this determiner is downward monotone on its nominal argument and, furthermore, that the noun phrase it generates (of the lifted category $s/^+(s \setminus np)$) is upward monotone on its verb phrase argument.

Restrictive Modification: A boolean category that corresponds to a type $\tau\tau$ can be restrictive in imposing an ordering between an expression it combines with and the result of the combination. For instance, we say that *yesterday* is a restrictive (adverbial) modifier because anyone who ran yesterday necessarily ran. Likewise, *tall* is a restrictive adjective because anyone who is a tall student, for instance, is by necessity a student. The annotation for restrictiveness is 'R'.

Conjunction/Disjunction: A boolean category that corresponds to a type $\tau(\tau\tau)$ is in fact a "coordinator", which can be marked by a C/D sign, for conjunction/disjunction. Three different kinds of conjunctions are *conjunctive coordinations* (e.g. *walked and talked*), *relative clauses* (e.g. *a man who walked*) and *intersective modification*, where conjunction is implicit (e.g. *a blue car is a car which is blue*).

The set \mathbf{CAT}_d of *decorated categories* contains the boolean categories, possibly marked for monotonicity, conjunction/disjunction or restrictiveness.

Definition 5 (decorated categories) *The set of decorated categories is defined, together with its typing function \mathbf{type}^d extending \mathbf{type} , as the smallest set \mathbf{CAT}_d that satisfies:*

1. $\mathbf{CAT}_b \subseteq \mathbf{CAT}_d$. For any $A \in \mathbf{CAT}_b$: $\mathbf{type}^d(A) = \mathbf{type}(A)$.
2. If $A \in \mathbf{CAT}_d$ and $B \in \mathbf{CAT}_d$ then $A/^F B \in \mathbf{CAT}_d$ and $A \setminus^F B \in \mathbf{CAT}_d$, where F is in the set $\{+, -, R, C, D, \epsilon\}$ (ϵ is the empty marking) and the following conditions hold:
 - (a) $\mathbf{type}^d(A/^F B) = \mathbf{type}^d(A \setminus^F B) = (\mathbf{type}^d(A) \mathbf{type}^d(B))$,
 - (b) If $F = 'R'$ then $\mathbf{type}^d(A) = \mathbf{type}^d(B)$,
 - (c) If $F = 'C'$ or $F = 'D'$ and $\mathbf{type}^d(B) = \tau$ then $\mathbf{type}^d(A) = (\tau\tau)$.

Notation: We use $A/*B$ (or $A \setminus^* B$) as a pattern matching any of the categories $A/^F B$ (or $A \setminus^F B$), where $F \in \{+, -, R, C, D, \epsilon\}$.

We define two notions of equivalence between decorated categories. Equivalence between categories specifies when a pair of derivation trees of two categories is a legitimate order statement in the order calculus. *Formal* equivalence between two decorated categories means that they may only be different in their semantic features. Two decorated categories are *semantically equivalent* if their semantic type is the same. The latter relation contains of course the former relation, and hence allows more expressive order calculi. We define:

Definition 6 (formally equivalent categories) *For any two decorated categories $A, B \in \mathbf{CAT}^d$ we say that A is formally equivalent to B , and denote $A \equiv_f B$, iff:*

1. A and B are primitive and $A = B$ or $\{A, B\} = \{s, \bar{s}\}$, or
2. $A = A_1/*A_2$, $B = B_1/*B_2$, $A_1 \equiv_f A_2$ and $B_1 \equiv_f B_2$, or

3. $A = A_1 \setminus^* A_2$, $B = B_1 \setminus^* B_2$, $A_1 \equiv_f A_2$ and $B_1 \equiv_f B_2$.

Definition 7 (semantically equivalent categories) For any two decorated categories $A, B \in \mathbf{CAT}^d$ we say that A is semantically equivalent to B , and denote $A \equiv_s B$, iff $\text{type}^d(A) = \text{type}^d(B)$.

To give an example, it is standard to assign the type e to the primitive category np (for noun phrases) and to give the type et to the primitive category n (for nouns). Consequently, the categories n and $\text{s} \setminus \text{np}$ (for transitive verbs) become semantically equivalent. A grammar can thus assign the category $(\text{n} \setminus \text{n}) / \text{c} (\text{s} \setminus \text{np})$ to a word like *who*, in its use as a subject-oriented relativizer (as in the nominal *child who walks*), encoding the (correct) assumption that the semantics of this construction is conjunctive (e.g. x is a child who walks iff x is a child and x walks).

We use a variant of the standard applicative *AB calculus* over the decorated categories, which is responsible for derivation of syntactic analyses for given expressions. The only difference from the standard AB calculus is that here the argument category B in a functor category A/B (or $A \setminus B$) may be decorated differently than the category B' that combines with the functor. Therefore we require formal equivalence between B and B' , and not simply identity.

Definition 8 (AB calculus) For any two decorated categories $A/*B$ (or $A \setminus^* B$) and $B' \equiv_f B$ we have the following (back)slash elimination rules.

$$\frac{A/*B \quad B'}{A} \text{ /E} \quad \frac{B' \quad A \setminus^* B}{A} \text{ \E}$$

Notation: We use the ' $|$ ' sign as a pattern matching both ' $/$ ' and ' \setminus ', in cases where the direction of the slash is immaterial. In such cases the patterns

$$\frac{A/*B \quad B'}{A} \text{ |E} \quad \text{and} \quad \frac{B' \quad A \setminus^* B}{A} \text{ |E}$$

each match both of the above elimination rules.

The AB calculus together with a *lexicon* constitute a *grammar*, used for assigning *derivation trees* to natural language expressions. The derivation trees are the syntactic objects for which the order calculus is defined below. The definition of the lexicon and derivation trees is straightforward. We add to any lexicon the word w_\top (the "top word") of category $\bar{\mathfrak{s}}$, whose use will become clear in the sequel.

Definition 9 (lexicon) A *lexicon* is a pair $\langle \Sigma, \text{cat} \rangle$, where Σ is a finite set of words s.t. $w_\top \notin \Sigma$, and cat is a function from $\Sigma \cup \{w_\top\}$ to non-empty finite subsets of \mathbf{CAT}_d s.t. $\text{cat}(w_\top) = \{\bar{\mathfrak{s}}\}$.

Definition 10 (derivation tree) A *labeled tree* T is a *derivation tree* for a string $\sigma \in \Sigma^+$ iff: (i) The frontier of T is labeled by the elements of σ , (ii) Every leaf x in T has no brother and $\text{label}(\text{mother}(x)) \in \text{cat}(\text{label}(x))$, and (iii) The internal nodes in T (those not in its frontier) form a *proof tree* in the AB calculus.

The main part of the system is the *order calculus* defined below, which is the engine that derives semantic order relations between derivation trees of natural language

Premise	Conclusion	
	conjunction	disjunction
$z = x$ or $z = y$	$\wedge(x, y) \leq z$	$z \leq \vee(x, y)$
$z \leq x$ and $z \leq y$	$z \leq \wedge(x, y)$	
$x \leq z$ and $y \leq z$		$\vee(x, y) \leq z$

Table 1: conjunction and disjunction rules

expressions. The *formulas* manipulated by this calculus are order statements between derivation trees of formally or semantically equivalent categories. Thus, the general form of formulas in this calculus is:

$\alpha_A \leq \beta_B$, where α_A and β_B are derivation trees of categories A and B respectively such that $A \equiv_{s/f} B$.

Notational conventions: ' $T_2 \geq T_1$ ' instead of ' $T_1 \leq T_2$ '; ' $T_2 = T_1$ ' instead of ' $T_1 \leq T_2$ and $T_2 \leq T_1$ '. Note that all the decorated categories in \mathbf{CAT}_d are based on boolean categories in \mathbf{CAT}_b . Therefore, it makes sense to compare any two derivation trees that have semantically equivalent categories at their roots, as the semantic domains of all decorated categories are guaranteed to be ordered.

Having specified the item forms in the system, we can now move to the definition of the inference rules in the order calculus. Inference rules with no premises are order *axioms*, and they impose order statements between natural language expressions of certain constructions. Other rules derive order statements from given order statements. Two simple rules in the system, reflecting basic properties of the ' \leq ' relation (independent of derivation trees that it relates) are *Reflexivity* and *Transitivity*. In addition, there are rules that treat specific structures generated by the AB calculus. We identify three typical structures: function application, modification and coordination.

Function application is general application of (back)slash elimination rules with no further restrictions. In this case we derive an ordering between $f(x)$ and $g(y)$ using two rules: (i) *Monotonicity* – where $f = g$ is a monotonic function and an ordering is given between x and y ; and (ii) *Function replacement* – where an ordering is given in both relation between x and y (i.e. they are semantically equivalent) and an ordering is given between f and g (which do not have to be monotonic).

Modification holds in the special case of function application where the domain and the range of f are the same. A *Restrictive Modification* axiom derives then an order between x and $f(x)$. *Coordination* is a situation where a "Curried" function F is of type $\tau(\tau\tau)$. That is, F has two arguments of the same type as the result. Coordination rules derive an order between $F(x, y)$ and z using orders between z and x and/or between z and y . These rules are summarized in table 1 for conjunction and disjunction and they reflect the fact that conjunction/disjunction are *greatest lower bound/least upper bound* operators with respect to the ' \leq ' relation. In the order calculus defined below, we spare the definition of the disjunction rules, because of their obvious symmetry with the conjunction rules. An additional rule, combining coordination and function application, reflects our assumption (following Partee and Rooth (1983)) that all coordinators in natural language are *pointwise operators*. That is, if F is a polymorphic coordinator then $(F_{(\tau\sigma)((\tau\sigma)(\tau\sigma))}(f_{\tau\sigma}, g_{\tau\sigma}))(x_\tau) = F_{\sigma(\sigma\sigma)}(f(x), g(x))$.

The *order calculus* embodies these assumptions within a grammar induced by the directed AB calculus. We give two versions of this calculus. Version 1 of this calculus, formalizing Sánchez' proposal, uses formal equivalence between categories. The second version uses semantic equivalence between categories and adds conjunction rules.

Definition 11 (order calculus 1 (OC1)) *In the following deduction rules, '≡' stands for '≡_f' (formal equivalence between categories).*

$$\text{Reflexivity: } \frac{\emptyset}{T \leq T} \text{ REFL} \quad \text{Transitivity: } \frac{T_1 \leq T_2 \quad T_2 \leq T_3}{T_1 \leq T_3} \text{ TRANS}$$

$$\text{Monotonicity: } \frac{\alpha_B \leq \beta_B}{\frac{\frac{\gamma_{A|\pm B} \quad \alpha_B}{A} |E}{\leq} \frac{\frac{\gamma_{A|\pm B} \quad \beta_B}{A} |E}}{\geq} \text{ MON}$$

'≤' and '≥' for '+' and '-' respectively.

$$\text{Function Replacement: } \frac{\alpha_{A|*B} \leq \beta_{A'|*B'} \quad \gamma_B \leq \delta_{B'} \quad \delta_{B'} \leq \gamma_B}{\frac{\frac{\alpha_{A|*B} \quad \gamma_B}{A} |E}{\leq} \frac{\frac{\beta_{A'|*B'} \quad \delta_{B'}}{A'}}{|E}} \text{ FR}$$

where $A \equiv A'$, $B \equiv B'$.

$$\text{Restrictive Modification: } \frac{\emptyset}{\frac{\frac{\alpha_{A'|*A} \quad \beta_A}{A'}}{|E}} \leq \beta_A \text{ RMOD}$$

where $A \equiv A'$.

Definition 12 (order calculus 2 (OC2)) *The second version of the order calculus, OC2, uses the rules of the above calculus OC1, with '≡' standing for '≡_s' (semantic equivalence between categories), and adds the following deduction rules,*

$$\text{Conjunction: } \frac{\emptyset}{\frac{\frac{\frac{\alpha_{(A|B)|C} \quad \gamma_C}{A|B} |E}{A} \quad \beta_B |E}{\leq} T} \text{ C1} \quad \frac{\alpha'_{A'} \leq \beta_B \quad \alpha'_{A'} \leq \gamma_C}{\alpha'_{A'} \leq \frac{\frac{\frac{\alpha_{(A|B)|C} \quad \gamma_C}{A|B} |E}{A} \quad \beta_B |E}} \text{ C2}$$

where $T = \beta_B$ or $T = \gamma_C$

Pointwise Coordination:

$$\frac{\emptyset}{\frac{\frac{\frac{\frac{\alpha_{((A|B)|(A|B)|(A|B)} \quad \beta_{A|B}}{(A|B)|(A|B)} |E}{A|B} \quad \gamma_{A|B} |E}{A} \quad \delta_B |E}{\leq} \frac{\frac{\frac{\beta_{A|B} \quad \delta_B}{A} |E}{A|A} \quad \gamma_{A|B} \quad \delta_B |E}{A} |E}} \text{ PWC}$$

We define *provability* in the order calculus C as the relation that exists between a finite set $P = \{\alpha_i \leq \beta_i : i \in \{1..n\}\}$ of premise order statements and any order statement $\alpha \leq \beta$ derived from P in C using application of inference rules.

Note the following simple fact about this system.

Fact 1 (i) Any conjunctive category $(A|B)^{|^c}C$ can be replaced by $(A|^R B)^{|^c}C$ with no loss of generality. (ii) Any conjunctive category $(A|B)^{|^c}C$ can be replaced by $(A|^+ B)^{|^+}C$ with no loss of generality

In other words, this fact states that it is provable in the system that (i) any conjunction combines with its first argument to yield a restrictive modifier and that (ii) that conjunction is upward monotone on both its arguments.

To illustrate the use of the above rules, we give below some examples for order statements between natural language expressions that can be obtained by direct application of these rules to trees derived by a simple lexicon for the AB calculus like the one defined in the next section. For simplicity of notation we omit the derivation trees of the expressions and just add parentheses where necessary.

- $tall(student) \leq student$ (restrictive adjective modification)³
 $yesterday(ran) \leq ran$ (restrictive adverb modification)
- $some(tall student) \leq some(student)$ (upward monotonicity of *some*)
 $every(student) \leq every(tall student)$ (downward monotonicity of *every*)
- $(every student)(ran) \leq (every tall student)(ran)$ (function replacement)
- $student\ who\ ran \leq student,$
 $student\ who\ ran \leq ran$ (conjunctive behavior of relatives)
- $every\ student\ and\ some\ teacher\ ran = every\ student\ ran\ and\ some\ teacher\ ran$ (pointwise coordination)

For proving the *soundness* of this system we use a standard extension of the AB calculus such that any derivable expression is assigned, besides a category A , also a lambda term ψ of type $\mathbf{type}^d(A)$. For lexical items this term is stipulated and for complex items it is derived inductively using the Curry-Howard isomorphism. The models for an order statement $T_1 \leq T_2$, where T_1 and T_2 are derivation trees deriving categories A_1, A_2 and terms ψ_1, ψ_2 respectively, are the standard models for Intensional Logic with the restriction that the denotation of a term assigned to any lexical item w of category B has in every model the semantic properties decorating B . For instance, the denotation of a word w of category $B_1|^+ B_2$ should be an upward monotone function in every model. The order statement $T_1 \leq T_2$ gets the value **true/false** in a model M according to whether the denotation in M of the term ψ_1 assigned to T_1 is ordered as “less or equal” to the denotation in M of the term ψ_2 assigned to T_2 . This semantic ordering is according to the natural semantic relation \sqsubseteq defined as follows for any lambda terms ψ_1, ψ_2 of the same boolean type τ :

Primitive τ : whether $\llbracket \psi_1 \rrbracket_M \sqsubseteq \llbracket \psi_2 \rrbracket_M$ holds is given for all possible values of τ .

$\tau = (\tau_1 \tau_2)$: $\llbracket \psi_1 \rrbracket_M \sqsubseteq \llbracket \psi_2 \rrbracket_M$ holds iff for every X in M of type τ_1 the ordering $\llbracket \psi_1 \rrbracket_M(X) \sqsubseteq \llbracket \psi_2 \rrbracket_M(X)$ holds.

Using these fairly standard notions, it is easy to show that the rules in the two order calculi defined above are sound. Whether these calculi are complete, or extendable to a complete proof system, is currently unknown to us.

³Non-restrictive adjectives like *fake* do not follow this pattern (e.g. a fake diamond is not a diamond), and we assume that this non-restrictiveness should be reflected in the category of the adjective, so that the modification rule does not apply to them.

4 The order calculus as an inference system for natural language

In this section we briefly illustrate how the order calculus OC2 as defined above can be used for deriving inferences in natural language. To do that we have to add three ingredients to the system described so far: a presentation of natural language sentences as order relations that can be manipulated by the order calculus, a concrete lexicon, and lexical/extra-logical order relations, which are not accounted for by the current version of the order calculus. We describe these three parts and then give some examples for derivations in the resulting system.

4.1 Assertions as order relations

The premises and the goal in the system assertions of natural language sentences. We regard such assertions as order statements of the form $\top \leq T$, where T is a derivation tree of a sentence and \top is notation for the following special derivation tree.

$$\top = \begin{array}{c} w_{\top} \\ | \\ \bar{s} \end{array}$$

Thus, we treat assertions as a special case of order statements, between a derivation tree of a sentence and a "top" element that corresponds to the truth value "true". This allows us to have a single uniform item form in the system, encoding both assertions of sentences and order relations between any linguistic expressions. Thus, in a linguistically motivated system that gets inputs in the form of natural language sentences, the assumptions and the goal are of the following form.

Assumptions/Goal: $\top \leq \alpha_s$, α_s is a derivation tree deriving s .

4.2 Lexicon and ad hoc syntactic rules

We use a lexicon that will allow us to demonstrate the main properties of the system in treating monotonic and non-monotonic expressions. The set \mathbf{T}^0 of primitive types we use includes the types t (for truth values), e (for entities) and v (for natural numbers). The boolean primitive categories in \mathbf{T}_b^0 are t and v . The set of primitive categories with their corresponding types (assigned by the type^0 function) is:

$$s:t \quad \bar{s}:t \quad np:e \quad n:(et) \quad num:v$$

The categories s and \bar{s} are for sentences, np is for noun phrases, n is for nouns and num is for numerals. A lexicon using these categories is given in table 2.

In order to give a proper treatment of coordination and of quantified NPs in object position, the AB calculus itself is insufficient. Instead of using a more sophisticated core syntactic calculus, we use *ad hoc* schemes that adds elements to the lexicon given in figure 2. These schemes add more categories to lexical items of two kinds:

1. Lexical items of categories that contain the category $s^F(s/np)$, of quantifiers in subject position. The scheme replaces this category by the corresponding category for quantifiers in object position: $(s \setminus np)^F((s \setminus np)/np)$. For instance: we add to the determiner *every* the category $((s \setminus np) \setminus^+ ((s \setminus np)/np)) / ^- n$.
2. Coordinators of category $(s \setminus s)^F s$ are also given a category $(A \setminus A)^F A$, for each category A in the (finite!) category closure by the AB calculus of categories in the lexicon obtained in step 1.

Word(s)	Category
every	$(s/^+(s\backslash np))/^-n$
no	$(s/^- (s\backslash np))/^-n$
some	$(s/^+(s\backslash np))^+n$
at least	$((s/^+(s\backslash np))^+n)/^-num$
at most	$((s/^- (s\backslash np))/^-n)^+num$
exactly	$((s/(s\backslash np))/n)/num$
two,three,four	num
student, teacher, person	n
boys, girls, people	n
ran, walked, smiled, moved	$s\backslash np$
hugged, kissed, touched, admired	$(s\backslash np)/np$
tall, short, young, old	$n/^Rn$
deliberately, yesterday	$(s\backslash np)^R(s\backslash np)$
who	$(n\backslash n)^C(s\backslash np)$
and	$(s\backslash s)^Cs$

Table 2: a lexicon

$$\begin{array}{c}
\text{every} \\
| \\
\frac{(s/^+(s\backslash np))/^-n}{s/^+(s\backslash np)} \quad \alpha_n \quad /E \\
\frac{\frac{\quad}{s} \quad \beta_{s\backslash np}}{s} \quad /E \\
\hline
\boxed{\alpha_n} \leq \boxed{\beta_{s\backslash np}} \quad \forall
\end{array}$$

Figure 1: postulate on *every*

We do not give full formalizations of these schemes here, since they are used only for our wish to use a simplest version of categorial grammar so to concentrate on the problems of computing order statements. A more comprehensive syntactic calculus (e.g. as in Carpenter (1997)) would eliminate these schemes altogether.

4.3 Lexical/extra-logical order statements

We postulate the following order statements between derivation trees of simple expressions (for simplicity, only the expressions are given, without the corresponding derivation trees in the grammar):

- *exactly* = *at least* and *at most*
- *two* \leq *three* \leq *four*
- *student* \leq *person*, *teacher* \leq *person*, *boys* \leq *people*, *girls* \leq *people*
- *ran* \leq *moved*, *walked* \leq *moved*
- *hugged* \leq *touched*, *kissed* \leq *touched*

Another *ad hoc* order relation is derived by the presence of the determiner *every* and is given in figure 1. This rule is used to enrich the system with more order relations derived from natural language assumptions (in the form ‘*every x y*’). A more complete treatment of quantifiers would eliminate this rule.

4.4 Examples

Example 1 Sánchez (1991) observes that in a premise sentence the number of upward/downward monotonic functions that take scope over a given expression determines whether a replacement of this expression by a "bigger"/"smaller" expression is truth-preserving. For instance, consider the two sound inferences below.

- (1) $\frac{\text{every student kissed every teacher}}{\text{every student kissed every tall teacher}}$
 (2) $\frac{\text{every student who kissed every tall teacher smiled}}{\text{every student who kissed every teacher smiled}}$

In inference (1), the object noun phrase *every teacher* can be replaced by *every tall teacher*, preserving the truth of the premise. This is by virtue of the fact that the derivation trees for the nominals *teacher/tall teacher*, which are ordered according to the Modification axiom, are arguments of *every* in a position that is downward monotonic, and the whole object NP is in the scope of the noun phrase *every student* in a position that is upward monotonic. By contrast, in inference (2) the same noun phrase appears in a downward monotonic position: the nominal *student who kissed every tall teacher* is a first argument of *every*, which is marked as downward monotone on this argument. Consequently, (2) shows a reverse inference relation to (1): in (2) the sentence with the modified nominal entails the sentence with the non-modified nominal, and is not entailed by it. The opposite directions for these inferences are not provable thanks to the soundness of the system.

Let us see how the above inference system derives this fact. First, the system derives a ' \leq ' order relation between the two verb phrases *kissed every teacher* and *kissed every tall teacher*. This is formally described in figure 2. In inference (1), the verb phrase *kissed every (tall) teacher* is an argument of the (lifted) noun phrase *every student*, whose category is marked with '+' on this argument. Therefore, the system directly proves the \leq ordering between the premise S_1 and the conclusion S_2 using the MON rule. Once such an order statement $S_1 \leq S_2$ is proven between sentences, the fact that the assertion of S_1 is given in the form $\top \leq S_1$ allows deriving by Transitivity $\top \leq S_2$, which is the assertion we need. By contrast, the inference in (2) is derived because the category of the relative *who* is marked by 'C', hence by fact 1 it behaves like a category marked by '+'. This derives using MON the statement *who kissed every teacher* \leq *who kissed every tall teacher*. Further, Function Replacement derives *student who kissed every teacher* \leq *student who kissed every tall teacher*. But using MON and the '-' sign on the category of *every* we now derive *every student who kissed every teacher* \geq *every student who kissed every tall teacher*, and another Function Replacement step derives a \leq order statement between the premise and the conclusion in (2).

Example 2 Another property of the system is that when a given premise is manipulated using the Monotonicity rule, different arguments of one function may require substitution by other arguments. This means that sometimes one application of the Monotonicity rule may not be enough to derive order relations even at one syntactic level, and intermediate derivation trees may need to be generated in the proof process. For instance, consider the sound inferences in (3).

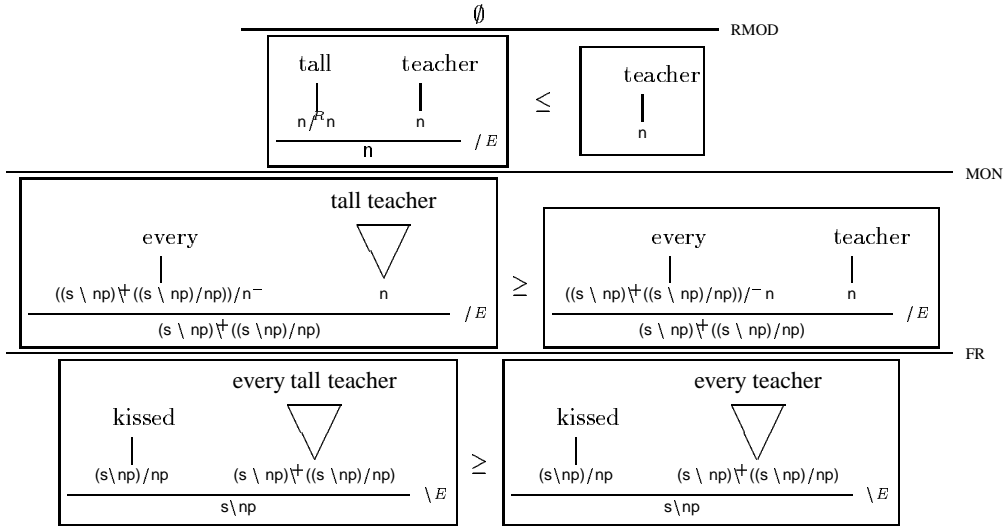


Figure 2: $\text{kissed every teacher} \leq \text{kissed every tall teacher}$

- (3)
$$\frac{\text{no teacher ran}}{\text{no tall teacher ran yesterday}} \quad \frac{\text{every teacher ran yesterday}}{\text{every tall teacher ran}}$$

Using only one application of the Monotonicity rule we cannot derive these inferences, and we need to use intermediate sentences like *no tall teacher ran* (or *no teacher ran yesterday*) and *every teacher ran* (or *every tall teacher ran yesterday*). The conclusion is then proven using the Transitivity rule.

Example 3 We can observe the use of more than one premise and the conjunctive meaning of the relative by proving the following inference.

- (4)
$$\frac{\text{every student smiled} \quad \text{no student who smiled walked}}{\text{no student walked}}$$

The proof is based on the following steps. First we obtain $\text{student} \leq \text{smiled}$ using the *ad hoc* rule on simple *every* sentences. Then by Reflexivity $\text{student} \leq \text{student}$ and by conjunction rule C2 we get $\text{student} \leq \text{student who smiled}$. Downward monotonicity of *no* and Function Replacement derive the statement $\text{no student who smiled walked} \leq \text{no student walked}$.

Example 4 One of the surprising features of this system is that using the monotonicity and conjunction rules we can prove some non-trivial inferences with *non-monotonic* expressions. Thijsse (1983) notes that many non-monotone expressions in natural language can be expressed as conjunctions of monotonic expressions. For instance, the non-monotonic *exactly* is equivalent to a conjunction of the monotonic *at least* and *at most*. This fact makes it possible to prove inferences like the following.

- (5)
$$\frac{\text{exactly four tall boys walked} \quad \text{at most four boys walked}}{\text{exactly four boys walked}}$$

The proof starts by using our lexical assumption $\text{exactly} = \text{at least and at most}$. Consecutive steps of Pointwise Coordination and Function Replacement lead to $\text{at least and at most four tall boys walked} = \text{at least four tall boys walked and at most four tall}$

boys walked. By C1 the derivation tree of this sentence is smaller than *at least four tall boys walked*, which in turn, by upward monotonicity of *at least four* and RMOD, is smaller than *at least four boys walked*. The conclusion $\top \leq \textit{at least four boys walked}$ and the second premise in (5) lead by C2 to $\top \leq \textit{at least four boys walked and at most four boys walked}$, which using some steps of FR and Pointwise Coordination lead to *least and at most four boys walked*, which as before is equal to the conclusion in (5).

5 Proof search algorithm

The algorithm we introduce below is a function **derive** that searches for a proof of an order statement $S_0 \leq S$ in the order calculus OC1 (definition 11), given a (possibly empty) finite set A of order statement premises $T_1 \leq T'_1, T_2 \leq T'_2$ etc. The trees T_1, T'_1, T_2, T'_2 etc. are all derivation trees in the AB calculus using a given lexicon (not necessarily the one given in section 4). This algorithm can be immediately used to search for a proof of an assertion $\top \leq S$ given a set of premises $\top \leq S_1, \top \leq S_2$ etc., where S, S_1, S_2 etc. are derivation trees of natural language sentences.

We introduce the function **derive** in two steps. First, we simplify the problem by assuming that the order premises in A can be used to replace *any subtree* of S_0 in order to derive S . This means we ignore two aspects of the order calculus: the generation of axiomatic order statements using the Restrictive Modification axiom, and the structural restrictions the Monotonicity and Function Replacement rules impose on the possibility to replace one subtree by another. Ignoring these two aspects, what we get is a tree generating *regular system* as defined in Brainerd (1969). After defining this tree generating system we introduce a version of the algorithm **derive** that decides whether an order statement $S_0 \leq S$ is **derive** by the premises in P and the regular system. A simple modification of this algorithm solves the original problem of proof search in the order calculus. For space considerations, we defer correctness proofs to the full version of this paper.

Let us first define a *tree generating regular system*, which can be viewed as a simplification of the order calculus.

Definition 13 (tree generating regular system) *A tree generating regular system is a pair $\langle \Sigma, P \rangle$, where Σ is a finite alphabet and P is a finite set $\{\langle T_i, T'_i \rangle : 1 \leq i \leq n\}$ s.t. for each i : T_i and T'_i are binary trees labelled with symbols from Σ , and $T_i \neq T'_i$.*

This system is used to transform a labelled tree S_0 into a labelled tree S by sequentially replacing occurrences of T_i by T'_i . This is formally defined in the next two definitions. In the notation below, when T is a labelled tree over an alphabet Σ , we use $\text{label}(T)$ to denote the label of T 's root, $\text{l}(T)$ and $\text{r}(T)$ to refer to the (possibly empty) left and right subtrees of T .

Definition 14 (derivation step) *Let S_0 and S be two binary trees labelled with symbols from Σ . We say that the relation $S_0 \xrightarrow{1} S$ (read: S_0 derives S in one step) holds in a grammar $\langle \Sigma, P \rangle$ iff one of the following holds:*

1. $\langle S_0, S \rangle \in P$.

2. $\langle S_0, S \rangle \notin P$, $S_0 \neq S$, $\text{label}(S_0) = \text{label}(S)$ and one of the following holds:
 - (a) $\text{l}(S_0) = \text{l}(S)$ and $\text{r}(S_0) \stackrel{1}{\Rightarrow} \text{r}(S)$.
 - (b) $\text{r}(S_0) = \text{r}(S)$ and $\text{l}(S_0) \stackrel{1}{\Rightarrow} \text{l}(S)$

Definition 15 (derivation sequence) Let S_0, S_1, \dots, S_m be a sequence of binary trees labelled with symbols from Σ . We say that the sequence is a derivation sequence in a grammar $\langle \Sigma, P \rangle$ iff one of the following holds:

1. $m = 0$
2. $m > 0$, $S_0 \stackrel{1}{\Rightarrow} S_1$, and S_1, \dots, S_m is a derivation sequence.

The algorithm given described below determines whether a labelled binary tree S is **derive** from S_0 using a regular system $\langle \Sigma, P \rangle$. The algorithm uses two additional parameters called A (for Assumptions) and $Goals$. The A parameter contains the indices i of the pairs $\langle T_i, T'_i \rangle$ that are available in the proof. The $Goals$ parameter keeps track of all the pairs of trees that appeared in recursive calls to the algorithm. Thus, the initial call is with $A = \{1, \dots, n\}$ and $Goals = \emptyset$. When attempting to derive S from S_0 we distinguish between two cases:

1. In the derivation of S that is being searched for, there is no full replacement of a tree using an assumption $\langle T_i, T'_i \rangle$. In such a case we can follow the definition of derivation sequences and search for proofs from $\text{l}(S_0)$ to $\text{l}(S)$ and from $\text{r}(S_0)$ to $\text{r}(S)$, provided $\text{label}(S_0) = \text{label}(S)$. This is done using a function called **subderive**.
2. In the derivation of S that is being searched for, a tree is replaced completely at some point. If the leftmost tree with this property is T_i , we have to **subderive** T_i from S_0 (since there is no full replacement left to T_i) and to **derive** S from T'_i . In this case we can obviously assume that there will be no reoccurring use of the same assumption for such a complete replacement. Hence, in the recursive call to **derive**, we omit i from the indices of available assumptions.

In each recursive call to **derive** or **subderive**, we add the present goal $\langle S_0, S \rangle$ to the set of goals in the $Goals$ parameter, a proof for which should not be searched for. This prevents repetition of attempts to prove the same goal, which may cause the algorithm not to terminate.

The **derive** and **subderive** functions are defined below for a regular system $\langle \Sigma, P \rangle$ s.t. $|P| = n$ and two binary trees, S_0 and S , marked by signs from Σ .

derive($S_0, S, A, Goals$) =

1. if $\langle S_0, S \rangle \in Goals$ then return **false**
2. $Goals' \leftarrow Goals \cup \{\langle S_0, S \rangle\}$
3. if **subderive**($S_0, S, Goals'$) then return **true**
4. for each $i \in A$:
 - if **subderive**($S_0, T_i, Goals'$) and **derive**($T'_i, S, A \setminus \{i\}, Goals'$)
 - then return **true**

5. return **false**

subderive($R_0, R, Goals$) =

1. if $R_0 = R$ then return **true**
2. if $\text{label}(R_0) = \text{label}(R)$ and **derive**($\text{l}(R_0), \text{l}(R), \{1 \dots n\}, Goals$) and **derive**($\text{r}(R_0), \text{r}(R), \{1 \dots n\}, Goals$) then return **true**
3. return **false**

We claim that the function **derive**($S_0, S, \{1, \dots, n\}, \emptyset$) terminates for all S_0, S and regular systems and returns **true** if and only if there is a derivation sequence of S from S_0 in the system.

The above algorithm is a decision procedure for regular systems, alternative to the proposal in Brainerd (1969), which is more convenient to our present purposes. It captures only the use of order premises to replace one subtree by another. The algorithm mimics the Transitivity rule of calculus OC1, and it can be further modified to yield a proof search algorithm for this calculus. The modifications are needed only in the **subderive** function, since the replacement of a whole tree using an order statement remains unchanged. If R_0 and R are two labeled derivation trees, we can determine if R is **subderived** from R_0 by checking the possibilities for changing derivation trees in the order calculus:

1. If R_0 is a restrictive modification construction, then omission of a modifier "enlarges" it to R' according to the RMOD axiom should allow to **derive** R from R' .
2. If R_0 is another function-argument construction, the possibilities to **subderive** R from R_0 depend on the other possible function-argument relations, according to the FR and MON rules.

subderive($R_0, R, Goals$) =

1. if $R_0 = R$ then return **true**
2. if $R_0 = \frac{\beta_{a|R_a} \alpha_a}{a}$ and **derive**($\alpha_a, R, \{1 \dots n\}, Goals$) then return **true**
3.
 - a. if $R_0 = \frac{\beta_{b|+a} \alpha_a}{b}$ and $R = \frac{\beta'_{b|*a} \alpha'_a}{b}$ and **derive**($\beta, \beta', \{1 \dots n\}, Goals$) and **derive**($\alpha, \alpha', \{1 \dots n\}, Goals$) then return **true**
 - b. if $R_0 = \frac{\beta_{b|*a} \alpha_a}{b}$ and $R = \frac{\beta'_{b|+a} \alpha'_a}{b}$ and **derive**($\beta, \beta', \{1 \dots n\}, Goals$) and **derive**($\alpha, \alpha', \{1 \dots n\}, Goals$) then return **true**
4.
 - a. if $R_0 = \frac{\beta_{b|-a} \alpha_a}{b}$ and $R = \frac{\beta'_{b|*a} \alpha'_a}{b}$ and **derive**($\beta, \beta', \{1 \dots n\}, Goals$) and **derive**($\alpha', \alpha, \{1 \dots n\}, Goals$) then return **true**
 - b. if $R_0 = \frac{\beta_{b|*a} \alpha_a}{b}$ and $R = \frac{\beta'_{b|-a} \alpha'_a}{b}$ and **derive**($\beta, \beta', \{1 \dots n\}, Goals$) and **derive**($\alpha', \alpha, \{1 \dots n\}, Goals$) then return **true**

5. if $R_0 = \frac{\beta_{b|*a} \quad \alpha_a}{b}$ and $R = \frac{\beta'_{b|*a} \quad \alpha'_a}{b}$ and **derive**($\beta, \beta', \{1 \dots n\}, Goals$)
and **derive**($\alpha_a, \alpha'_a, \{1 \dots n\}, Goals$) and **derive**($\alpha'_a, \alpha_a, \{1 \dots n\}, Goals$)
then return **true**
6. return **false**

6 Conclusions

The design of a proof system for natural language is of course a huge task, most of whose limits are presently still unknown. We believe that in order to explore these limits it is advisable to attempt directions that use as many insights as possible from logic, computer science and linguistics. In this paper we attempted to show that an attempt to use natural language syntax together with principles from higher order modeltheoretic semantics may be worthwhile. Much work is left to be done on extending the system, exploring possible completeness results and improving the proof search algorithm, which is presently of (at least) exponential complexity. On the other hand, the semantic and syntactic flexibility of the proposed system, and its conceptual simplicity, suggest that such an enterprise could be highly rewarding.

Acknowledgments

The parts of the second and third authors were supported by the fund for the promotion of research at the Technion, research no. 120-042, and by a BSF grant "Extensions and Implementations of Natural Logic". We are grateful to Raffaella Bernardi, Ed Keenan, Rani Nelken and Ian Pratt-Hartmann for their remarks on this work.

References

- Bernardi, R. (1999). Monotonic reasoning from a proof-theoretical perspective. In *Proceedings of Formal Grammar*.
- Brainerd, W. S. (1969). Tree generating regular systems. *Information and Control*, 14:217–231.
- Carpenter, B. (1997). *Type-Logical Semantics*. MIT Press, Cambridge, Massachusetts.
- Keenan, E. and Faltz, L. (1985). *Boolean Semantics for Natural Language*. D. Reidel, Dordrecht.
- McAllester, D. A. and Givan, R. (1992). Natural language syntax and first-order inference. *Artificial Intelligence*, 56:1–20.
- Partee, B. and Rooth, M. (1983). Generalized conjunction and type ambiguity. In Bauerle, R., Schwarze, C., and von Stechow, A., editors, *Meaning, Use and Interpretation of Language*. De Gruyter, Berlin.
- Purdy, W. C. (1991). A logic for natural language. *Notre Dame Journal of Formal Logic*, 32:409–425.
- Sánchez, V. (1991). *Studies on Natural Logic and Categorical Grammar*. PhD thesis, University of Amsterdam.
- Thijsse, E. (1983). On some proposed universals of natural language. In ter Meulen, A., editor, *Studies in Modeltheoretic Semantics*. Foris, Dordrecht.