QUANTIFIER SCOPING IN THE SRI CORE LANGUAGE ENGINE

Douglas B. Moran
Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025, USA

ABSTRACT

An algorithm for generating the possible quantifier scopings for a sentence, in order of preference, is outlined. The scoping assigned to a quantifier is determined by its interactions with other quantifiers, modals, negation, and certain syntactic-constituent boundaries. When a potential scoping is logically equivalent to another, the less preferred one is discarded.

The relative scoping preferences of the individual quantifiers are not embedded in the algorithm, but are specified by a set of rules. Many of the rules presented here have appeared in the linguistics literature and have been used in various natural language processing systems. However, the coordination of these rules and the resulting coverage represents a significant contribution. Because experimental data on human quantifier-scoping preferences are still fragmentary, we chose to design a system in which the set of preference rules could be easily modified and expanded.

The algorithm described has been implemented in Prolog as part of a larger natural language processing system. Extensions of this algorithm are in progress.

INTRODUCTION

One of the major sources of ambiguity in sentences results from the different scopes that can be assigned to the various quantified noun phrases in the sentence. Part of the problem in determining the preferred scopings of quantifiers is the number of factors involved. For example, consider these three sentences

John visited every house on a street. (1)
John visited every house on a square. (2)
John visited every patient in a private room. (3)

Each of these sentences has two quantifier scopings: in one, "*every*" has wider scope over "*a*," and while in the other, "*a*" has the wider scope. However, the readings that most people obtain for these sentences are quite different. In (1), the reading in which "*a*" has wider scope is highly preferred; in (3), the reading in which "*every*" has wider scope is highly preferred; in (2), the reading with wide-scope "*every*" is preferred, but wide-scope "*a*" is also acceptable. A plausible explanation for the difference between (1) and (2) is that, since the typical house is located on a street but not on a square, the default preference represented by (2) is overridden by a conversational maxim of quantity—if "*a street*" has narrow scope, "*on a street*" would contribute too little information to justify its presence. A plausible explanation for the difference between (2) and (3) is based on the relationship among the components. The reading of (3) in which "*a*" is given wider scope is improbable because the domain of quantification for "*every*" would then be the single patient in the selected room—an infelicitous use of "*every*," whereas there is no similar problem in (2) because there are normally multiple houses on a square. Similarly, in

John visited a person on every committee. (4)
John visited a house on every street. (5)

the reading in which "*a*" has wider scope is reasonable for (4) but not for (5)—in a normal domain of discourse, it is conceivable that there could be a person who is on all of the committees, but it is highly improbable that the geometry of the streets is such that a single house could be located on all of them.

In (1), (3), and (5), discourse criteria and domain information seem to be the primary factors in determining the preferred quantifier scopings, whereas in (2) and (4), linguistic criteria seem to

be the determining factors.

Our approach presumes that the determination of a sentence's preferred scoping can be divided into two phases, the first of which is the subject of the algorithm described here. In this initial phase, linguistic information is used to generate the possible quantifier scopings in order of preference. The relevant linguistic information consists of surface position, syntactic structure, and the relationship among the function words (determiners, modals, and negation). In the second phase (future work), domain and discourse information is applied successively to these scopings, modifying the scores produced by the first phase. We expect that the modifications will be only penalties, thus making it possible to identify the best choice when it is encountered (cutting off the processing of remaining scopings generated by the first phase).

The primary study of quantifier scoping preferences was done by VanLehn (1978). The experimental data reported therein was of limited usefulness in developing the algorithm described here—it was gathered and evaluated under assumptions arising from a different linguistic theory.

We shall first present the rules that governed the structure of our design, then outline the algorithm. This scoping algorithm has been implemented as a component of a larger system that is under continuing development. In this system, called the *Core Language Engine* or *CLE* (Alshawi *et al.*, 1987), the semantic interpretation phase produces unscoped logical forms in which quantifier expressions are represented by quantifier terms (qterms). For example, the sentence "*John saw a student*" has the unscoped logical form[1]

see'(john',qterm(a',X,student'(X)))

Since the only permissible scope for this quantifier is the whole sentence, the qterm is raised to produce the scoped logical form

quant(∃,X,student'(X), see'(john',X))

The qterm expression can best be thought of as a quant expression before its scope has been established. In the above qterm and quant expressions, student'(X) is the **restriction** of the quantified variable X; that is, it specifies a set of the possible values of X over which the quantifier ranges. In the above quant expression, see'(john',X) is referred to as either the **body** or the **scope** of the quantifier. This treatment of the logical form of quantifiers follows that employed in many previous systems (e.g., LUNAR (Woods, 1977), Moore (1981), Barwise and Cooper (1981), and Hobbs and Shieber (1987)).

[1] The logical form's syntax in the implementation is actually [see1,john1,qterm(a1,X,[student1,X])], but the more conventional notation will be used for perspicuity.

RULES AND PREFERENCES

Many of the following rules have appeared in various forms in multiple places in the literature, and most natural language processing systems include some mechanism for selecting a preferred quantifier scoping. However, the published descriptions of many of those systems' capabilities tend to be cursory, with the scoping rules utilized in the LUNAR system still among the best described in the NLP literature. Because of space limitations, it is not possible to cite much of this discussion, nor to compare this system to others.

Rule 1 *A quantifier* A *that is not in the restriction of quantifier* B *and that occurs within the scope of* B *cannot outscope any of the quantifiers in the restriction of* B.

Rule 2 *If a quantifier is raised past an operator, then any quantifier that occurs within its restriction must also be raised past that operator.*

These rules, presented by Hobbs and Shieber (1987), can best be explained with examples.

A bishop visits every chapel by a river. (6)

has an unscoped logical form of

```
visit'(qterm(a',B,bishop'(B)),
     qterm(every',C,and(chapel'(C),
                        by'(C,qterm(a',R,river'(R))))))
```

The following is one of the possible permutations of the quantifiers, but is not a valid scoping because the restriction of "*every*" ("*chapel by a river*") has been fragmented:

```
*quant(∀,C,chapel'(C),
       quant(∃,B,bishop'(B),
             quant(∃,R,and(river'(R),by'(C,R)),
                   visit'(B,C))))
```

Similarly, for the sentence

John did not visit a chapel by a river. (7)

the quantifier permutation

```
*quant(∃,C,chapel'(C),
        not(quant(∃,R,and(river'(R),by'(C,R)),
                visit'(john',C))))
```

is not a possible scoping of the unscoped logical form

```
not(visit'(john',qterm(a',C,and(chapel'(C),
                by'(C,qterm(a',R,river'(R)))))))
```

Rule 3 *For a set of quantifiers, which quantifier receives wide-scope preference can be determined by a pairwise comparison of the determiners. This comparison is based upon a combination of factors that include their relative strengths and surface positions, and whether or not either has been raised.*

In many systems, determiners are assigned numerical strengths and these values are compared to determine what scope should be assigned to each quantifier. Such a ranking is implicit in our preference rules and can be viewed as a first approximation of the relationships represented by our rules. Our algorithm permits a set of properties to be associated with determiners and for these to be used in ascertaining which determiner has wide-scope preference. The properties currently employed are surface position (the integer index of the determiner) and a Boolean value indicating when a quantifier has already been raised.

Preference 3.1 *There is a strong preference for "each" to outscope other determiners.*

That "*each*" is the strongest determiner is a common feature of most quantifier-scoping treatments. However, the evidence for the relative strengths of the remaining quantifiers is much less clear—our current ranking of them is an *ad hoc* blending of those in TEAM (Grosz *et al.*, 1987) and VanLehn (1978).

Preference 3.2 *There is a strong preference for WH-terms to outscope all determiners except "each," which outscopes WH-terms.*

In the unscoped logical forms currently produced, WH-words ("*which*," "*who*," "*what*") and phrases are represented as qterms. Our scoping-preference rules assign wide scope to "*each*" in

Which exams did each student pass? (8)

There is a reported dialect in which sentences of the above form are judged to be malformed, but that dialect was not found among our informants. The design of our algorithm makes it easy to replace the current preferences with these.

The definite determiner "*the*" is currently treated as a very strong quantifier, but this approach is not entirely satisfactory. Consider

Every student passed the exam. (9)

The student in every race celebrated. (10)

The student in each race celebrated. (11)

Every student in the race celebrated. (12)

Each student in the race celebrated. (13)

In (9)–(12), the preferred scopings are as predicted by the rules. However in (13), the preferred reading selected is the one with wide-scope "*each*." Although both scopings of this sentence are logically equivalent (as are those for (9) and (12)), wide-scope "*the*" seems to be the preferred reading.

Our algorithm does not distinguish between specific and nonspecific use of indefinite articles. It is debatable whether this belongs in quantifier scoping or in another part of the system.

Preference 3.3 *A logically weaker interpretation is preferred. This preference is strong when it maintains surface order, weak when it inverts surface order.*[2]

The quantifier order $\forall\exists$ is weaker than $\exists\forall$, accounting for the preferences in

A man loves every woman. (14)

Every man loves a woman. (15)

In both sentences, the reading with wide-scope "*every*" is the preferred one; the reading with wide-scope "*a*" is possible for (14), but is very strained for (15).

Rule 4 *Raising a quantifier out of certain syntactic constituents changes the strength of its determiner.*

VanLehn presents an "embedding hierarchy" of the probability of a quantifier in the modifier of an *NP* being raised to have wider scope than the quantifier in the *NP*'s head

[2] VanLehn proposes a more general form of this preference—that, when comparing two quantifiers within the same general group, the "more numerous" one will have a preference for wider scope. For example, "*many*" would take wider scope over "*few*." However, for everything except "*every*"/"*a*," such preferences appear to be very slight.

PP > Reduced Relative Clause > Relative Clause

A method frequently proposed to account for this distinction is to use, as a measure of the cost of raising, a count of the number of nodes in the syntactic structure over which the quantifier is raised. However, such accounts are acknowledged to have various deficiencies and to be overly sensitive to the syntactic representation used. We have chosen to permit rules to associate a cost for raising a quantifier with certain types of nodes (other nodes can be viewed as having zero costs). This capability of the system is currently invoked only on an all-or-nothing basis.

Preference 4.1 *A quantifier cannot be raised across more than one major clause boundary.*

A common rule in the quantifier-scoping literature is "quantification is generally clause bound." While it is possible to generate sentences with acceptable readings when a quantifier has wider scope than the clause in which it occurs, we have been unable to find any examples showing that it can be raised out of two clauses.

Preference 4.2 *A quantifier cannot be raised out of a relative clause.*

This is a common restriction in many quantifier-scoping algorithms. In our system, this is not a special rule, but one of the preferences. Consequently, this could easily be modified from never being permitted to being "highly unpreferred."

Rule 5 *In unscoped logical form, quantifiers can occur within the scope of an opaque operator. Whether or not to raise such a quantifier outside that operator is determined by a pairwise comparison between the operator and the determiner in the quantifier, as well as by their relative surface position.*

Preference 5.1 *There is a strong preference for "some" to outscope negation.*

Preference 5.2 *There is a preference for negation to outscope "every." This preference is strong when it maintains surface order, weak when it doesn't.*

Different scopings of "*some*" and "*every*" under negation produce equivalent readings ($\exists\neg$ is equivalent to $\neg\forall$). The preferred scopings for the two sentences

John did not see someone. (16)

John did not see everyone. (17)

have equivalent logical forms

quant(∃,P,person'(P),not(see'(john',P)))
not(quant(∀,P,person'(P),see'(john',P)))

Similarly, the preferred scopings of sentences

Someone did not see John. (18)

Everyone did not see John. (19)

have equivalent logical forms

quant(∃,P,person'(P),not(see'(P,john')))
not(quant(∀,P,person'(P),see'(P,john')))

The reading of (16), which would assign narrow scope to "*some*" is produced by substituting "*any*"[3] for "*some*":

John did not see anyone. (20)

This has the following logical form (no other scopings exist):

not(quant(∃,P,person'(P),see'(john',P))) ,

which is logically equivalent to

quant(∀,P,person'(P),not(see'(john',P))) ,

which corresponds to the strongly "unpreferred" readings of (16) and (17). Similarly, the sentence

No one saw John. (21)

which has a scoped logical form of

quant(∀,P,person'(P),not(see'(P,john')))

corresponds to the "unpreferred" scoping for (18) and (19).

One of LUNAR's scoping rules was that in the antecedent of "*if-then*" statements, quantifiers "*some*" and "*any*" should be assigned wide scope, and that "*a*" and "*every*" should be given narrow scope. If such antecedents were treated as a negative environment (or equivalent thereto), the foregoing preferences could produce this effect.

[3] The *CLE* system does not currently provide a treatment of "*any*." However, within the quantifier-scoping component, "*any*" is treated as being potentially ambiguous between the usual universal quantifier, free-choice "*any*," and a second form, polarity-sensitive "*any*," which occurs in conjunction with negative-polarity items. Polarity-sensitive "*any*," is treated as a narrow-scope existential quantifier (Ladusaw, 1980).

Preference 5.3 *There is a strong preference for free-choice "any" to have wider scope than modals. There is a strong preference for all other determiners that occur within the scope of a modal to have narrower scope than that modal.*

Did some student take every test? (22)
Does some student take every test? (23)
Some student took every test. (24)
Some student takes every test. (25)
Some student is taking every test. (26)

For sentences (23), (25), and (26), there are two acceptable quantifier scopings. However, for (22) and (24), the scoping in which "*every*" is assigned narrower scope seems to be strongly preferred. We ascribe this to the presence in the logical form of a modal operator corresponding to the past tense. This effect is accentuated in (27), which exhibits an ambiguity resulting from whether "*some teacher*" is scoped inside or outside the modal, corresponding to (28) and (29), respectively:

Some teacher took every course. (27)
Last summer, some teacher took every course. (28)
As a student, some teacher took every course. (29)

The scoping in which "*every*" outscopes "*some*" is possible, although unpreferred, for the reading (28); but it is not a possible scoping for (29) in any dialect that we have encountered.

Rule 6 *If polarity-sensitive "any" occurs within a clause in which its trigger does not occur, it must be raised out of that clause.*

De Dicto/De Re *The mechanism described here can provide an account for the de dicto/de re distinction.*

Another ambiguity associated with quantifier terms is whether or not the referent is required to exist. In *PTQ* (Montague, 1973), the sentence

John seeks a unicorn. (30)

is assigned a *de dicto* reading (which does not require that any unicorns exist),

seek'(^john',^λ(P,quant(∃,X,unicorn'(X),^P(X))))

and a *de re* reading (which requires the existence of some unicorn)

quant(∃,X,unicorn'(X),seek'(^john',^λ(P,^P(X))))

In *PTQ*, this distinction is produced by syntactic rules. Cooper (1975, 1983) demonstrated that a mechanism using a store could produce both readings from a single logical form.

Our mechanism obtains similar results. Starting from the unscoped logical form

seek'(^john',^λ(P,^P(qterm(a',X,unicorn'(X)))))

with the intension operator ^ treated as being optionally opaque, both readings are produced by the quantifier-scoping algorithm described here. Additional (unwarranted) scopings are not produced because these are the only two sites at which quantifiers can be pulled from the store.

Nonrule *There is a strong preference for a noun phrase in a prepositional phrase complement to outscope the head noun.*

This criterion is used in many quantifier scoping mechanisms. It is a good heuristic, but it is not a reliable rule. In

John visited every house on a street. (31)
John visited every house with a dog. (32)

the heuristic correctly predicts the preferred scoping for (31), but fails for (32).[4] This heuristic is *not* part of our scoping algorithm; we believe that its effects are part of the processing consigned by us to the second phase of quantifier scoping (future work).

BASIC ALGORITHM

The first level of our scoping algorithm generates the possible scopings, as described by Hobbs and Shieber (1987). However, we implemented this with a different algorithm, partly for reasons of efficiency and partly because it could be easier expanded to include additional capabilities. The performance of the Hobbs and Shieber algorithm deteriorates as the number of quantifiers in the sentence increases—our analysis is that it spends a significant amount of time repeatedly traversing the logical form and doing structure copying (their goal was to produce a provably correct algorithm, not a highly efficient one). Our algorithm traverses the unscoped logical form, collecting the qterms (quantifier terms) into a store; then as the scoping for each qterm is determined, it is pulled out of the store, producing a scoped logical form.

[4] This was brought to my attention by Richard Crouch.

For a sentence with four quantifiers, our algorithm is typically an order of magnitude faster than that presented by Hobbs and Shieber.

A simple example of the use of the store is provided by the sentence "*John saw a student,*" which has an unscoped logical form of

```
see'(john',qterm(a',X,student'(X)))
```

After quantifier scoping has placed the qterm in the store, the logical form is

```
see'(john',X)
```

and the store is

```
[[qterm(a',X,student'(X))]]
```

The scope for this quantifier is the whole sentence, so the qterm is pulled out of the store to produce the scoped logical form

```
quant(∃,X,student'(X), see'(john',X))
```

The sentence "*Few students pass most exams*" has the unscoped logical form

```
pass'(qterm(few',X,student'(X)),
      qterm(most',Y,exam'(Y)))
```

After the qterms have been extracted, the remaining logical form and the store are

```
pass'(X,Y)
[[qterm(few',X,student'(X))],
 [qterm(most',Y,exam'(Y))]]
```

A qterm can have other qterms in its restriction and our quantifier store is a structured collection (unlike the stores of Cooper and LUNAR). The structure of qterms in the store corresponds to their relative positions in the unscoped logical form. For example, the unscoped logical form for "*every student in a college attends the lecture*" is

```
attend'(qterm(every',X,and(student'(X),
                    in'(X,qterm(a',Y,college'(Y))))),
        qterm(the',Z,lecture'(Z)))
```

When such qterms are placed in the store, this relationship is maintained by representing the collected qterms as trees (called qtrees), with the outer qterm as the root and those in its restriction as daughters:

```
[[qterm(every',X,and(student'(X),in'(X,Y))),
    qterm(a',Y,college'(Y))],
 [qterm(the',Z,lecture'(Z))]]
```

Consequently, the store is a forest of such qtrees, and the qterms occurring in the restriction of a qterm are themselves a forest of qtrees and are treated as if they were a store.

As qterms are collected, they are inserted into the store in inverse order of preference—e.g., the qterm that has narrowest-scope preference appears at the front of the list representing the forest. In implementing this algorithm in Prolog, we found that it was considerably easier to generate the scopings by working from the narrowest to the widest scope, rather than *vice versa*. As the various permutations of the quantifiers are generated, equivalent scopings are detected, and all but the most preferred one are then filtered out. In the following, both scopings of each sentence are logically equivalent:

Every student takes every test.	(33)
Every student takes each test.	(34)
A student takes a test.	(35)
Some student takes a test.	(36)
Each student takes the test.	(37)
Every student takes the test.	(38)
The student takes every test.	(39)

In (33), (35), (37), and (39), the preferred order is the same as the surface order, while in (34), (36), and (38), the stronger quantifier occurs second in surface order, and the scoping that corresponds to surface order is discarded. Filtering of equivalent permutations is achieved simply by comparing the qtree currently being pulled from the store with the preceding one; if the quantifiers in their head qterms are logically equivalent, this quantifier scoping is discarded unless the qtree being pulled has wide-scope preference over its predecessor (in which case the other logically equivalent ordering will be discarded).

Logically equivalent scopings can also be produced when a quantifier is raised out of the restriction of another. However, the quantifier permutations that produce equivalent scopings by raising are a subset of those produced by permuting siblings:

Every student in every race celebrated.	(40)
A student in a race celebrated.	(41)
Some student in a race celebrated.	(42)

Each student in the race celebrated. (43)

Every student in the race celebrated. (44)

The student in every race celebrated. (45)

Note that the scopings for (40) and (45) are not logically equivalent. The scopings in the others are logically equivalent, but in (41) and (43), the preferred scoping is the one corresponding to constituent structure, whereas in (42) and (44), the preferred scoping has the *NP* from the *PP* raised to have wider scope over the head noun.

When a qtree is pulled from the store, the algorithm tries to produce additional permutations by raising subsets of qterms (actually of qtrees) out of that qtree's restriction. When a qtree is raised, it is put back into the store—since qtrees are being assigned scope from narrowest to widest, this ensures that a raised qtree will receive wider scope than the qtree out of which it was raised.

Because a raised qtree may have its strength reduced when it is placed back in the store (an option in our system), a set of logically equivalent scopings could have *all* instances filtered out by a naive implementation. The problem arises in the following manner. Before the qtree is raised, the algorithm determines that the unraised scoping is logically equivalent to a raised one and that the latter is preferred, so it discards the former. When the qtree is raised and its strength reduced, it becomes weaker than the qtree out of which it was raised. The algorithm detects that the raised scoping is logically equivalent to an unraised one, and determines—on the basis of the current strengths—that the unraised scoping is preferred, so it now discards the raised one. This problem is avoided by doing some additional bookkeeping.

The current implementation of the above rules is very coarse-grained. The "score" indicating whether or not a quantifier should be assigned wide scope over another quantifier, logical form operator (e.g., a modal, negation), or syntactic constituent is one of four values: *always* (narrow scope is impossible), *never* (wide scope is impossible), *pref* (wide scope is preferred, but narrow scope is acceptable), and *unpref* (narrow scope is preferred). In the current implementation of the above preferences, a strong preference to take wider scope is treated as an instance of *always*, and a weak preference is treated as *pref*. For example, Preferences (3.1)–(3.3) are given by the following rules, in which Pref is the preference of a determiner Det1 to take wider scope over another determiner Det2:

if Det1 and Det2 are both "*each*":
- if Det1 precedes Det2 in surface order, Pref = *pref*,
- otherwise, Pref = *unpref*.

otherwise, if Det1 is "*each*" (and Det2 is not), Pref = *always*

otherwise, if Det1 is an interrogative determiner, Pref = *always*

otherwise, if the logical forms for Det1 and Det2 are ∀ and ∃, respectively:
- if Det1 precedes Det2 in surface order, Pref = *always*
- otherwise, Pref = *pref*.

Overshoot *The method described here results in some quantifiers' being assigned scopes that are wider than appropriate, relative to other predicates (but not quantifiers) in the logical form.*

The sentence "*John visited every person on a committee*" has an unscoped logical form of

```
visit'(john',qterm(every',P,and(person'(P),
                on'(P,qterm(a',C,committee'(C))))))
```

and its preferred scoping is

```
quant(∀,P,quant(∃,C,committee'(C),
                  and(person'(P),on'(P,C))),
      visit'(john',P))
```

Note that person'(P) is independent of C; thus it can be outside the scope of the quantifier for C

```
quant(∀,P,and(person'(P),
                quant(∃,C,committee'(C),on'(P,C))),
      visit'(john',P))
```

Such transformations can have a significant impact on the performance of the system, substantially reducing the processing time of queries for even a modest database. Rather than pass additional information so that quantifiers could be pulled at the correct point in the traversal of the logical form, we chose to let the scoping algorithm "overshoot" its mark and then lower the quantifiers to the correct position. This was considerably easier to implement, and it does not seem to have any performance penalty in our system.

CONCLUSION

For lack of a reasonable corpus of human quantifier scoping preferences, the testing of this system has been limited to checking conformance to

the stated rules.[5] The semantic component of the *CLE* does not produce logical forms with mass or count *NP*s or collective readings, but that capability is currently being developed. The foregoing description of qterms is a slight simplification; an extended form is now being used to support generalized quantifiers in the new semantic rules.

Examples offered by VanLehn (1978) indicate that dative movement affects quantifier scoping, but the cause may actually be domain or discourse information. Our examples show that passivization affects quantifier scoping, but we have not yet found a means of determining whether the effect is due solely to the cost of raising out of the *PP*.

The algorithm does not handle "donkey sentences," nor is it intended to. A scheme for handling such sentences is being explored as part of the continuing development of the *CLE* (Fernando Pereira, personal communication). This would be a separate mechanism, rather than an extension of quantifier scoping.

ACKNOWLEDGMENTS

The research on which this paper is based was supported by the Natural Language Processing Club (NATTIE) of the Alvey Directorate program in Intelligent Knowledge-Based Systems (Project No. ALV/PRJ/IKBS/105). Most of it was performed while I was a member of SRI's Cambridge Computer Science Research Centre. This work benefited from extensive discussion with and suggestions from Robert C. Moore and Hiyan Alshawi.

REFERENCES

Alshawi, Hiyan; Moore, Robert C.; Moran, Douglas B.; and Pulman, Steven G. 1987. Research Programme in Natural-Language Processing, Annual Report to the Natural Language Processing Club (NATTIE) of the Alvey Directorate Program in Intelligent Knowledge-Based Systems, Cambridge Computer Science Research Centre, SRI International, Cambridge, England.

Barwise, Jon and Cooper, Robin 1981. Generalized Quantifiers and Natural Language. *Linguistics and Philosophy* 4(2): 159-219.

Cooper, Robin 1975. *Montague's Semantic Theory and Transformational Syntax*. Ph.D. dissertation, Department of Linguistics, University of Massachusetts at Amherst, Massachusetts.

Cooper, Robin 1983. *Quantification and Syntactic Theory*, D. Reidel, Dordrecht, Holland.

Grosz, Barbara J.; Appelt, Douglas E.; Martin, Paul A.; and Pereira, Fernando C.N. 1987. TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces. *Artificial Intelligence* 32(2): 173-243.

Hobbs, Jerry R. and Shieber, Stuart M. 1987. An Algorithm for Generating Quantifier Scopings. *Computational Linguistics*, 13(1-2): 47-63.

Ladusaw, William 1980. *Polarity Sensitivity as Inherent Scope Relations*. Ph.D. dissertation, Department of Linguistics, University of Texas at Austin; published by Garland Press, New York, New York.

Montague, Richard 1973. The Proper Treatment of Quantification in Ordinary English. In: Hintikka, J.; Moravcsik, J.; and Suppes, P. (eds.) 1973. *Approaches to Natural Language*, D. Reidel, Dordrecht, Holland: 221-242. Reprinted in: Montague, Richard 1974. *Formal Philosophy: Selected Papers of Richard Montague*, edited and with an introduction by Richmond Thomason, Yale University Press, New Haven, Connecticut: 247-270.

Moore, Robert C. 1981. Problems in Logical Form. In *Proc. of the 19th Annual Meeting of the Association for Computational Linguistics*: 117-124.

VanLehn, Kurt A. 1978. Determining the Scope of English Quantifiers. Report AI-TR-483, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Woods, William A. 1977. Semantics and Quantification in Natural Language Question Answering. In: *Advances in Computers, Volume 17*, Academic Press, New York, New York: 1-87.

[5] The range of quantified noun phrases covered in the algorithm is larger than what is currently produced by the syntactic and semantic components of the *CLE* system. Such extensions have been tested by starting from the anticipated logical form.