
Order-Based Inference in Natural Logic

YAROSLAV FYODOROV

*Computer Science Faculty, Technion - IIT, Haifa, Israel.
E-mail: yaroslav@cs.technion.ac.il*

YOAD WINTER

*Computer Science Faculty, Technion - IIT, Haifa, Israel.
E-mail: winter@cs.technion.ac.il*

NISSIM FRANCEZ

*Computer Science Faculty, Technion - IIT, Haifa, Israel.
E-mail: 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 [7], we develop a small fragment that computes semantic order relations between derivation trees in Categorical Grammar. 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 decision procedure for provability is developed for a fragment of Natural Logic.

KEYWORDS: inference, monotonicity, natural logic, order, semantics

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. This paper 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.

This methodology means that “higher order” properties of natural language expressions become an integral part of the system. We believe that this is descriptively necessary, and may furthermore have distinct advantages over common techniques of translation into First Order Logic. To consider one example for this point, an adjective like *tall* is a *restrictive* modifier of nouns: any *tall N* is an *N*, for every noun *N*. Similarly, the degree word *very* is a restrictive modifier of *adjectives*: anything that is *very A* is *A*, for any adjective *A*. In first order logic, it is hard (or even impossible) to represent such inferential properties of modifiers like *tall* and *very*. In the proposed system of natural logic, one feature with the proper inference rules, represents this property for all modificational syntactic categories. The main challenge of Natural Logic is to find an optimal set of semantic features that allows to capture computationally as much as possible of the inference properties of natural language.

In [9] and [7], a similar conception of *Natural Logic* is used for describing 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 [2] and [1]. Therefore, Sánchez’ system and its current descendents 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 [8] are equivalent to conjunctions of monotonic expressions. After introducing this system, we show a simple algorithm that, under certain restrictions, finds a proof of any order relation that is provable in the order calculus.

Section 2 briefly discusses previous works on direct inference in natural language, and their relations with the present enterprise. Section 4 develops a categorial inference system that formalizes Sánchez’ work, and extends it to treat coordination. Section 5 gives a small lexicon for the system that is used for illustrating its application for some inferences with natural language sentences. Section 6 develops the proposed preliminary decision procedure for provability and sketches briefly elements of its correctness proof, which is given in more detail in the appendix. 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

[6] 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.

[5] 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 Semantic order relations in natural language

Semantic order relations between expressions in natural language are very common, and, as we will show in this paper, are useful for the derivation of inferences in an effective way. There are three origins for linguistic order relations

on which we would like to concentrate:

1. *Construction-based* order relations are generated from specific constructions or lexical items of the language.
2. *Lexical* order relations are relations between words that appear due to their lexical meaning.
3. *Imported* order relations are deduced by inference processes on input in natural language or are given as other forms of information.

As an example for order relations of the first kind consider the following entailments:

- (1) a. John is a tall student \Rightarrow John is a student
b. John is very tall \Rightarrow John is tall
c. John is a very tall student \Rightarrow John is a student.

These entailments reflect general properties of the words *tall* and *very* when they function as modifiers, which are informally written below as order relations, where **N** and **A** are a noun and an adjective respectively.

- (2) a. $tall(\mathbf{N}) \leq \mathbf{N}$
b. $very(\mathbf{A}) \leq \mathbf{A}$
c. $(very(tall))(\mathbf{N}) \leq \mathbf{N}$.

It should be pointed out that the relation in (2c), which reflects a general property of the complex expression *very tall*, logically follows from the ordering properties of the words *tall* and *very* as described in (2a) and (2b). This kind of construction-based ordering is central to the inference processes in natural logic, which we aim to capture within what we call an *Order Calculus*.

Lexical order relations reflect knowledge about the meaning of words. For instance:

- (3) a. $student \leq person$
b. $run \leq move$
c. $father \leq parent$

Of course, lexical relations may be more complex than order relations. For instance, the following proposition reflects the familiar connection between the transitive verb *kill* and the intransitive verb *die*. Standardly, the notations **kill'** and **die'** represent the denotations of these words.

- (4) $\forall x[\exists y[\mathbf{kill}'(y, x)] \rightarrow \mathbf{die}'(x)]$

The properties of the logic required to express lexical relations is a subject that we will not touch here. We simply assume that lexical *order* relations are given, as part of a lexical knowledge base that is not part of the order calculus itself. Additional 'imported' order relations may be inferred from other knowledge bases. For instance, a premise stating that *all swans are white* may be inferred from some description (linguistic or another) of animal life. Linguistic expressions like *every*, *all*, *always* etc. are useful for expressing

such order statements.

Let us now review some notions and techniques from natural language semantics, which are central for the development of the order calculus in the next section. We assume that expressions in natural language are associated with *types*. The primitive types are a finite set, a subset of which is identified as the set of *primitive partial order types* (PO types). In any given model, each primitive type is associated with a *domain*. The domains associated with primitive PO types are partially ordered by a given partial order relation. Non-primitive types inductively describe the functional compounds of primitive types. Standardly, Currying allows us to describe n -ary functions by unary types. A type that describes n -ary functions with a PO range is called a (non-primitive) *PO type*. An ordering relation on non-primitive PO domains is defined *pointwise* using the orderings of primitive PO domains. These notions are formally defined below.

DEFINITION 3.1 (types)

Let \mathbf{T}^0 be some finite set of primitive types. 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 $(\tau\sigma) \in \mathbf{T}$.

Standardly, types e (for ‘entities’) and t (for ‘truth values’) are among the primitive types.

DEFINITION 3.2 (domains)

For each primitive type $\tau \in \mathbf{T}^0$, let D_τ be the corresponding finite non-empty domain. Let these primitive domains be mutually disjoint. For each non-primitive type $(\tau\sigma) \in \mathbf{T} \setminus \mathbf{T}^0$, let $D_{\tau\sigma}$ be the set of functions from D_τ to D_σ .

DEFINITION 3.3 (PO types)

Let $\mathbf{T}_{po}^0 \subseteq \mathbf{T}^0$ be some finite set of primitive PO types s.t. $t \in \mathbf{T}_{po}^0$. The set of PO types is the smallest set $\mathbf{T}_{po} \subseteq \mathbf{T}$ that satisfies:

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

DEFINITION 3.4 (pointwise partial order)

For each primitive PO type $\tau \in \mathbf{T}_{po}^0$, let $'\leq_\tau'$ be a partial order on D_τ . For each non-primitive PO type $(\tau\sigma) \in \mathbf{T}_{po} \setminus \mathbf{T}_{po}^0$, the *pointwise partial order* $'\leq_{\tau\sigma}'$ on $D_{\tau\sigma}$ is defined by:

$x \leq_{\tau\sigma} y$ holds iff for every $z \in D_\tau$, $x(z) \leq_\sigma y(z)$ holds.

We sometimes use the notation $'x \geq y'$ instead of $'y \leq x'$, when convenient. Standardly, the domain D_t of truth values is the doubleton $\{0, 1\}$, for falsity and truth respectively, with the numerical partial order ‘less or equal’. This

boolean order imposes similar boolean orders on many other PO domains. The domain D_e of entities is arbitrarily chosen (with no partial order assumed).

After defining the elementary semantic notions with respect to order, we may move on to special behavior of functions with respect to order relations, and to some order-based notions that are important for the purposes of this paper.

DEFINITION 3.5 (restrictive function)

A function $f \in D_{\tau\tau}$, where τ is a PO type, is called *restrictive* iff for every $x \in D_\tau$: $f(x) \leq x$.

Using these definitions we get an intuitive account of entailments as in (1). The function that the adjective *tall* denotes is of type $(et)(et)$ – from a nominal of type et to a nominal of type et , and we assume that it is a restrictive function. Similarly, the function that the degree modifier *very* denotes is of type $((et)(et))((et)(et))$ – from an adjectival phrase of type $(et)(et)$ to an adjectival phrase of type $(et)(et)$, and we assume that it is a restrictive function as well. Consequently, the function that the adjectival phrase *very tall* denotes is of type $(et)(et)$. By restrictiveness of *very* we get the relation $very(tall) \leq_{(et)(et)} tall$. Hence by definition of ' $\leq_{(et)(et)}$ ', we get the relation $(very(tall))(student) \leq_{et} tall(student)$. By restrictiveness of *tall* we get: $tall(student) \leq_{et} student$. Hence, by transitivity of ' \leq_{et} ': $(very(tall))(student) \leq_{et} student$.

Of course, the notion of restrictive function can be extended to n -ary functions, and then it is sometimes said that a function of type 'ending with τ ' is restrictive on one of its τ type arguments. A special kind of restrictive functions is known as *greatest lower bound* (glb) functions. Consider the definition of the binary case, which is most useful for our purposes.

DEFINITION 3.6 (greatest lower bound)

A function $f \in D_{\tau(\tau\tau)}$, where τ is a PO type, is called *glb* iff for all $x, y, z \in D_\tau$ the following two conditions hold:

1. $(f(x))(y) \leq x$ and $(f(x))(y) \leq y$;
2. if $z \leq x$ and $z \leq y$ then $z \leq (f(x))(y)$.

The first requirement states that f is restrictive, or returns a lower bound, on both its arguments; the second requirement states that f returns a *greatest* lower bound on its arguments. A glb function is sometimes denoted by ' \wedge ' (boolean *meet*), and then ' $(f(x))(y)$ ' is abbreviated as ' $x \wedge y$ '.

A symmetric notion is the notion of least upper bound (lub) functions.

DEFINITION 3.7 (least upper bound)

A function $f \in D_{\tau(\tau\tau)}$, where τ is a PO type, is called *lub* iff for all $x, y, z \in D_\tau$ the following two conditions hold:

1. $(f(x))(y) \geq x$ and $(f(x))(y) \geq y$;

2. if $z \geq x$ and $z \geq y$ then $z \geq (f(x))(y)$.

The first requirement states that f returns an upper bound on both its arguments; the second requirement states that f returns a *least* upper bound on its arguments. A lub function is sometimes denoted by ' \vee ' (boolean *join*), and then ' $(f(x))(y)$ ' is abbreviated as ' $x \vee y$ '.

In natural language there are at least three kinds of glb functions:

1. Conjunctions. The standardly assumed meaning of conjunctions such as *dance and smile*, *Mary danced and John smiled*, and *every teacher and some student* is the glb of the meanings of the conjuncts.
2. Relative clauses. A 'subject oriented' relative clause such as *child who sneezed* (e.g. in the nominal *the child who sneezed*) is treated as a glb of the noun (*child*) denotation and the verb phrase (*sneezed*) denotation.
3. Intersective adjectives. Adjectives like *blue* and *pregnant* are often assumed to denote 'intersective functions': functions of type $(et)(et)$ that intersect their argument with an implicit argument of type et . For instance, the nominal *blue car* is synonymous with the nominal *car that is blue*, which is formed using a glb relative.

A lub function in natural language is the disjunction *or*: the standardly assumed meaning of disjunctions such as *dance or smile*, *Mary danced or John smiled*, and *every teacher or some student* is the lub of the meanings of the disjuncts.

In addition to general ordering properties of functions like restrictiveness, glb or lub, we are also interested in more specific properties. One of the useful properties of functions in natural language is *monotonicity*.

DEFINITION 3.8 (monotonicity)

Let σ_1 and σ_2 be PO types. A function $f \in D_{\sigma_1\sigma_2}$ is:

upward monotone iff for all $x, y \in D_{\sigma_1}$: $x \leq_{\sigma_1} y \Rightarrow f(x) \leq_{\sigma_2} f(y)$;

downward monotone iff for all $x, y \in D_{\sigma_1}$: $x \leq_{\sigma_1} y \Rightarrow f(x) \geq_{\sigma_2} f(y)$.

For instance, natural language *determiners* like *some* and *every* are treated as functions of type $(et)((et)t)$. The determiner *some* is analyzed as a function that is upward monotone on both its arguments, due to entailments as in (5). The determiner *every* is analyzed as a function that is downward monotone on its first argument and upward monotone on its second argument, due to entailments as in (6).

- (5)
 - a. Some tall student ran \Rightarrow Some student ran
 - b. Some student ran \Rightarrow Some student moved
- (6)
 - a. Every student ran \Rightarrow Every tall student ran
 - b. Every student ran \Rightarrow Every student moved

Note further that a determiner like *exactly one* is neither upward nor downward monotone on none of its arguments. This is verified by the following lack of entailments.

- (7) a. Exactly one tall student ran $\not\Rightarrow$ Exactly one student ran
 b. Exactly one student ran $\not\Rightarrow$ Exactly one student moved

4 The Categorical Inference Engine

In this section we introduce the system that will be used in order to derive semantic order relations between syntactic representations of expressions in natural language without resorting to their meanings. A categorial calculus, a variant of the simple AB-calculus, is responsible for assigning derivation trees to expressions of natural language. The categories in these trees are decorated by some *semantic features*, which give information on the order-based semantic properties of the expressions under the given syntactic analysis. An *order calculus* is responsible for deriving order statements between such decorated derivation trees. Especially, this calculus derives order relations between derivation trees of sentences, which correspond to semantic entailment relations between readings of sentences themselves.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 monotonicity, restrictiveness, or glb/lub behavior of expressions that derive them). A standard mapping from (decorated) categories to 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 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 *partial order categories* (PO categories), those categories that correspond to PO types. Order relations are defined for this set. We follow [4] and eliminate non-PO categories from the set of decorated categories. Here an henceforth, by *types* we refer to semantic types and by *categories* to syntactic categories.

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).

¹Here and henceforth, when we talk about entailment relations, we also consider cases where there are two or more premises. In the next section it will become clear how the system captures such entailments.

DEFINITION 4.1 (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 \mathbf{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 4.2 (PO categories)

The set \mathbf{CAT}_{po} of PO categories is defined by $\{A \in \mathbf{CAT} : \mathbf{type}(A) \in \mathbf{T}_{po}\}$ - the set of categories whose type is PO.

We use annotations on elements from the set of PO 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 $A/F B$ that is assigned to an expression α means that α is upward/downward monotone, restrictive etc.

Monotonicity: any category A/B or $A \setminus B$ where both A and B are PO categories 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: Any PO category that corresponds to a type $\tau\tau$ is a category of a *semantic modifier* and can be marked by 'R' to describe the fact that its semantic function is restrictive. For instance, a restrictive adverb like *yesterday*, in its role as a modifier of intransitive verbs, will be assigned the decorated category $(s \setminus np) \setminus^R (s \setminus np)$. A restrictive adjective like *tall*, in its role as a nominal modifier, will be assigned the decorated category $n \setminus^R n$.

Conjunction/Disjunction: A PO 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 ($=\text{glb/lub}$). For instance, the relative pronoun *who* will get the marking $(n \setminus n) \setminus^C (s \setminus np)$ for describing its conjunctive role in relative clauses missing a subject (e.g. as in *a child who walks*).

The set \mathbf{CAT}_d of *decorated categories* contains the PO categories, possibly

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 4.10 below). A more comprehensive treatment would allow decorating a category by a *set* of semantic features.

marked for monotonicity, conjunction/disjunction or restrictiveness. A category that is marked for conjunction or disjunction does not have any other marking.³

DEFINITION 4.3 (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. $\mathbf{CAT}_{po} \subseteq \mathbf{CAT}_d$. For any $A \in \mathbf{CAT}_{po}$: $\mathbf{type}^d(A) = \mathbf{type}(A)$.
2. If $A \in \mathbf{CAT}_d$ and $B \in \mathbf{CAT}_d$ then $A/F B \in \mathbf{CAT}_d$ and $A \setminus^{\epsilon} B \in \mathbf{CAT}_d$, where F is in the set $\{+, -, R, C, D, \epsilon\}$ (ϵ is the empty marking) and the following conditions hold:
 - (a) $\mathbf{type}^d(A/F B) = \mathbf{type}^d(A \setminus^{\epsilon} B) = (\mathbf{type}^d(A) \mathbf{type}^d(B))$,
 - (b) If $F = 'R'$ then $\mathbf{type}^d(A) = \mathbf{type}^d(B)$,
 - (c) If $F = 'C'$ or $F = 'D'$ then:
 - (i) If $A = A_1/F' A_2$ or $A = A_1 \setminus^{F'} A_2$ then $F' = \epsilon$.
 - (ii) If $\mathbf{type}^d(B) = \tau$ then $\mathbf{type}^d(A) = (\tau\tau)$, and $A = C/\epsilon D$ or $A = C \setminus^{\epsilon} D$.

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

We define two notions of equivalence between decorated categories. *Formal* equivalence between two decorated categories means that they are derived from the same standard syntactic category and may only be different in their semantic annotations. Two decorated categories are called *semantically equivalent* if their semantic type is the same. The equivalence classes of categories that are formed under formal equivalence are the standard categories of applicative categorial grammar. We assume that the syntax is insensitive to semantic annotations, and therefore these equivalence classes are relevant only for the definition of the syntactic inference rules in the grammar.⁴ Semantic equivalence between categories means that denotations of syntactic representations of expressions of these categories are semantically comparable.

DEFINITION 4.4 (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 $B_1 \equiv_f B_2$.

³The restrictiveness and upward monotonicity of conjunction and the upward monotonicity of disjunction will follow from the inference rules on these categories.

⁴We do not treat here the case of negative polarity items like *any* and *ever*, which are commonly assumed to be sensitive to monotonicity marking. For an account of NPI within a version of natural logic see [2].

DEFINITION 4.5 (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 $\mathbf{type}^d(A) = \mathbf{type}^d(B)$.

To give an example, it is standard to assign the type e to the primitive category \mathbf{np} (for noun phrases) and to give the type et to the primitive category \mathbf{n} (for nouns). Consequently, the categories \mathbf{n} and $\mathbf{s}\backslash\mathbf{np}$ (for intransitive verbs) become semantically equivalent. Under the assumption above that the grammar assigns the category $(\mathbf{n}\backslash\mathbf{n})/{}^c(\mathbf{s}\backslash\mathbf{np})$ to the word *who*, the calculus is therefore able to compare the derivation tree of the nominal *child who walks slowly* with the derivation trees of the verb *walks*, and to prove that the former is ‘smaller than’ the latter, although they are of different syntactic categories.

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 $A\backslash B$) 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 4.6 (AB calculus)

For any two decorated categories $A/*B$ (or $A\backslash B$) and $B' \equiv_f B$ we have the following (back)slash elimination rules.

$$\frac{A/*B \quad B'}{A} /E \quad \frac{B' \quad A\backslash B}{A} \backslash E$$

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

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

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 $\bar{\mathbf{s}}$, the use of which will become clear in the sequel.

DEFINITION 4.7 (lexicon)

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

DEFINITION 4.8 (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 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 $A \equiv_s B$.

Notational conventions: ' $T_2 \geq T_1$ ' instead of ' $T_1 \leq T_2$ '; ' $T_2 \equiv T_1$ ' instead of ' $T_1 \leq T_2$ and $T_2 \leq T_1$ '. This (semantic) equivalence between derivation trees should be distinguished from formal or semantic equivalence between categories as defined above. Note that all the decorated categories in \mathbf{CAT}_d are based on PO categories in \mathbf{CAT}_{po} . 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. Note further that the notation ' \leq ' is now treated as a syntactic relation between trees, and not as a semantic relation between denotations as in section 3.

Having specified the item form in the system, we can now move on 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 consider here 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 $f \equiv g$ is a monotonic function and an ordering is given between x and y ; and (ii) *Function replacement* – where $x \equiv y$ 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 reflect the fact that conjunction/disjunction are *greatest lower bound/least upper bound* operators with respect to the ' \leq ' relation. An additional rule, combining coordination and function application, reflects our assumption that all coordinators

in natural language are *pointwise operators*.

The *order calculus* embodies these assumptions within a grammar induced by the AB calculus. In the following definition of this calculus, the notation Ψ_Y stands for a derivation tree Ψ where the derived category (at the root) is Y .

DEFINITION 4.9 (order calculus (OC))

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

$$\text{Monotonicity: } \frac{\alpha_{B'} \leq \beta_{B''}}{\frac{\frac{\gamma_{A|\pm B} \quad \alpha_{B'}}{A} \mid_E \leq \frac{\gamma_{A|\pm B} \quad \beta_{B''}}{A} \mid_E}{\leq} \text{ MON}}$$

' \leq ' and ' \geq ' for '+' and '-' respectively; $B \equiv_f B' \equiv_f B''$

$$\text{Function Replacement: } \frac{\alpha_{A|*B} \leq \beta_{C|*D} \quad \gamma_{B'} \equiv \delta_{D'}}{\frac{\frac{\alpha_{A|*B} \quad \gamma_{B'}}{A} \mid_E \leq \frac{\beta_{C|*D} \quad \delta_{D'}}{C} \mid_E} \text{ FR}}$$

where $A \equiv_s C$, $B \equiv_s D$, $B \equiv_f B'$, $D \equiv_f D'$.

$$\text{Restrictive Modification: } \frac{\emptyset}{\frac{\frac{\alpha_{A'|R A} \quad \beta_{A''}}{A'} \mid_E \leq \beta_{A''}} \text{ RMOD}}$$

where $A \equiv_s A' \equiv_s A''$.

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

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

$$\text{Disjunction: } \frac{\emptyset}{T \leq \frac{\frac{\frac{\alpha_{(A|B)|^D C} \quad \gamma_C}{A|B} \mid_E \quad \beta_B \mid_E}{A} \leq \alpha'_{A'}} \text{ D1} \quad \frac{\beta_B \leq \alpha'_{A'} \quad \gamma_C \leq \alpha'_{A'}}{\frac{\frac{\alpha_{(A|B)|^D C} \quad \gamma_C}{A|B} \mid_E \quad \beta_B \mid_E \leq \alpha'_{A'}} \text{ D2}}$$

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

Pointwise Coordination:

$$\begin{array}{c}
 \emptyset \\
 \hline
 \frac{\frac{\alpha_{((A|B)|(A|B))|^{C/D}(A|B)} \quad \beta_{A|B}}{(A|B)|(A|B)} \quad |^E \quad \gamma_{A|B}}{A|B} \quad |^E \quad \delta_B}{A} \quad |^E \\
 \hline
 \frac{\frac{\alpha_{(A|^{C/D}A)|A}}{A|A} \quad |^E \quad \frac{\beta_{A|B} \quad \delta_B}{A} \quad |^E}{A} \quad |^E \quad \frac{\gamma_{A|B} \quad \delta_B}{A} \quad |^E}{A} \quad |^E
 \end{array}
 \quad \stackrel{\text{PWC}}{=}$$

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

The Pointwise Coordination (PWC) rule presupposes that any conjunctive/-disjunctive expression of category $((A|B)|(A|B))|^{C/D}(A|B)$ can also be analyzed as category $(A|^{C/D}A)|A$ respectively, and conversely: any expression of category $(A|^{C/D}A)|A$ is also of category $((A|B)|(A|B))|^{C/D}(A|B)$, where B is a category derivable in the AB calculus from a given lexicon. This PWC rule partly follows from the Function Replacement, Conjunction and Disjunction rules. When a conjunction appears in the function as in $(f \wedge g)(x)$ we can apply C1 and conclude $f \wedge g \leq f$ and $f \wedge g \leq g$. Using function replacement we conclude $(f \wedge g)(x) \leq f(x)$ and $(f \wedge g)(x) \leq g(x)$. Hence by C2: $(f \wedge g)(x) \leq f(x) \wedge g(x)$. However, the other direction of the inequality does not follow without PWC. Symmetrically, when a disjunction appears in the function as in $(f \vee g)(x)$ we can conclude by D1, D2 and FR: $f(x) \vee g(x) \leq (f \vee g)(x)$, but PWC is needed for the other direction of the inequality.

Note the following simple fact about this system.

FACT 4.10

(i) Any conjunctive category $(A|B)|^C C$ can be replaced by $(A|^R B)|^C C$ with no additional inferences derived by this notation. (ii) Any conjunctive/disjunctive category $(A|B)|^{C/D} C$ can be replaced by $(A|^+ B)|^+ C$ with no additional inferences derived by this notation.

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 and disjunction are both upward monotone on both

arguments. This fact justifies our assumption in the definition of decorated categories that conjunctive and disjunctive categories are otherwise unmarked on their arguments.

Sketch of proof: (i) is directly by C1.

For (ii), the following inference (in abbreviated format) shows that conjunction is upward monotonic in its second argument:

$$\frac{\frac{\emptyset}{y \wedge z \leq y} \text{ C1} \quad \frac{\frac{\emptyset}{y \wedge z \leq z} \text{ C1} \quad z \leq z' \text{ (assumption)}}{y \wedge z \leq z'} \text{ TRANS}}{y \wedge z \leq y \wedge z'} \text{ TRANS}$$

The reasoning for the other cases with conjunction and disjunction is similar.

To illustrate the use of the order calculus, 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)⁵
- $yesterday(ran) \leq ran$ (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 such a term is stipulated with no free variables.

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 the extensional typed lambda calculus 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 lambda expression assigned to a syntactic proof of a 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

⁵Non-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.

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 semantic relation ' \leq ' defined in section 3. Using these standard notions, it is easy to show that the rules in the two order calculi defined above are sound relative to a standard semantics. A harder question is whether the system is complete, that is: for a given lexicon of words with decorated categories, does the system derive all the valid order relations over this lexicon that semantically follow from the decorations? Whether the order calculus above is complete in this sense, or extendable to a complete proof system, is currently unknown to us.

5 The order calculus as an inference system for natural language

In this section we illustrate how the order calculus 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 used as additional *assumptions* of the order calculus. We describe these three parts and then give some examples for derivations in the resulting system.

5.1 Assertions as order relations

The premises and the goal in the system are 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 ('top') is notation for the following special derivation tree.

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

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, the formulas of the order calculus that represent assertions of natural language sentences are of the following form.

$$\top \leq \alpha_{\mathfrak{s}}, \quad \alpha_{\mathfrak{s}} \text{ is a derivation tree deriving } \mathfrak{s}.$$

5.2 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)

Word(s)	Category
every	$(s/\bar{s}^+(s\backslash np))/^-n$
no	$(s/\bar{s}^-(s\backslash np))/^-n$
some	$(s/\bar{s}^+(s\backslash np))/^+n$
at least	$((s/\bar{s}^+(s\backslash np))/^+n)/^-num$
at most	$((s/\bar{s}^-(s\backslash np))/^-n)/^+num$
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	$n/\bar{R}n$
very, extremely	$(n/n)/\bar{R}(n/n)$
deliberately, yesterday	$(s\backslash np)/\bar{R}(s\backslash np)$
who	$(n\backslash n)/^C(s\backslash np)$
and	$(s\backslash s)/^C_s$

TABLE 1. a lexicon

and v (for natural numbers). The PO primitive categories in \mathbf{T}_{po}^0 are t and v . The set of primitive categories with their corresponding types (assigned by the \mathbf{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 1.

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 1. These schemes add more categories to lexical items of two kinds:

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

In addition, we add the following restriction on the usages of the coordinator categories:

- A derivation tree of category $(X|Y)|^{C/D}Z$ appears only in derivation trees where is combines first with derivation trees Z and Y .

This assumption rules out derivation trees where the categories $(X|Y)|^{C/D}Z$ combines with $(T|(X|Y)|Z)$ or where $(X|Y)|^{C/D}Z$ combines first with Z and the result $X|Y$ combines with $T|(X|Y)$.

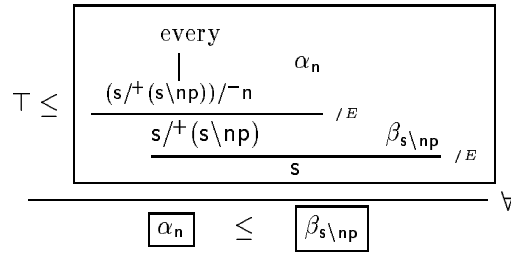


FIG. 1. postulate on *every*

We do not give full formalizations of these *ad hoc* schemes and assumption here. 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 [3]) would eliminate these schemes altogether.

5.3 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*'). Of course, a more complete treatment of quantifiers would eliminate this rule.

5.4 Examples

Example 1 [7] 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.

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

In inference (8), 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 (9) 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, (9) shows a reverse inference relation to (8): in (9) 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 (8), 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 (9) is derived because the category of the relative *who* is marked by 'C', hence by fact 4.10 it behaves like a category marked by '+'. This derives the order 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 (9).

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, and intermediate derivation trees may need to be generated in the proof process. For instance, consider the sound inferences in (10).

$$(10) \quad \frac{\text{no teacher ran}}{\text{no tall teacher ran yesterday}} \qquad \frac{\text{every teacher ran yesterday}}{\text{every tall teacher ran}}$$

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. For

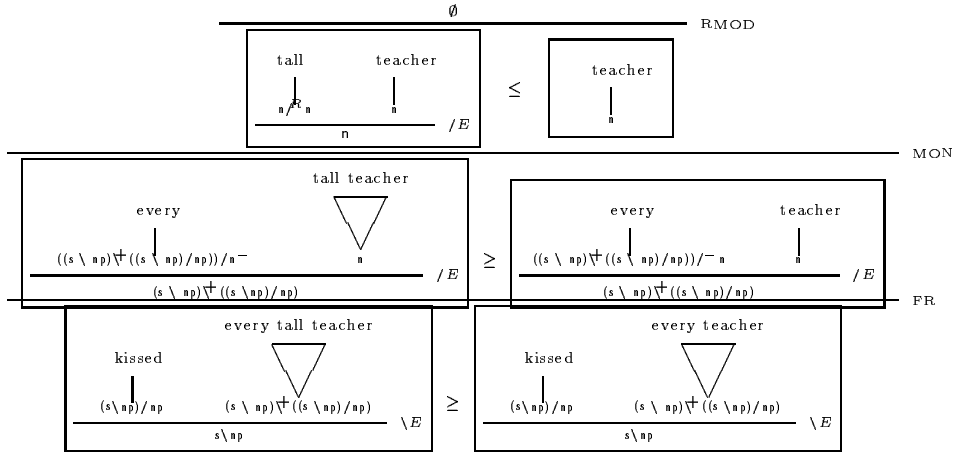


FIG. 2. $\text{kissed every teacher} \leq \text{kissed every tall teacher}$

instance, the order statement *every teacher ran yesterday* \leq *every tall teacher ran yesterday* is directly derived using RMOD and downward monotonicity of the determiner *every* on its first argument. The order statement *every tall teacher ran yesterday* \leq *every tall teacher ran* is directly derived using RMOD and upward monotonicity of *every* on its second argument.

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

$$(11) \quad \frac{\text{every student smiled} \quad \text{no student who smiled walked}}{\text{no student walked}}$$

The proof is based on the following steps. First we obtain *student* \leq *smiled* using the *ad hoc* rule on simple *every* sentences. Then by Reflexivity *student* \leq *student* and by conjunction rule C2 we get *student* \leq *student who smiled*. Downward monotonicity of *no* and Function Replacement derive the statement *no student who smiled walked* \leq *no student walked*. The formal process is given in figure 3.

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. [8] 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.

$$(12) \quad \frac{\text{exactly four tall boys walked} \quad \text{at most four boys walked}}{\text{exactly four boys walked}}$$

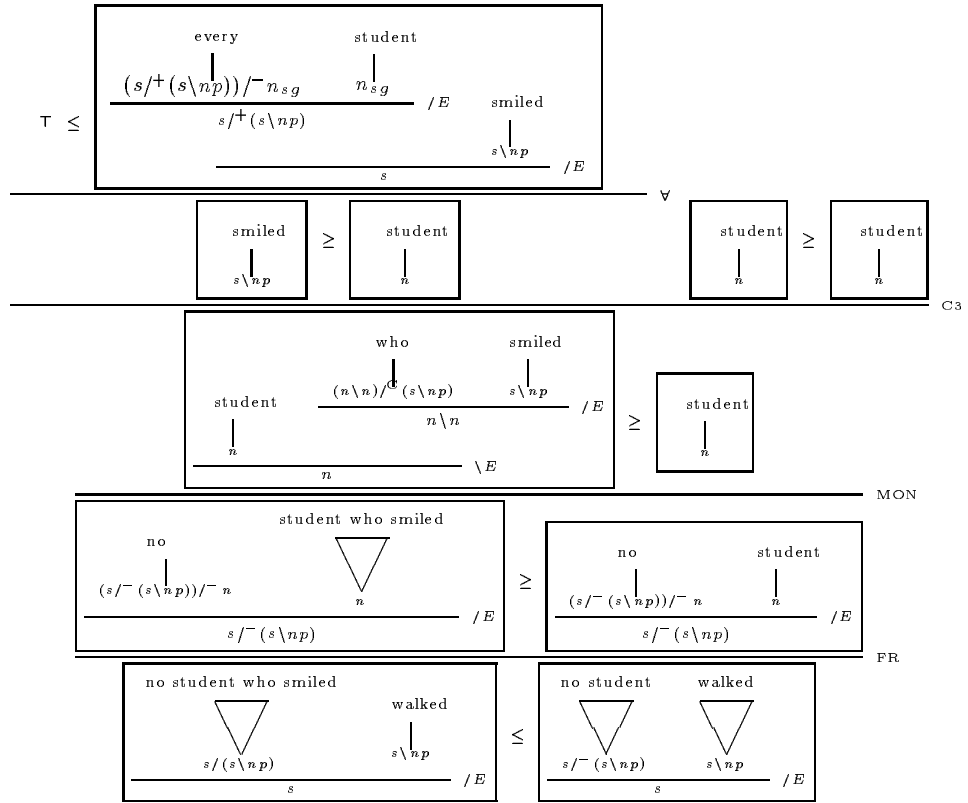


FIG. 3. conjunctive behavior of relatives

The proof starts by using our lexical assumption *exactly = at least and at most*. Consecutive steps of Pointwise Coordination and Function Replacement lead to *at least and at most four tall boys walked = 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 $\top \leq$ *at least four boys walked* and the second premise in (12) lead by C2 to $\top \leq$ *at least four boys walked and at most four boys walked*, which using some steps of FR and Pointwise Coordination lead to *at least and at most four boys walked*, which as before is equal to the conclusion in (12).

6 Decision procedure for provability

6.1 Description of the algorithm

The algorithm we introduce below is a recursive function **derive** that decides whether a proof of an order statement $T_0 \leq T$ exists in the order calculus, given a (possibly empty) finite set A of n order statement premises $T_1 \leq T'_1$, $T_2 \leq T'_2$ etc. The trees $T_0, T, T_1, T_1, T_2, T'_2$ etc. are all derivation trees in the AB calculus using a given lexicon (not necessarily the one given in section 5). 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.

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 $Goals = \emptyset$. A proof of an order relation $T_0 \leq T$ can be viewed as a series of order relations $T_0 = V_1 \leq \dots \leq V_m = T$, where the proof of each order relation $V_j \leq V_{j+1}$ does not involve any Transitivity step $V_j \leq U \leq V_{j+1}$. In attempting to prove an order relation between T_0 and T , we distinguish between two cases:

1. In the order relation $T_0 = V_1 \leq \dots \leq V_m = T$ that is being searched for, there is no full replacement of a tree using a premise. In other words: no $i \in \{1..n\}$ and no $j \in \{1..m-1\}$ satisfy $V_j = T_i$ and $V_{j+1} = T'_i$. In this case, the algorithm attempts the following possibilities:
 - (a) The possibilities to recursively prove the order relation $T'_0 \leq T$, where T'_0 is a subtree of T_0 that is guaranteed to satisfy $T_0 \leq T'_0$ by one of the rules of the order calculus. Symmetrically: the possibilities to recursively prove the order relation $T_0 \leq T'$, where T' is a subtree of T that is guaranteed to satisfy $T' \leq T$ by one of the rules of the order calculus. These possibilities are searched, according to form of the rules, in a way that is described below.
 - (b) The possibilities to prove directly the order relation $T_0 \leq T$ using the FR (function replacement) and MON (monotonicity) rules.

This process is performed using a function called **subderive**.

2. In the order relation $T_0 = V_1 \leq \dots \leq V_m = T$ that is being searched for, there is a full replacement of a tree using a premise. In other words: for some $i \in \{1..n\}$ and some $j \in \{1..m-1\}$, $V_j = T_i$ and $V_{j+1} = T'_i$. Assuming that the leftmost occurrence of such a replacement is a replacement of a tree T_i by T'_i using the premise $T_i \leq T'_i$, the algorithm attempts to derive T_i from T_0 (using the function **subderive**, since there is no full replacement left to T_i) and to **derive** T from T'_i .

In each recursive call to **derive** or **subderive**, we add the present goal $\langle T_0, T \rangle$ to the set of goals in the *Goals* parameter. A proof for this goal is not searched

$$\frac{\frac{\frac{\emptyset}{\text{very}(\text{tall}) \leq \text{tall}}{\text{RMOD}}}{(\text{very}(\text{tall}))(\text{student}) \leq \text{tall}(\text{student})} \text{FR} \frac{\frac{\emptyset}{\text{tall}(\text{student}) \leq \text{student}}{\text{RMOD}}}{(\text{very}(\text{tall}))(\text{student}) \leq \text{student}} \text{TRANS}$$

FIG. 4. very tall student \leq student

for recursively, which prevents repetition of attempts to prove the same goal, which may cause the algorithm not to terminate.

Before introducing the algorithm itself, we make a small change in the proof system of the order calculus, which does not affect provability, but which is more convenient for the description of the algorithm. Consider the valid inference in the order calculus that is given in figure 4. Let us denote the derivation tree for *very tall student* in this figure by T_0 , and the derivation tree for *student* by T . The function category of T_0 is n/n , which is not marked as a restrictive modifier. Consequently, it is impossible to generate T directly from T_0 using the RMOD rule. In addition, T is a simple word, without a function-argument structure, and therefore no direct use of the Function Replacement or Monotonicity rules is possible. The needed order relation between the trees for *very tall* and *tall* are therefore not derived using the algorithm as sketched above. This problem is more general: it would reappear even if *very* were not defined as restrictive but However, there is a simple way to change the RMOD rule in a way that allows us to derive this order relation. Consider the following simple fact.

FACT 6.1

A function f of type $\tau\tau$, where τ is a poset type, is restrictive if and only if $f \leq id$, where id is the identity function of type $\tau\tau$.

We replace the RMOD axiom by the following rules:

$$\frac{\frac{\emptyset}{\alpha_{A'|R}A \leq id_{A'|A}} \text{RMOD}_1}{\frac{\frac{\alpha_{A'|*}A \leq id_{A'|A}}{\text{RMOD}_2} \frac{\frac{\alpha_{A'|*}A \quad \beta_{A''}}{A'}{|E}}{\leq \beta_{A''}}}{\text{where } A \equiv_s A' \equiv_s A''}}$$

We assume that for each pair of PO categories A and A' that are in the (finite) category closure of the lexicon by the AB calculus, there exists a word $id_{A'|A}$. Note that this restatement of RMOD is only for the sake of the statement of the algorithm. To show that this does not change the order calculus in any significant way, it is sufficient to observe that the replacement of RMOD by RMOD₁ and RMOD₂ and the addition of the needed *id* elements to the lexicon do not change the provability of order relations that do not involve *id*. The *id*

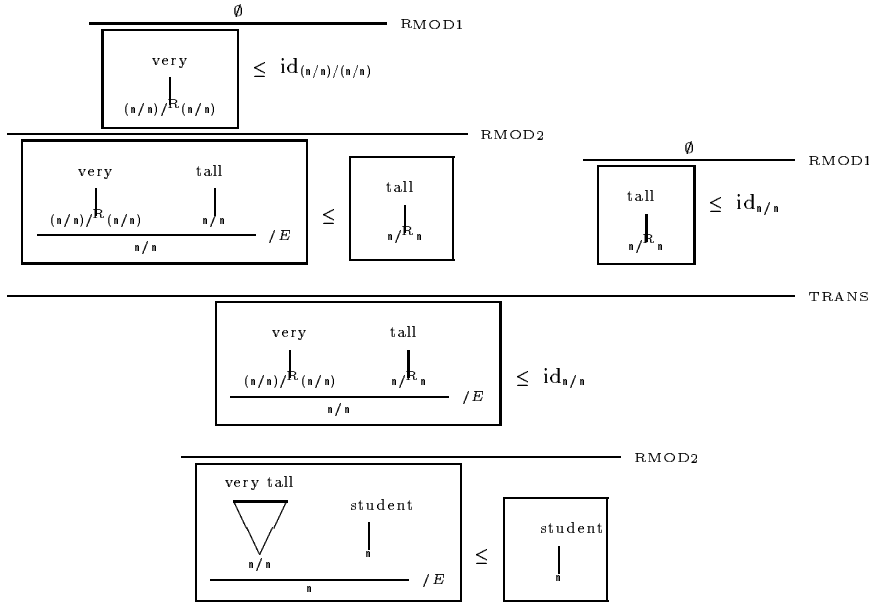


FIG. 5. very tall student \leq student

elements do have of course a natural semantic correlate: the identity functions in the respective domains. The proof of the order relation between T_0 and T is now as given in figure 5. And this proof is found by the algorithm described below.

To describe the algorithm itself, let us first observe the following fact about the axioms C1, PWC and RMOD_1 . Given a left hand tree T_l in the conclusion $T_l \leq T_r$ of these rules, the tree T_r can be defined as a function of T_l . Let us call the corresponding functions for these rules $f_{C1,r}$, $f_{PWC,r}$ and $f_{\text{RMOD}_1,r}$.⁶ The letter r designates the fact that the right hand side of the conclusion is generated from the left hand side. In a similar way we can define the functions $f_{D1,l}$ and $f_{PWC,l}$ that generate the left hand sides of the conclusions of D1 and PWC from their right hand side. Observe now a similar point about the derivation rules C2 and D2. In C2, the premises can be expressed as a function of the conclusion $T_l \leq T_r$. Let us therefore denote the premises by $T_l \leq f_{C2,1}(T_r)$ and $T_l \leq f_{C2,2}(T_r)$. Similarly, in D2, we denote the premises by $f_{D2,1}(T_l) \leq T_r$ and $f_{D2,2}(T_l) \leq T_r$.

The function **subderive** uses these functions in trying to extract provable “smaller” order relations from the two derivation trees in the goal $T_0 \leq T$.

⁶These functions can be computed in a constant time independent on their arguments. Note further that the function $f_{\text{RMOD}_1,r}$ needs only the category C derived by its argument, a derivation tree T , so that $f_{\text{RMOD}_1,r}(T)$ is id_C .

The functions for the coordination rules C1, C2, D1, D2 and PWC are used in a straightforward way to search for such “smaller” order relations. Similarly for the restrictive modification rules RMOD₁ and RMOD₂. The Monotonicity and Function Replacement rules are used in a combined manner: when two function-argument structures are given, the functions are compared to each other, and the arguments are compared to each other according to monotonicity marking of the functions, to search for a proof of an order relation between the two structures. We denote $\alpha \in \mathbf{MON} \uparrow$ or $\alpha \in \mathbf{MON} \downarrow$ when the monotonicity of the category derived in a derivation tree α is marked by + or – respectively. The **derive** and **subderive** functions are given below. Recall that A is the set of given premises and that $n = |A|$.

derive($T_0, T, Goals$) =

1. if $\langle T_0, T \rangle \in Goals$ then return **false**
2. $Goals' \leftarrow Goals \cup \{\langle T_0, T \rangle\}$
3. if **subderive**($T_0, T, Goals'$) then return **true**
4. for each $i \in \{1..n\}$:
 - if **subderive**($T_0, T_i, Goals'$) and **derive**($T'_i, T, Goals'$) then return **true**
5. return **false**

subderive($T_0, T, Goals$) =

1. if $T_0 = T$ then return **true**
2. for each $ax \in Axioms$:
 - 2.1 if $f_{ax,r}(T_0)$ is defined and **derive**($f_{ax,r}(T_0), T, Goals$) then return **true**.
 - 2.2 if $f_{ax,l}(T)$ is defined and **derive**($T_0, f_{ax,l}(T), Goals$) then return **true**.
3. 3.1 if $f_{D2,1}(T_0)$ and $f_{D2,2}(T_0)$ are defined and **derive**($f_{D2,1}(T_0), T, Goals$) and **derive**($f_{D2,2}(T_0), T, Goals$) then return **true**.
 - 3.2 if $f_{C2,1}(T)$ and $f_{C2,2}(T)$ are defined and **derive**($T_0, f_{C2,1}(T), Goals$) and **derive**($T_0, f_{C2,2}(T), Goals$) then return **true**.
4. if $T_0 = \frac{\alpha \quad \beta}{c}$ and $T = \frac{\gamma \quad \delta}{c}$ and **derive**($\alpha, \gamma, Goals$)
 - 4.1 if $\alpha \in \mathbf{MON} \uparrow$ or $\gamma \in \mathbf{MON} \uparrow$ and **derive**($\beta, \delta, Goals$) return **true**.
 - 4.2 if $\alpha \in \mathbf{MON} \downarrow$ or $\gamma \in \mathbf{MON} \downarrow$ and **derive**($\delta, \beta, Goals$) return **true**.
 - 4.3 if ($\alpha \in \mathbf{MON} \downarrow$ and $\gamma \in \mathbf{MON} \uparrow$) or ($\alpha \in \mathbf{MON} \uparrow$ and $\gamma \in \mathbf{MON} \downarrow$) return **true**.
 - 4.4 if **derive**($\beta, \delta, Goals$) and **derive**($\delta, \beta, Goals$) return **true**.
5. If $T_0 = \frac{\alpha_{A|A'} \quad \beta_{A''}}{A}$, where $A \equiv_s A' \equiv_s A''$ and **derive**($\alpha, id_{A|A'}, Goals$) and **derive**($\beta, T', Goals$) return **true**.

6. return false

Step 2 in the algorithm covers the C1, D1 and PWC axioms: for each of these axioms, the algorithm attempts to derive a an order relation between a modification of T_0 (in the case of D1 and PWC) and T , or between T_0 and a modification of T (in the case of C1 and PWC). Step 3 covers the derivation rules C2 and D2: it attempts to derive order relations between subtrees of T_0 (in the case of D2) and T , between T_0 and subtrees of T (in the case of C2). Step 4 covers the MON and FR rules, and step 5 covers the RMOD₁ and RMOD₂ rules.

6.2 Discussion

The reason for the *Goals* parameter in **derive** and **subderive** is to prevent a situation where the same goal is used in recursive steps of the algorithm, which may lead to non-termination. To see this, consider the following (artificial) example. Let the premises be the following, in a grammar where α' , β' and γ' are derivation trees of category C .

$$\text{Premise 1: } T_1 \leq T_1, \text{ where } T_1 = \alpha' \text{ and } T'_1 = \begin{array}{c} \alpha' \quad \beta' \\ \triangle \\ C \end{array}$$

$$\text{Premise 2: } T_2 \leq T'_2, \text{ where } T_2 = \begin{array}{c} \beta' \quad \beta' \\ \triangle \\ C \end{array} \text{ and } T'_2 = \gamma'.$$

Now consider a call to **derive**, with parameters $T_0 = \alpha'$, $T = \beta'$, and $Goals = \emptyset$. In step 4 of the algorithm, with $i = 1$, the call **subderive**($T_0, T_1, Goals'$) will return **true** because $T_0 = T_1 (= \alpha')$. The function **derive** will be recursively called with T'_1 and $T = \gamma'$. When step 4 is performed again, this time with $i = 2$, the call to **subderive** will be with $T_0 = T'_1$ and T_2 . Step 4 of **subderive** will now call **derive** with $\alpha = \alpha'$ and $\gamma = \beta'$ (the left subtrees of T'_1 and T_2 respectively). This pair is already in *Goals*, as it is the initial order statement we tried to prove, and consequently **derive** will return **false**. In the absence of the *Goals* parameters, the algorithm would not have terminated, because it would attempt to prove the order relation $\alpha' \leq \beta'$ again and again.

It is now possible to see the problem for the method of the algorithm with the original RMOD rule. A proof as in (4) cannot be found by our method, since there is no simple direct way to derive from the goal $(very(tall))(student) \leq student$ the needed subgoal $very(tall) \leq tall$. With the revised rules RMOD₁ and RMOD₂, step 5 in **subderive** generates the subgoal $very(tall) \leq id_{n|n}$, and for this subgoal, the subsequent **subderive** call generates the subgoal

$very \leq id_{(n|n)|(n|n)}$. The latter subgoal guarantees $very(tall) \leq tall$, although it is not needed to generate this specific order relation as a subgoal in the proof search process.

6.3 Correctness of the algorithm

In this subsection we sketch the main restrictions that guarantee correctness of the above algorithm, and the main step of the correctness proof in the appendix. The *soundness* of the algorithm, i.e. the fact that any order statement for which it returns **true** has a proof in the Order Calculus, follows quite directly from the way the algorithm was defined. *Termination* is guaranteed by two facts: first, given a goal $T_0 \leq T$ and a set of premises there is only a finite set of derivation trees that can appear as arguments of the **derive** function in the recursive search process for a proof of $T_0 \leq T$. Second, no pair of trees in this finite set appears more than once as an inner node of the search tree induced by the recursive search process in the algorithm. This situation is guaranteed by the *Goal* parameter of **derive**, in the way exemplified above. For more details see the appendix.

In order for the algorithm to be *complete*, i.e. to return **true** for each order statement that has a proof in the order calculus, we have to adopt certain assumptions about the lexicon and the premises in the system. Let us first observe a simple fact about the PWC axiom. Let us abbreviate the conclusion of this axiom as stating that $(f \text{ coor } g)(x) \equiv f(x) \text{ coor } g(x)$, where **coor** is a cross-categorical coordinator. We observe the following fact.

FACT 6.2

The order statements $(f \wedge g)(x) \leq f(x) \wedge g(x)$ and $(f \vee g)(x) \geq f(x) \vee g(x)$ are derivable in the order calculus without the PWC rule.

A sketch of the proof of the first part of this fact is the following simplified derivation in the order calculus.

$$\frac{\frac{\frac{\emptyset}{f \wedge g \leq f} \text{ C1}}{(f \wedge g)(x) \leq f(x)} \text{ FR} \quad \frac{\frac{\frac{\emptyset}{f \wedge g \leq g} \text{ C1}}{(f \wedge g)(x) \leq g(x)} \text{ FR}}{(f \wedge g)(x) \leq f(x) \wedge g(x)} \text{ C2}}$$

The proof of the second part is similar. Let us call usages of PWC to prove these order relations *trivial usages* of PWC.

Let us define now a *normal form* of a proof in the order calculus. We will claim that under certain assumptions, a normal form exists for every proof in the Order Calculus that has no usage of PWC (or only trivial usages). Then we classify a subset of normal forms that do not include any sequences of order relations of *mixed-monotonicity*. We claim that normal forms that satisfy this

restriction are arrived at by the algorithm, and speculate that the elimination of mixed monotonicity sequences (unlike the elimination of PWC!) from normal forms, is linguistically innocuous.

Let us first define three (non-disjoint) sets of inference rules and axioms in the Order Calculus:

1. *Right Productive* (R-PROD) rules are rules where the right hand tree T_r in the derived order relation $T_l \leq T_r$ is a function of the left hand tree T_l . These rules are REFL, PWC, RMOD₁, RMOD₂, C1, and D2.
2. *Left Productive* (L-PROD) rules are rules where the left hand tree T_l in the derived order relation $T_l \leq T_r$ is a function of the right hand tree T_r . These rules are REFL, PWC, C2 and D1.
3. *Subtree Replacement* (STR) rules are rules in which T_r is a simple replacement of subtrees in T_l , given order relations between those subtrees. These rules are MON and FR.

DEFINITION 6.3 (normal form)

Let $T \leq T'$ be an order statement that is derivable in the order calculus from the premises $T_1 \leq T'_1, T_2 \leq T'_2, \dots, T_n \leq T'_n$. Then the a *normal forms* of a proof of $T \leq T'$ is one of the following structures:

1. A *type 1* normal form is one of the following structures:

$$(a) \frac{\alpha_1 \boxed{\begin{array}{c} \nabla \\ S_1 \leq S'_1 \end{array}} \quad \alpha_2 \boxed{\begin{array}{c} \nabla \\ S_2 \leq S'_2 \end{array}} \quad \dots \quad \alpha_n \boxed{\begin{array}{c} \nabla \\ S_n \leq S'_n \end{array}}}{T \leq T'} R$$

where $R \neq TRANS$, $n \geq 0$ and $\alpha_1, \dots, \alpha_n$ are normal forms.

$$(b) \frac{\emptyset}{T_i \leq T'_i} PREM$$

where $T_i \leq T'_i$ is a premise.

2. A *type 2* normal form is the following structure:

$$\frac{\alpha_1 \boxed{\begin{array}{c} \nabla \\ \hline S_1 \leq S'_1 \end{array}} R_1 \quad \alpha_2 \boxed{\begin{array}{c} \nabla \\ \hline S_2 \leq S'_2 \end{array}} R_2 \quad \dots \quad \alpha_n \boxed{\begin{array}{c} \nabla \\ \hline S_n \leq S'_n \end{array}} R_n}{T \leq T'} TRANS^*$$

where $n \geq 2$, $\alpha_1, \dots, \alpha_n$ are type 1 normal forms, $T = S_1, S'_1 = S_2, S'_2 = S_3, \dots, S'_{n-1} = S_n, S'_n = T$ and the string formed by the rules $R_1 R_2 \dots R_n$ belongs to the regular language:

(PREM*, R-PROD*, STR*, L-PROD*) (PREM+, R-PROD*, STR*, L-PROD*)* PREM*.

In words, a normal form of a proof is a structure that satisfies the following requirements:

- All occurrences of the transitivity rule are “flattened”.
- The last steps of subproofs in a “flattened” transitivity step are repetitions of the pattern *premises, right productive rules, subtree replacement rules, left productive rules, premises*.
- The subproofs in each normal form are normal forms.

Next, we define the notion of a normal form that is free from sequences of mixed monotonicity.

DEFINITION 6.4 (mixed monotonicity)

Let $f_1(x_1), \dots, f_n(x_n)$ be derivation trees, where $f(x)$ is a derivation

$$\frac{f_{A|B} \quad x_B}{A}$$

We say that a part of normal form:

$$f_1(x_1) \leq f_2(x_2) \leq \dots \leq f_n(x_n)$$

has mixed monotonicity iff one the following holds:

- $f_1, f_n \in \mathbf{MON} \uparrow \cup \mathbf{MON} \downarrow$ and there exist $i, j, k: 1 \leq i \leq j \leq k \leq n$, s.t. $f_i, f_k \in \mathbf{MON} \uparrow, f_j \in \mathbf{MON} \downarrow$ or $f_i, f_k \in \mathbf{MON} \downarrow, f_j \in \mathbf{MON} \uparrow$.
- $f_1 \notin \mathbf{MON} \uparrow \cup \mathbf{MON} \downarrow$ or $f_n \notin \mathbf{MON} \uparrow \cup \mathbf{MON} \downarrow$ and there exist $i, j: 1 \leq i \leq j \leq n$ s.t. $f_i \in \mathbf{MON} \uparrow, f_j \in \mathbf{MON} \downarrow$ or $f_i \in \mathbf{MON} \downarrow, f_j \in \mathbf{MON} \uparrow$.
- $f_1, f_n \notin \mathbf{MON} \uparrow \cup \mathbf{MON} \downarrow$ and there exists $i: 1 < i < n$, s.t. $f_i \in \mathbf{MON} \uparrow \cup \mathbf{MON} \downarrow$.

In the appendix we show that under a certain (syntactically plausible) assumptions on the usage of coordination categories and restrictive modification, the algorithm considers every proof that is PWC-free, provided its normal form is free from sequences of mixed monotonicity. At present we are not aware of any linguistic implication of this assumption, but this is an open question.

In the rest of this section, we discuss three topics. First, we mention the syntactic assumptions that are needed on coordination and restrictive modification. Then we show that elimination of PWC is not innocuous, and it imposes a restriction on the effectiveness of the proposed algorithm. Whether or not there is a correct decision procedure for provability in the Order Calculus including (non-trivial usages of) PWC is unknown to us.

Assumption on coordination We have already assumed above that a derivation tree α of category $A|A$ for a “half-coordinated” expression *coord* X combines only with derivation trees of category A (and not of category

$B|(A|A)$). This is a reflection of the fact that coordination is an n -ary syntactic function (with $n \geq 2$), and hence Curryng as in the categorial fragment above is not a satisfying syntactic treatment of this phenomenon. In addition, we now assume that no premise is an order relation $\alpha \leq \beta$ or $\beta \leq \alpha$, for such a derivation tree α of a “half coordination”. Given this, the appendix (lemma A.2) we prove that each proof using FR of an order relation that involves a coordination $\alpha \text{ } \textit{coord}^0 \text{ } \beta$ on its left side is of the following form (\textit{coord}^0 is some coordinator):

$$\overline{\overline{\frac{\frac{\textit{coord}^0_{c \setminus c / c} \beta_c}{\alpha_c} \quad /E}{c} \quad \setminus E}} \leq \overline{\overline{\frac{\textit{coord}^1_{c \setminus c / c} \beta'_c}{\alpha'_c} \quad /E}{c} \quad \setminus E}} \quad \text{FR}$$

A similar fact holds when the coordination appears on the right side of the order relation. Assume now that in the process of mapping a proof to a normal form we get a structure of the following form:

$$(13) \quad \frac{\overline{\overline{\dots}} \quad D1 \quad \overline{\overline{\dots}} \quad FR \quad \dots}{\alpha \leq \alpha \vee \beta \quad \alpha \vee \beta \leq \alpha' \vee \beta' \quad \dots} \quad \gamma \leq \delta$$

In the FR step it must be the case that either $\alpha \leq \alpha'$ or $\beta \leq \beta'$. Therefore, we can replace (13) by the following, smaller, form:

$$\frac{\overline{\overline{\dots}} \quad \overline{\overline{\dots}} \quad D1 \quad \dots}{\alpha \leq \alpha' \quad \alpha' \leq \alpha' \vee \beta' \quad \dots} \quad \gamma \leq \delta$$

This reduction of the size of the proof allows us to rely on an induction assumption that each proof of a smaller size than the original proof has a normal form.

A simpler assumption on coordinator categories is that they are only lexical. Thus, we assume that the C/D marking for conjunction and disjunction appears only in categories of the form $(X|Y)^{C/D}Z$. This assumption is linguistically plausible: we do not know any need for a category $X|Y$, where X is a coordinator category.

Assumption on restrictive modification In order for a normal form to exist, we assume that no restrictive modifier is created from application of a lexical category to other categories. That is, no lexical category is of the form $((\dots|^\mathbb{R}\dots)|A)$. Suppose for instance that a derivation tree $f(x)$ has a category that is marked by R . Then the following is a proof of $g(x) \leq id$, but it is easy to see that there is no normal proof for this order relation.

$$\frac{\frac{g \leq f}{g(x) \leq f(x)} \quad \text{FR} \quad \frac{\emptyset}{f(x) \leq id} \quad \text{RMOD}_1}{g(x) \leq id} \quad \text{TRANS}$$

Proofs that are not PWC-free Unlike the restriction on mixed monotonicity, the requirement that the proof does not involve a (non-trivial) usage

of the PWC rule is of course a limitation of the algorithm. One example for a proof that the algorithm cannot find is the proof of the inference (12), described in section 5.4. A simpler example, which illustrates better the problem of the proposed algorithm to cope with non-trivial usages of PWC, is the following proof (in simplified format).

SUB1				
some person	\leq	$\text{some teacher or some student}$	PREM	
$\text{some person smiled}$	\leq	$\text{some teacher or some student smiled}$	FR	$\text{some teacher or some student smiled} \leq \text{some teacher smiled or some student smiled}$
$\text{some person smiled} \leq \text{some teacher smiled or some student smiled}$			TRANS	
SUB2				
$\text{some teacher smiled}$	\leq	some girls walked	PREM	
$\text{some teacher smiled or some student smiled} \leq \text{some girls walked}$			D2	$\text{some student smiled} \leq \text{some girls walked}$
$\text{some person smiled}$	\leq	$\text{some teacher smiled or some student smiled}$	SUB1	$\text{some teacher smiled or some student smiled} \leq \text{some girls walked}$
$\text{some person smiled} \leq \text{some girls walked}$			TRANS	

The problem for the algorithm to find the proof is because in the attempt to derive sentences that are 'less or equal' the sentence *some girls walked*, not attempt is made to use the disjunction *some teacher smiled or some student smiled*. Further, the algorithm does not attempt to derive *some teacher or some student smiled* from *some person smiled*.

7 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 helpful to attempt directions that use as many insights as possible from logic, computer science and linguistics. In this paper we tried to show that an attempt to use natural language syntax together with principles from higher order model-theoretic semantics may be worthwhile. Much work is left to be done on extending the system, exploring possible completeness results and improving the decision procedure for provability, which is presently of (at least) exponential complexity, and has some non-trivial restrictions on the input grammar and derivable order relations. On the other hand, the semantic and syntactic flexibility of the proposed inference system, together with its conceptual simplicity, suggest that such an enterprise could be highly rewarding.

A Correctness proof

Assumptions on the Lexicon and Premises

1. The only categories in the lexicon that involve the C/D marking (conjunction/disjunction) are of the form $(A|\epsilon A')|^{C/D} A''$ (= the 'coordinators' in the lexicon). The category of the coordinator $A|B|^{C/D} C$ is basic category in the lexicon, that is it cannot be derived by the Categorical Grammar in any way, but obtained from the lexicon, i.e. no category ends with the coordinator category, but the coordinator itself.
2. No restrictive modifier is created from application of a lexical category to other categories. That is, no lexical category is of the form $((\dots|^{R}\dots)|A)$.
3. A derivation tree α of category $A|A'$ for a "half-coordinated" expression $coor X$ combines only with derivation trees of category A (and not of category $B|(A|A)$).
4. No premise is an order relation $\alpha \leq \beta$ or $\beta \leq \alpha$, for such a derivation tree α .
5. There is no way to derive a constant zero function, i.e a function that sends every argument to the bottom element when it exists. ⁷
6. The lexicon includes items of categories $A|A|^C A$ and $A|A|^D A$ for each PO category A .

LEMMA A.1 (Soundness)

If a call to **derive**(T, T', \emptyset) returns **true** then there is a proof of the order relation $T \leq T'$.

The proof in the Order Calculus can be reconstructed from the call tree of the algorithm in a straightforward way. The algorithm can be defined so that it returns a proof tree in the Order Calculus instead of the mere **true**. We do not include this part of the correctness proof here, since it is rather straightforward.

LEMMA A.2 (Function replacement with coordination)

Let P be a proof of the order relation $T \leq T'$ in the Order Calculus, s.t. the main rule of P is FR. T is a coordination $\alpha \ coor^0 \ \beta$ iff T' is a coordination $\gamma \ coor^0 \ \delta$. In other words: the proof can only be of the following form:

$$\frac{\dots}{\frac{\frac{\frac{coor^0_{c \setminus c / c} \ \beta_c}{\alpha_c} \ / E}{c \setminus c} \ / E}{c} \ \backslash E} \leq \frac{\dots}{\frac{\frac{\frac{coor^1_{c \setminus c / c} \ \delta_c}{\gamma_c} \ / E}{c \setminus c} \ / E}{c} \ \backslash E} \quad \text{FR}$$

Proof We will prove only one direction. The proof of the other direction is similar. Let $T = \alpha_c \ (coor^0_{c \setminus c / c} \ \beta_c)$. The proof of P is of the form

$$\frac{\dots}{\frac{\frac{\frac{coor^0_{c \setminus c / c} \ \beta_c}{\alpha_c} \ / E}{c \setminus c} \ / E}{c} \ \backslash E} \stackrel{R}{\leq} \epsilon_c \setminus c \quad \alpha_c \equiv \gamma_c \quad \text{FR}$$

$$\frac{\dots}{\frac{\frac{\frac{coor^0_{c \setminus c / c} \ \beta_c}{\alpha_c} \ / E}{c \setminus c} \ / E}{c} \ \backslash E} \leq \frac{\frac{\frac{\gamma_c \ \epsilon_c \setminus c}{c} \ \backslash E}{\gamma_c} \ / E}{c} \ \backslash E$$

We will show now, that the rule R can only be FR and therefore $\epsilon = coor^1 \ \delta$.

⁷We need this assumption in the one place only in the proof of the completeness of the algorithm. However we could have can done without it by adding a correction to the algorithm which we avoided for the sake of simplicity

- R is not PREM because no premise is an order relation of the form $coord^0 \beta \leq \epsilon$ by Assumption 4.
- R is not PWC because $coord^0 \beta$ is not a coordination.
- R is not C1 or D2 for the same reason.
- R is not D1 or C2 because that would require $\epsilon = \eta coord^1 (coord^0 \beta)$, which violates Assumption 3.
- R is not RMOD because $coord^0 \beta$ has category c/c where the slash is unmarked by Assumption 1. (In the proof of the second direction the RMOD is impossible because of Assumption 3).
- R is not MON because $coord^0 \beta$ has category c/c where the slash is unmarked by Assumption 1.
- If R is TRANS then the proof of $(coord^0 \beta) \leq \epsilon$ is of the form $(coord^0 \beta) \leq \epsilon' \leq \epsilon$. We can apply a similar reasoning to show that $\epsilon' = coord^2 \delta'$ and $\epsilon = coord^1 \delta$.

The proof of $T \leq T'$ is therefore:

$$\begin{array}{c}
\text{coord}^0 \leq \text{coord}^1 \beta_c \equiv \delta_c \\
\frac{\frac{\text{coord}^0_{c \setminus c/c} \beta_c}{c \setminus c} / E}{c} \leq \frac{\frac{\text{coord}^1_{c \setminus c/c} \delta_c}{c \setminus c} / E}{c} \quad \text{FR} \\
\alpha_c \equiv \gamma_c \\
\frac{\frac{\frac{\text{coord}^0_{c \setminus c/c} \beta_c}{c \setminus c} / E}{\alpha_c} \setminus E}{c} \leq \frac{\frac{\frac{\text{coord}^1_{c \setminus c/c} \delta_c}{c \setminus c} / E}{\gamma_c} \setminus E}{c} \quad \text{FR}
\end{array}$$

LEMMA A.3 (Existence of the Normal Form)

If an order statement $T \leq T'$ has a proof in the Order Calculus, which involves no non-trivial usages of the PWC rule then, given the assumptions 1-6 above, there exists a normal form of the proof of $T \leq T'$.

Proof Let $T \leq T'$ be a proof in the Order Calculus, where there are no non-trivial usages of PWC. As we mentioned above (Fact 6.2), all the trivial cases of the PWC can be rewritten, so we can assume that the proof does not make any use of the PWC rule. We will derive the normal form of the proof in two stages. First let us define the pre-normal form of the proof as the normal form of the proof without the requirement (*) that the rules $R_1 R_2 \dots R_n$ belong to the regular language:

(PREM*, R-PROD*, STR*, L-PROD*) (PREM+, R-PROD*, STR*, L-PROD*)* PREM*.

In words: we only require that all occurrences of the transitivity in the proof are “flattened”.

It is straightforward to show that every proof in the Order Calculus can be rewritten in a pre-normal form. We now prove that a normal form can be built from the pre-normal form.

If the pre-normal form satisfies the requirement (*) it is in the normal form. Otherwise, the only possible violation of (*) is a violation of the relative ordering of the R-PROD, STR and L-PROD: an L-PROD rule precedes an R-PROD or an STR rule, or an STR rule precedes an R-PROD rule. We show that, all these violation can be removed, thus creating a proof in the normal form. We classify three kinds of violations of (*):

- impossible violations - combinations of the rules of the calculus that cannot occur in the proof;
- removable violations - combinations of the rules of the calculus that can be removed from the proof, producing a smaller proof;
- re-writable violations - combinations of the rules of the calculus that can be rewritten, producing a tree with a smaller number of violations.

Recall the classification of rules according to their productivity:

1. *Right Productive* (R-PROD) rules are PWC, RMOD₁, RMOD₂, C1, and D2.
2. *Left Productive* (L-PROD) rules are PWC, C2 and D1.
3. *Subtree Replacement* (STR) rules are MON and FR.

Impossible violations These violation are impossible due to the assumptions on the lexicon.

C2,D2(L-PROD,R-PROD) No coordinator in the Lexicon is marked with both C and D , because there are no multiple features.

D1,C1(L-PROD,R-PROD) Similar to the previous case.

D1,RMOD₂(L-PROD,R-PROD) There is no category $(a|^{Rb})|^{Dc}$ because of Assumption 1.

C2,RMOD₂(L-PROD,R-PROD) There is no category $(a|^{Rb})|^{Cc}$ because of Assumption 1.

D1,RMOD₁(L-PROD,R-PROD) Because of Assumptions 3 and 4, the derivation $(\text{coord}X)_{A|A}$ cannot be proved to be less or equal than $id_{A|A}$. It cannot be compared to any derivation other than $(\text{coord}Y)_{A|A}$ and no category $A|A$ of a derivation tree $(\text{coord}X)_{A|A}$ is marked for restrictiveness.

C2,RMOD₁(L-PROD,R-PROD) Similar to the previous case.

MON,RMOD₁(STR,R-PROD) Impossible because of Assumption 2.

FR,RMOD₁(STR,R-PROD) Impossible because of Assumption 2.

MON,C1(STR,R-PROD) The arguments of the coordination are not marked for monotonicity, because of Assumption 1.

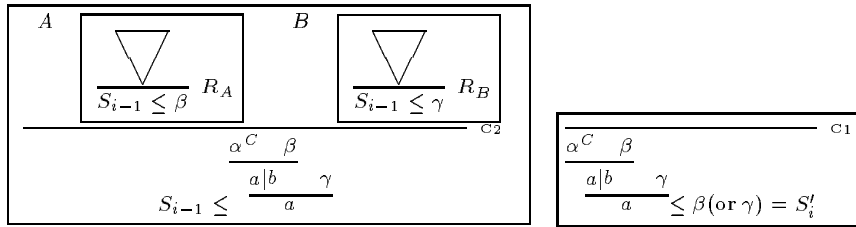
D1,FR(L-PROD,STR) In a similar way.

MON,D2(STR,R-PROD) In a similar way.

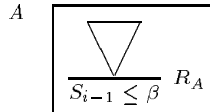
C2,FR(L-PROD,STR) In a similar way.

Removable violations These violations can be removed by a local transformation on a pre-normal form, thus producing a pre-normal form of a smaller size. By consistent application of these transformations all violations of this kind can be removed.

C2,C1(L-PROD,R-PROD) Let the violation be

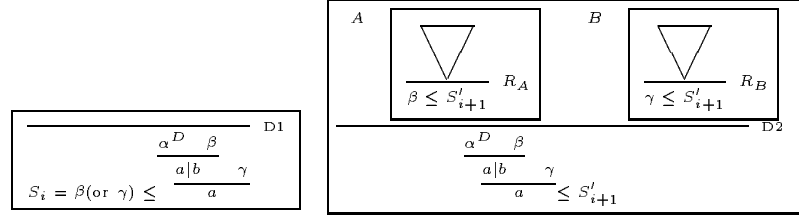


This part of the proof can be replaced by the following proof of $S_{i-1} \leq S_i^l$

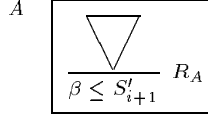


where $\beta = S_i^l$.

D1,D2(L-PROD,R-PROD) Let the violation be

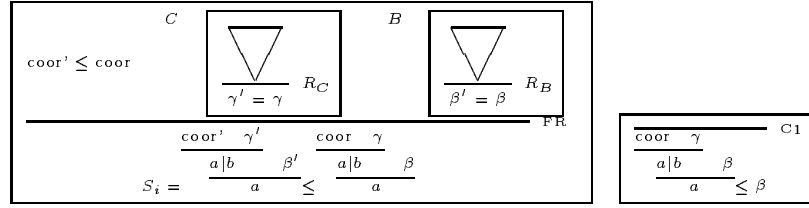


This part of the proof can be replaced by the following proof of $S_i \leq S'_{i+1}$

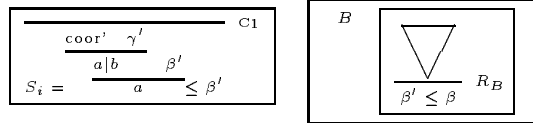


where $S_i = \beta$.

FR,C1(STR,R-PROD) Let the violation be



where $\beta = S'_{i+1}$. This is the only form according to Lemma A.2. This part of the proof can be replaced by the following proof of $S_i \leq S'_{i+1}$



where $\beta = S'_{i+1}$

D1,FR(L-PROD,STR) The proof is symmetric

Rewritable violations These are $(STR, D2)$, $(C2, STR)$ and $(STR, RMOD_1)$, $(STR, RMOD_2)$.

We have proven above that every removable violation can be removed from the proof, creating a smaller proof. Now we will prove that the proof of $T \leq T'$ that has only rewritable violations has a normal form. The proof is by induction on the size n of the pre-normal form of a proof, where the size of the proof is, by standard definition, the number of nodes in the proof.

Base $n = 1$. The proof is a single axiom or a premise. It is already in the normal form.

Hypothesis For all proofs in the pre-normal form of size less or equal n there exists a normal form of the proof.

Step Let $T \leq T'$ be a pre-normal form of a proof of size $n + 1$.

If it is in a pre-normal form of *type 1* then each of the subproofs $\alpha_1, \dots, \alpha_k$ is of size at most n , and by the induction hypothesis, it has a normal form. Then the proof itself

has a normal form by definition of the normal form.
 If it in the pre-normal form of type 2:

$$\begin{array}{c}
 R_1 \quad \boxed{\frac{\nabla}{S_1 \leq S'_1} \text{ alpha}_1} \quad R_2 \quad \boxed{\frac{\nabla}{S_2 \leq S'_2} \text{ alpha}_2} \quad \dots \quad R_k \quad \boxed{\frac{\nabla}{S_k \leq S'_k} \text{ alpha}_k} \\
 \hline
 T = S_1 \leq S'_k = T' \quad \text{TRANS}^*
 \end{array}$$

and one of the main rules R_i in α_i is PREM, then by the induction hypothesis each of the proofs $S_1 \leq S_i$ and $S_{i+1} \leq S_k$ has a normal form and by definition of the normal form the proof $S_1 \leq S_k$ also has a normal form.

Assume there are no uses of PREM rule. By the induction hypothesis each of the proofs $\alpha_1, \dots, \alpha_k$ has a normal form. If the string formed by the rules $R_1 R_2 \dots R_k$ belongs to the regular language:

(R-PROD*, STR*, L-PROD*) we are done. Otherwise there is a violation of the order of the rules. We will show that this violation can be removed, by showing how each kind of violation is rewritten:

FR,D2(STR,R-PROD) The proof in the pre-normal form (at the place of the violation) is

$$\boxed{
 \begin{array}{c}
 \frac{\text{coord}' \leq \text{coord} \quad \gamma' \equiv \gamma \quad \beta' \equiv \beta}{\text{coord}' \quad \gamma'} \text{FR} \\
 \frac{\frac{a|b}{a} \quad \beta'}{\frac{a|b}{a} \quad \beta} \\
 S_i = \frac{\quad}{a} \leq \frac{\quad}{a}
 \end{array}
 }
 \quad
 \boxed{
 \begin{array}{c}
 \frac{\gamma \leq S'_{i+1} \quad \beta \leq S'_{i+1}}{\text{coord} \quad \gamma} \text{D2} \\
 \frac{\frac{a|b}{a} \quad \beta}{\frac{a|b}{a} \quad \beta} \\
 \frac{\quad}{a} \leq S'_{i+1}
 \end{array}
 }$$

This is the only form because of Lemma A.2. There exists the following proof in a pre-normal form:

$$\boxed{
 \begin{array}{c}
 \frac{\gamma' \leq \gamma \leq S'_{i+1} \quad \beta' \leq \beta \leq S'_{i+1}}{\text{coord}' \quad \gamma'} \text{D2} \\
 \frac{\frac{a|b}{a} \quad \beta'}{\frac{a|b}{a} \quad \beta} \\
 S_i = \frac{\quad}{a} \leq S'_{i+1}
 \end{array}
 }$$

By the induction hypothesis, the proofs $\gamma' \leq S'_{i+1}$ and $\beta' \leq S'_{i+1}$ have a normal form.

C2,FR(L-PROD,STR) The proof is similar to the one above.

MON,RMOD₂(STR,R-PROD) The proof in the pre-normal form is

$$\boxed{
 \begin{array}{c}
 B \quad \boxed{\frac{\nabla}{\delta_a \leq \beta_a} R_B} \\
 \hline
 \frac{\alpha_{a|a} \quad \delta_a \quad \alpha_{a|a} \quad \beta_a}{\alpha_{a|a} \quad \beta_a} \text{MON} \\
 S_{i-1} = \frac{\quad}{a} \leq \frac{\quad}{a}
 \end{array}
 }
 \quad
 \boxed{
 \begin{array}{c}
 C \quad \boxed{\frac{\nabla}{\alpha_{a|a} \leq id_{a|a}} R_C} \\
 \hline
 \frac{\alpha_{a|a} \quad \beta_a}{\alpha_{a|a} \quad \beta_a} \text{RMOD}_2 \\
 \frac{\quad}{a} \leq \beta_a = S'_i
 \end{array}
 }$$

This is the only form because α , which is a restrictive function, cannot have a downward monotonic behavior: it is provable that every restrictive downward monotonic function is a constant function sending everything to a *bottom* element. According to

Assumption 5, no zero function is derivable from our Lexicon. Therefore, there exists the following proof in a pre-normal form

$$S_{i-1} = \frac{\frac{\frac{\alpha_a | a \leq id_{a|a}}{R_C}}{R_{MOD_2}}}{\frac{\alpha_a | a \leq \delta_a}{a} \leq \delta_a} \quad B \quad \frac{\frac{\frac{\delta_a \leq \beta_a}{R_B}}{R_B}}{R_B} = S'_i$$

By the induction hypothesis the subproofs B and C have normal form. **FR, RMOD₂(STR, R-PROD)** The proof in the pre-normal form is

$$S_{i-1} = \frac{\frac{\frac{\frac{\frac{\gamma_a | a \leq \alpha_a | a}{R_A}}{R_A}}{R_{PR}} \quad \frac{\frac{\delta_a \equiv \beta_a}{R_B}}{R_B}}{\frac{\gamma_a | a \leq \delta_a \quad \alpha_a | a \leq \beta_a}{a} \leq \frac{\alpha_a | a \leq \beta_a}{a}}}{\frac{\alpha_a | a \leq \beta_a}{a} \leq \beta_a} \quad R_{MOD_2}$$

where $\beta_a = S'_i$

There exists the following proof in a pre-normal form

$$S_{i-1} = \frac{\frac{\frac{\frac{\frac{\gamma_a | a \leq \alpha_a | a}{R_A}}{R_A}}{R_{MOD_2}} \quad \frac{\frac{\alpha_a | a \leq id_{a|a}}{R_C}}{R_C}}{\frac{\gamma_a | a \leq id_{a|a}}{R_{MOD_2}} \quad \frac{\alpha_a | a \leq id_{a|a}}{R_C}}}{\frac{\gamma_a | a \leq \delta_a}{a} \leq \delta_a} \quad R_{MOD_2} \quad B \quad \frac{\frac{\frac{\delta_a \leq \beta_a}{R_B}}{R_B}}{R_B}$$

where $\beta_a = S'_i$

(the proof of $\delta_a \equiv \beta_a$ is actually a proof of $\delta_a \leq \beta_a$ and $\delta_a \geq \beta_a$). By the induction hypothesis the subproofs A, B and C and the proof of $\gamma \leq id_{a|a}$ have normal form.

In this way we can remove all the rewritable violations, thus producing a proof in the normal form.

LEMMA A.4 (The provability of the Normal Form)

Let $T \leq T'$ be an order relation that has a proof in a normal form, s.t. every sequence of STR rules of the form:

$$f_1(x_1) \leq f_2(x_2) \leq \dots \leq f_n(x_n)$$

$$\frac{f_A | B \quad x_B}{A}$$

, where $f(x)$ is a derivation $\frac{f_A | B \quad x_B}{A}$, is free from sequences of mixed monotonicity (as defined by Definition 6.4), then **derive**(T, T', \emptyset) returns **true**.

Proof The proof is by induction on the size of the normal form n .

Base $n = 1$. The proof has the form $\frac{\emptyset}{T_0} \leq T \quad R$, where R is one of the REFL, PREM, RMOD₁, PWC, C1, D1.

- REFL. $T_0 = T$. The algorithm finds the proof by **subderive** Step 1

- PREM. $\langle T_0, T \rangle$ is a premise. The algorithm detects this proof by **derive** Step 4 and then twice by **subderive** Step 1.
- RMOD₁. $T_0 = \alpha_{A'|R_A}$ and $T = id_{A'|A}$. The algorithm detects the proof by **subderive** Step 2 for $f_{RMOD_1,r}$ and **subderive** Step 1 for $id_{A'|A} \leq id_{A'|A}$
- C1. $T = \alpha$ and β and $T = \alpha$ or $T = \beta$. The algorithm detects the proof by **subderive** Step 2.1 for $f_{C1,r}$ and **subderive** Step 1 for $T' \leq T'$.
- D1. $T = \alpha$ or $T = \beta$ and $T' = \alpha$ or β . The algorithm detects the proof by **subderive** Step 2.2 for $f_{D1,l}$ and **subderive** Step 1 for $T \leq T$.
- PWC. The algorithm detects the proof by **subderive** Step 2.1 for $f_{PWC,r}$ and **subderive** Step 1 for $T' \leq T'$.

Hypothesis Let us assume that algorithm detects a proof for every $T_0 \leq T$, that has a proof with a normal form of size n or less.

Step Let $T_0 \leq T$ be an order relation that has a proof with a normal form of size $n + 1$.

1. The proof is of a normal form of *type 2*:

$$\frac{\alpha_1 \quad \boxed{\frac{\nabla}{S_1 \leq S'_1} R_1} \quad \alpha_2 \quad \boxed{\frac{\nabla}{S_2 \leq S'_2} R_2} \quad \dots \quad \alpha_l \quad \boxed{\frac{\nabla}{S_l \leq S'_l} R_l}}{T = S_1 \leq S'_l = T'} \text{TRANS}^*$$

where $l \geq 2$ and $\alpha_1, \dots, \alpha_l$ are proofs in normal form of *type 1*, $S'_1 = S_2, S'_2 = S_3, \dots, S'_{l-1} = S_l$ and the string formed by the rules $R_1 R_2 \dots R_l$ belongs to the regular language:

- (PREM*, R-PROD*, STR*, L-PROD*) (PREM+, R-PROD*, STR*, L-PROD*)* PREM*.
- (a) Assume $i \ 1 \leq i_1 \leq i_2 \leq \dots \leq i_m \leq l$, s.t. for every $j \ 1 \leq j \leq m : R_{i_j} = \text{PREM}$, that is $S_{i_j} \leq S'_{i_j}$ is a premise. Let us look at the leftmost use of the PREM rule, i.e. $S_{i_1} \leq S'_{i_1}$. In **derive** Step 4 the algorithm will try the premise $\langle S_{i_1}, S'_{i_1} \rangle$ and will call **subderive**($T, S_{i_1}, Goals'$) and **derive**($S'_{i_1}, T', Goals'$). The order relation $T \leq S_{i_1}$ has a proof in the normal form:

$$\frac{\alpha_1 \ \alpha_2 \ \dots \ \alpha_{i_1-1}}{T \leq S_{i_1}} \text{TRANS}^*$$

where none of the $R_1 \dots R_{i_1-1}$ is PREM, therefore when attempting to prove the order statement $T \leq S_{i_1}$ **derive** calls **subderive** immediately. The order relation $S_{i_1+1} \leq T'$ has a proof in the normal form:

$$\frac{\alpha_{i_1+1} \ \alpha_{i_1+2} \ \dots \ \alpha_l}{S_{i_1+1} \leq S_l = T'} \text{TRANS}^*$$

Both of these proofs are of a size at most n and by the induction hypothesis the algorithm detects these proofs.

- (b) There is no i , $1 \leq i \leq l$, s.t. $R_i = \text{PREM}$. The string formed by the rules $R_1 R_2 \dots R_l$ belongs to the regular language R-PROD*STR*L-PROD*.
 - i. R_1 is an R-PROD rule, that is the algorithm can effectively produce subgoals of a smaller size, which are provable by induction hypothesis.
 1. R_1 is an axiom, that is $S'_1 = f_{a,r}(S_1)$. In the **subderive** Step 2 the algorithm will try to prove the order relation $S'_1 \leq T$, which has a proof in the normal form

$$\frac{\alpha_2 \ \dots \ \alpha_l}{S'_1 \leq S_l = T'} \text{TRANS}$$

of size at most n . By the induction hypothesis the algorithm will detect the proof.
2. R_1 is a rule, i.e one of the rules D2 and RMOD₂.

(i) If it is D2 we have

$$\alpha_1 = \frac{S_{1,1} \leq S'_1 \quad S_{1,2} \leq S'_1}{S_{1,1} \text{ or } S_{1,2} \leq S'_1} \text{D2}$$

The algorithm will try to prove in **derive** Step 3.1 $S_{1,1} \leq T$ and $S_{1,2} \leq T$. Both of these order relations have proofs in the normal form

$$\frac{S_{1,i} \leq S'_1 \quad \alpha_2 \quad \dots \quad \alpha_l}{S_{1,i} \leq T} \text{TRANS}$$

where $i \in \{1, 2\}$, of the size at most n and by the induction hypothesis the algorithm will detect these proofs.

(iii) It is RMOD₂. That is $T = \frac{\alpha_{A'|A} \quad \beta_A}{A'} \text{I}^E$. The algorithm will try to prove in **subderive** Step 5 that $\alpha_{A'|A} \leq id_{A'|A}$ and $\beta_A \leq T'$. Both of these proofs are of a smaller size and by the induction hypothesis the algorithm will return **true** for these order statements.

- ii. R_l is an L-PROD rule. The proof is symmetric.
iii. If none of the above is true then the string formed by the rules $R_1 R_2 \dots R_l$ belongs to the regular language STR*. The proof has the form

$$f_1(x_1) \leq f_2(x_2) \leq \dots \leq f_l(x_l)$$

and one of the following claims hold:

- Both f_1 and f_l are marked for monotonicity and there does not exist i, j, k $1 \leq i < j < k \leq n$, s.t. $f_i, f_k \in \text{MON } \uparrow, f_j \in \text{MON } \downarrow$ or $f_i, f_k \in \text{MON } \downarrow, f_j \in \text{MON } \uparrow$. There are two possible cases:
 - (i) f_1, f_l are marked similarly. Let $f_1, f_l \in \text{MON } \uparrow$. Then all the functions f_1, \dots, f_l are MON \uparrow or unmarked. Therefore all the rules in the sequence are FR or MON for MON \uparrow case, which means that $f_1 \leq \dots \leq f_l$ and $x_1 \leq \dots \leq x_l$ (in the case of FR rule $x_i \equiv x'_i$ and hence also $x_i \leq x'_i$). The algorithm in **subderive** Step 4.1 will try to prove $f_1 \leq f_l$ and $x_1 \leq x_l$ and, since both of these order relations have a normal form proof of a smaller size, by the induction hypothesis the algorithm will detect the proofs. The case of the downward monotonicity is symmetric.
 - (ii) $f_1 \in \text{MON } \uparrow, f_l \in \text{MON } \downarrow$ (or vice versa). Then the functions are marked in the following way: from f_1 up to some f_i all the functions are MON \uparrow or unmarked and from f_{i+1} up to f_l all the functions are MON \downarrow or unmarked. The algorithm at **subderive** Step 4.3 tries to prove $f_1 \leq f_l$ and succeeds, since it has a proof in the normal form of a smaller size and then will generate the following form:

$$\frac{\frac{\overline{x_1 \geq lub} \text{D1}}{f_1(x_1) \leq f_1(lub)} \text{MON} \quad \frac{\dots}{f_1(lub) \leq f_l(lub)} \text{MON} \quad \frac{\overline{lub \leq x_l} \text{D1}}{f_l(lub) \leq f_l(x_l)} \text{MON}}{f_1(x_1) \leq f_l(x_l)} \text{TRANS}$$

where $lub = (x_1 \text{ or } x_l)$

This proof exists because of the Assumption 6 that guarantees existence of *or*.

⁸ The case of $f_1 \in \text{MON } \downarrow, f_l \in \text{MON } \uparrow$ is similar but uses $glb = x_1$ and x_l instead of the $lub = x_1$ or x_l .

⁸Since the type of numbers (of category *num*) is not a PO type, the algorithm will not detect a proof if the category of x_1, \dots, x_l is *num*.

- One of the categories for f_1 or f_l is not marked for monotonicity and there do not exist $i, j, 1 \leq i \leq j \leq n$, s.t. $f_i \in \text{MON } \uparrow, f_j \in \text{MON } \downarrow$ or $f_i \in \text{MON } \downarrow, f_j \in \text{MON } \uparrow$. Therefore, all the categories for f_1, \dots, f_l are $\text{MON } \downarrow$ or unmarked for monotonicity and one of f_1 and f_l is marked for monotonicity therefore $f_1 \leq \dots \leq f_l$ and $x_1 \leq \dots \leq x_l$ (or $x_1 \geq \dots \geq x_l$ if the marked function in $\text{MON } \downarrow$). The algorithm in **subderive** Step 4.1 will try to prove $f_1 \leq f_l$ and $x_1 \leq x_l$ (or $x_1 \geq x_l$) and succeed since both of them have a proof in the normal form of a smaller size. Then the following proof will be generated (if $f_1 \in \text{MON } \uparrow$):

$$\frac{\frac{\dots}{x_1 \leq x_l} \text{ MON } \frac{\dots}{f_1 \leq f_l} \text{ MON}}{f_1(x_1) \leq f_1(x_l)} \text{ MON } \frac{f_1(x_l) \leq f_l(x_l)}{f_1(x_1) \leq f_l(x_l)} \text{ TRANS}$$

The symmetric cases are similar.

- Both of the categories for f_1 and f_l are unmarked and there does not exist $i, 1 \leq i \leq n$, s.t. f_i is marked. This means that all the functions f_1, \dots, f_l are not marked, that is all the rules in the sequence are FR. Therefore $f_1 \leq \dots \leq f_l$ and $x_1 \equiv \dots \equiv x_l$. The algorithm in **subderive** Step 4.4 will try to prove $f_1 \leq f_l$ and $x_1 \equiv x_l$ and, since both of these order relations have a proof in the normal form of a smaller size, by the induction hypothesis the algorithm will find the proofs.

2. The proof is of the normal form of the *type 1*:

$$\frac{\alpha_1 \boxed{\begin{array}{c} \nabla \\ S_1 \leq S'_1 \end{array}} \quad \alpha_2 \boxed{\begin{array}{c} \nabla \\ S_2 \leq S'_2 \end{array}} \quad \dots \quad \alpha_l \boxed{\begin{array}{c} \nabla \\ S_l \leq S'_l \end{array}}}{T \leq T'} \text{ R}$$

where $R \neq \text{TRANS}$, $n \geq 0$ and $\alpha_1, \dots, \alpha_l$ are normal forms.

$$(b) \frac{\emptyset}{T_i \leq T'_i} \text{ PREM}$$

where $T_i \leq T'_i$ is a premise.

According to the rule R of the proof, the proof is similar to one of the cases above.

Set of all available subtrees. Let $T \leq T'$ be an order statement that **derive** tries to prove, $P = \{\langle T_i, T'_i \rangle\}$ is the finite set of the premises, CAT_l is the set of all the categories derivable from the lexicon. Let $P' = \{T_i | \exists \langle T_i, T'_i \rangle \in \text{Prem} \text{ or } \exists \langle T'_i, T_i \rangle \in \text{Prem}\}$, $\text{Trees} = \{T, T'\} \cup P' \cup \{id_c | \forall c \in \text{CAT}_l, \text{ s.t. } c = a|a\}$. Now let Trees' be the set of trees that can be obtained from Trees by the following transformation: for every $T \in \text{Trees}$ that has a coordination in it add all the trees that can be generated from T by a number of applications of 'coordination reduction'. Provably, the set Trees' is finite. The set $ST = \{T | \exists T' \in \text{Trees}', \text{ s.t. } T \text{ is a subtree } T'\}$ is the set of all available subtrees. The set ST is finite due to the finiteness of Trees' . The details of this process are simple but rather cumbersome, and we therefore leave it undefined here.

LEMMA A.5 (Available pairs)

Given T, T' and Prem the set of premises, **derive** is called on the trees from the ST only.

Proof By Induction on d - the depth of the call in the call tree.

Base $d = 0$. The first call is **derive**(T, T', \emptyset). $T, T' \in \text{Trees} \subseteq \text{Trees}' \subseteq ST$.

Assumption For depth less or equal d the lemma holds.

Step The depth of the call is $d + 1$.

- **derive** was called in $\text{derive}(T_0, T_1, Goals)$ at Step 4, that is the call is $\text{derive}(T'_i, T_1, Goals')$. $T_1 \in ST$ by the induction hypothesis and $T'_i \in Prem' \subseteq Trees \subseteq ST$.
- **derive** was called in $\text{subderive}(T_0, T_1, Goals)$ at Step 2.1, that is $T = f_{ax,r}(T_0)$ $T' = T_1$. $T_0, T_1 \in ST$ by the induction hypothesis. The function $f_{ax,r}$ is one of:
 1. $f_{\text{RMOD}_{1,r}}$. Then T is a subtree of $T_0 \in ST$ and by definition of ST $T \in ST$ also.
 2. $f_{C_{1,r}}$. Similar.
 3. $f_{PWC,r}$. $T_0 \in ST$, that is $\exists T'' \in Trees'$, s.t. T_0 is its subtree. $T = f_{PWC,r}(T_0)$ that is it is obtained from T_0 by a the movement of the coordinator according to the point-wise coordination principle. T_0 is a subtree of a tree T''' , which can be obtained from T'' by the movement of the coordinator. $T''' \in Trees'$ by the construction of $Trees'$ and $T_0 \in ST$ by the construction of ST .
- **derive** was called in $\text{subderive}(T_0, T_1, Goals)$ at Step 2.2. The proof is symmetric.
- **derive** was called in $\text{subderive}(T_0, T_1, Goals)$ at Step 3.1, that is $T = f_{D_{2,1}}(T_0)$ or $T = f_{D_{2,1}}(T_0)$ and $T' = T_1$. The proof is similar to the RMOD case.
- **derive** was called in $\text{subderive}(T_0, T_1, Goals)$ at Step 3.2, that is $T = f_{C_{2,1}}(T_0)$ or $T = f_{C_{2,1}}(T_0)$ and $T' = T_1$. The proof is similar to the RMOD case.
- **derive** was called in $\text{subderive}(T_0, T_1, Goals)$ at Step 5, that is $T_0 = \frac{\alpha_{A|A'} \beta_{A''}}{A}$. By the induction hypothesis: $T_0, T_1 \in ST$. There are two cases:
 - $T = \alpha$ and $T' = id_{A|A'}$. $T' \in Trees \subseteq ST$, T is a subtree of $T_0 \in ST$ and by definition of ST : $T \in ST$.
 - $T = \beta$ and $T' = T_1$. $T' = T_1 \in ST$, T is a subtree of $T_0 \in ST$ and by definition of ST : $T \in ST$.
- **derive** was called in $\text{subderive}(T_0, T_1, Goals)$ at Step 4, that is $T_0 = \frac{\alpha \beta}{c}$, $T_1 = \frac{\gamma \delta}{c'}$ and $T = \alpha$, $T' = \gamma$ or $T = \beta$, $T' = \delta$. All of the $\alpha, \beta, \gamma, \delta$ are subtrees of trees in ST and by definition of ST they are in ST too.

LEMMA A.6 (Termination)

The algorithm **derive** always terminates.

Proof By Lemma A.5 **derive** is called only on trees from ST . ST is a finite set, hence there are $|ST| \times |ST|$ different pairs $\langle T', T'' \rangle$ and every infinite path in the call tree must include a repeated calls with the same pair. But, the algorithm does not have paths in the call tree that contain a call to **derive** with the same pair $\langle T', T'' \rangle$ more than twice: the second call will always return **false** because of the *Goals* parameter. Then the call tree of the algorithm is finite and it will terminate when the whole call tree is traversed.

Acknowledgments

The parts of the second and third authors were partly 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". Part of the research by the second author was carried out during a stay at the Utrecht University, which was supported by an NWO grant no. B30-541. We are grateful to Johan van Benthem, Raffaella Bernardi, Ed Keenan, Michael Moortgat, Rani Nelken, Ian Pratt-Hartmann and the participants of ICOS-2 for their remarks on this work.

References

- [1] Carlos Areces and Raffaella Bernardi. Polarity in the base logic. Unpublished ms., Utrecht University, 2000.
- [2] R. Bernardi. Monotonic reasoning from a proof-theoretical perspective. In *Proceedings of Formal Grammar*, 1999.

- [3] B. Carpenter. *Type-Logical Semantics*. MIT Press, Cambridge, Massachusetts, 1997.
- [4] E. Keenan and L. Faltz. *Boolean Semantics for Natural Language*. D. Reidel, Dordrecht, 1985.
- [5] D. A. McAllester and R. Givan. Natural language syntax and first-order inference. *Artificial Intelligence*, 56:1–20, 1992.
- [6] W. C. Purdy. A logic for natural language. *Notre Dame Journal of Formal Logic*, 32:409–425, 1991.
- [7] V. Sánchez. *Studies on Natural Logic and Categorical Grammar*. PhD thesis, University of Amsterdam, 1991.
- [8] E. Thijssen. On some proposed universals of natural language. In A. ter Meulen, editor, *Studies in Modeltheoretic Semantics*. Foris, Dordrecht, 1983.
- [9] Johan van Benthem. Meaning: interpretation and inference. *Synthese*, 73:451–470, 1987.

Received 15 January 2001.