Scope Economy and TAG Locality

Michael Freedman
Yale University
370 Temple Street, Rm 210
New Haven, CT 06511, USA
michael.freedman@yale.edu

Abstract

This paper gives an analysis explaining various cases where the scope of two logical operators is non-permutable in a sentence. The explanation depends on a theory of derivational economy couched in the synchronous tree adjoining grammar framework. Locality restrictions made using the STAG formalism allow us to limit the computational complexity of using transderivational constraints and allows us to make interesting empirical predictions.

1 Introduction

Although the relative scope of quantifiers is generally free within a single clause in English, there are a number of cases where the relative scope of quantifiers/operators is non-permutable. We provide here an account for the data below that depends on a theory of derivational economy. It assumes a syntax-semantics interface couched in the synchronous tree adjoining grammar framework (STAG) (Shieber and Schabes, 1990; Han, 2006; Nesson and Shieber, 2007). The intuition underlying the current analysis is a familiar one: that interpretations of a given syntactic derivation are blocked when there is a simpler, competing derivation that can produce the same meaning. However, by couching the intuition in the context of STAG, we derive a number of important limitations on such competition by restricting the comparison class (using the TAG formalism), and maintain a system that is limited in the complexity of the computations it assumes. This allows us to make empirical predictions that other accounts of scope economy couched in other frameworks are unable to make because either they view competition to be global (Grice, 1989; Horn, 1989) or too local (Fox, 1995, 2000).

2 Universals-Negation

Universal quantifiers like *every* and negation are unable to scope freely with respect to one another. When a universal quantifier is in object position, as in (1), it is unable to take scope over negation.

(1) a. Peter didn't catch every crook. (*$\forall > \neg$, $\neg > \forall$)
b. Nobody caught every crook. (*$\forall > \neg\exists$, $\neg\exists > \forall$)

Similarly, when a universal is in subject position, it is unable to take scope over negation, as in (2). The pattern breaks down in a number of cases: in (3a), the presence of a raising predicate makes the wide scope universal reading possible; also, in (3b), the presence of an intervening operator allows the universal to scope over negation.

(2) Everyone didn't meet Peter. (*$\forall > \neg$, $\neg > \forall$)

(3) a. Everyone didn't seem to meet Peter. ($\forall > \neg$, $\neg > \forall$)
b. Everyone didn't meet someone. ($\forall > \exists > \neg$)

The goal of the analysis is to explain lack of scope permutability in (1) and (2) while allowing scope permutability in (3).

3 Preliminary assumptions

We assume following Schabes and Shieber (1994) the declarative conception of STAG where quantifier scope is determined in the derivation tree: Here scope can be resolved on the left-to-right order of nodes on a derivations tree; by convention rightmost nodes in the derivation tree correspond with higher scope in the derived tree. In Schabes and Shieber (1994) scope ambiguity is achieved by allowing either ordering in the derivation tree.

Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11), pages 223–231,
Paris, September 2012.

In contrast, we follow the restriction on derivation proposed in Freedman and Frank (2010) which is defined as follows:

> PROMINENCE RESTRICTION ON DERIVATION (PROD): The children of a node γ (representing elementary tree τ) in a derivation tree must be in an order consistent with both the domination and c-command relations of their corresponding elementary tree's attachment sites on τ.

Figure 1 shows how PRoD works: trees adjoining into δ or ϵ must be ordered before those that adjoin into β or γ which must be ordered before those that adjoin into α.

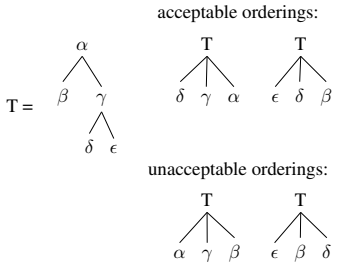

Figure 1: Sample elementary tree T (left) and sample derivation trees that show possible and impossible orderings of attachment following PRoD on derivation tree rooted with a node representing T.

By PRoD, one could not have the subject-tree ordered before the object-tree because the attachment site for the former c-commands the latter. To handle cases of inverse scope, multi-component tree-sets can combine through *split combination (SC)*:

> SPLIT COMBINATION: The node representing a scope-tree within a tree-set may be ordered later than specified by PRoD.

SC allows the scope portion of a quantifier tree set to take scope over a dominating or c-commanding quantifier. The use of split combination adds an extra step in the derivation by separating the scope-tree from the variable tree in the derivation tree. An example is provided in figure 2 for the sentence *a student read every book*; this compares to figure 3 where the tree-set for the quantifier is not split and surface scope is generated. Additionally, SC can be lexically restricted such that only certain quantifiers are able to split.

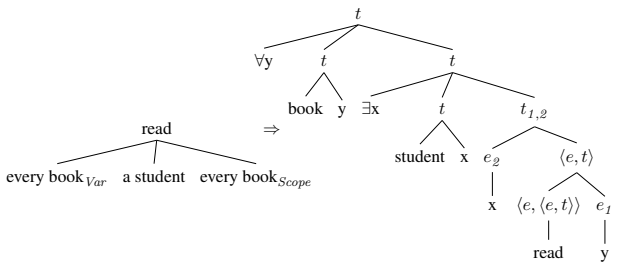

Figure 2: Derived and derivation tree for "A student read every book" with split combination of "every book".

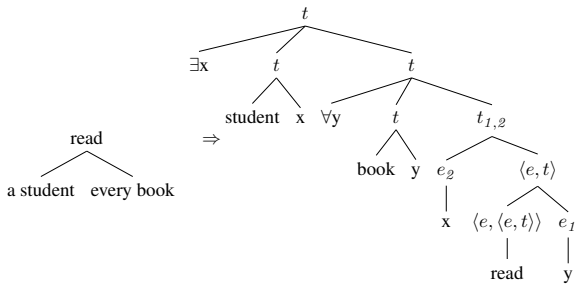

Figure 3: Derived and derivation tree for "A student read every book" without split combination of "every book".

The size of elementary trees follows the Condition on Elementary Tree Minimality (Frank, 2002). For our purposes this dictates that quantificational determiners are in the same elementary tree as the noun that is in their restriction (as depicted in figure 4). In the syntax, clausal negation resides in extended projection of a verb. In the semantics, we assume that negation adjoins in separately from the verb-tree, adding another step in the derivation (as depicted in figure 5).

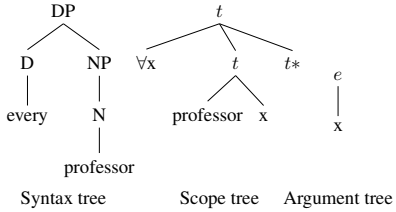

Figure 4: Elementary trees for the Quantifier Phrase "every professor" including the syntax tree and the multi-component semantic tree set consisting of the scope-tree and the argument tree.

4 Analysis

With this general picture in mind, let us proceed to analyse the data in §2: Observe that the unavail-

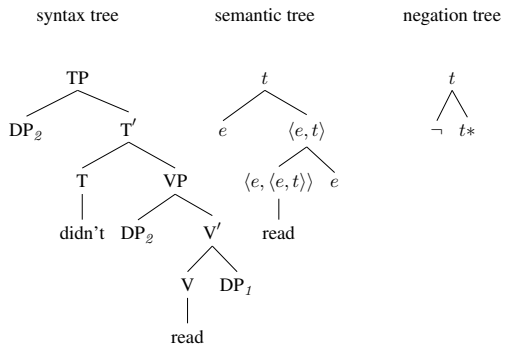

Figure 5: Elementary trees for the verb "read" with negation "didn't" in its extended projection. The semantic tree does not contain the negation; it is a separate tree.

able $\forall\neg$ reading for (1) is available for the example in (4), as $\forall\neg$ and $\neg\exists$ are logically equivalent.

(4) John didn't read a/any book(s). ($\neg>\exists$)

While the derivation producing this interpretation would require split combination for (1), it does not for (4). Because SC creates an additional step, the derivation for (4) is shorter than the one for (1). An intuitive way of understanding this pattern is to think that (4) blocks (1) on the $\forall\neg$/$\neg\exists$ reading. A similar line of analysis can explain (2)'s inability to have a $\forall\neg$ reading; it has a competitor (5) that can produce the same meaning ($\neg\exists$) with a shorter derivation.

(5) No one met Peter. ($\neg>\exists$)

(5) on the relevant reading has a shorter derivation because both quantifiers ($\neg$ and $\exists$) are lexicalized in a single tree; the negative and existential force in (2) must be combined in separately, creating an extra step. This blocking intuition can be formalized in the following manner:

> DERIVATIONAL COMPLEXITY CONSTRAINT ON SEMANTIC INTERPRETATION (DCCSI)
> A derivation *d* producing meaning *m* is ruled out if another shorter derivation d' also produces *m*.

This constraint explains the data in (1) and (2); both have more economical alternatives that block them. Derivation trees for (1) and (4) are provided in figure 6 showing the difference in derivation length. The DCCSI also explains the ability for the universal to take wide scope in (3); because an operator intervenes, there is no more economical competitor, and no blocking can take place.

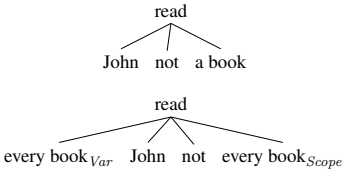

Figure 6: Derivation trees for "John didn't read every book" and "John didn't read a book" on the $\forall>\neg$ reading.

But, as the DCCSI stands, it is unrestricted with respect to comparison class. This causes empirical and computational problems. Computational issues are discussed in §6. Empirically, the unrestricted constraint leads to the puzzle posed by the data in (3a) (repeated as (6a)): why isn't the wide scope universal reading blocked by (6b). Given the definition of the DCCSI, blocking should take place as the relevant derivation for it is shorter than the relevant derivation for (3a).

(6) a. Everyone didn't seem to meet Peter. ($\neg>\forall$, $\forall>\neg$)
b. No one seemed to meet Peter. ($\neg>\exists$)

To restrict the comparison class we impose a constraint that closely follows TAG intuitions about locality. The TAG-like intuition behind this constraint is that comparison is localized to elements combining into a single elementary tree. The derivation can informally be defined in the following way: A derivation tree D is comparable to derivation tree D′ iff D and D′ are identical except for the daughters of a single node α. The possible differences between D and D′ include D excising or replacing one or more of α's daughters. A more formal definition follows:

> LOCALITY CONSTRAINT ON COMPETITION (LCC) (formal version)
> Derivation tree D′ can be compared to derivation tree D (where D and D′ are defined as triples consisting of a set of nodes N, a set of labels L and the immediate dominance relation P) iff:
>
> 1. $N' \subseteq N$
> 2. $P' \subseteq P$
> 3. $\exists! n \in N s.t.$
>   (a) $P' \supseteq P - \{(n, x) | x \in N\}$

(b) $\forall x \in N, (n, x) \notin P \rightarrow L(x) = L'(x)$

The first two clauses ensure that the derivation trees in the comparison class have no structure that is not present in the original derivation tree D. The third clause ensures that the only change is under a single node of the derivation tree and that daughters can be deleted or labels changed.

The LCC has the benefit of limiting the comparison class in a way that not only makes transderivational constraints feasible but also explains the puzzle of the wide scope reading for (3a). The availability of the wide scope universal reading for the example in (3a) can be understood in conjunction with the LCC; while there is a sentence that has a shorter derivation, (6b), it is not able to be compared to (3a) because there are differences under more than one node of the derivation trees (as depicted in (7)): The *didn't*-node is deleted under the *seem*-node and the *everyone* label is changed to the *no one*-label. This finding is possible because of the TAG analysis of raising where raising predicates adjoin into a VP (as shown in figure 8). Since the clausal negation combines into the raising predicate and the nominal quantifier negation combines into the main verb-tree the comparison is non-local, as depicted in figure 7.

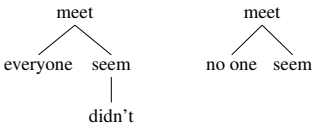

Figure 7: Derivation trees for "John didn't seem to meet everyone" and "No one seemed to meet everyone" on the $\forall > \neg$ reading.

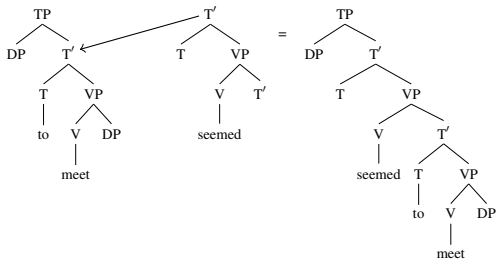

Figure 8: Derivation for a raising construction: the *seem*-tree (the raising predicate) adjoins into the *meet*-tree

.

Computationally, the DCCSI in conjunction with the LCC can be proven to not exceed the expressive power of TAG. This can be done with a proof by construction where a TAG with the DCCSI can be compiled into a TAG without the DCCSI. In addition, with proper linguistically motivated constraints, it can shown that the size of the grammar does not become unsuitably large with the compilation. This is shown in §7.

5 Double Object Constructions

It has been observed that the relative scope of the internal arguments of double object sentences are only able to have a surface scope reading (Larson, 1988; Bruening, 2001). That is, the sentence in (7a) can have the reading where every photograph was given to one individual but not a reading where the photographs differed in who they were given to. The prepositional dative sentences in (9) differ from those the double object in (7) in that they are scopally ambiguous; both the surface and inverse scopes for the internal arguments are available.

In the double object case, as in (7) and assuming the elementary tree for the double object in figure 9a, inverse scope interpretation can be obtained through the use of split combination which would place the scope of the second object in a higher position than the first object.

(7) a. Peter gave someone every/each photograph. (*$\forall > \exists$, $\exists > \forall$)

b. Harry told someone every/each plot. (*$\forall > \exists$, $\exists > \forall$)

Note however that the prepositional dative alternative in (8) (and assuming the elementary tree in figure 9b) does not need SC in order to obtain the same reading, making its derivation shorter.

(8) Peter gave every photograph to someone. ($\forall > \exists$)

(9) a. Peter gave every photograph to someone. ($\forall > \exists$, $\exists > \forall$)

b. Peter put a bagel on every shelf. ($\forall > \exists$, $\exists > \forall$)

Thus, blocking removes the inverse scope structure from the grammar. Why then are the ditransitive sentences in (9) ambiguous? For the

prepositional dative cases in (9a,b) the PP argument is able to optionally attach to a higher position than the direct object argument thereby allowing "inverse scope" without SC (the PP can also be under the direct object in a VP-shell), as depicted in figure 9c. Evidence for this structural explanation come from examples like that in (10) where complex existential quantifiers (which have been observed not to take scope over other higher quantifiers (Beghelli and Stowell (1997); Heim (2001))) take scope over a higher quantifier.

(10) John gave an apple to more than three students.

The examples in 11 provide additional evidence for the blocking analysis: In (11a), no blocking occurs if there is an intervening operator that would force the prepositional dative construction to utilize SC to obtain the same scope reading (making the derivation length the same). Likewise, as in (11b), no blocking occurs in constructions that have no prepositional dative construction to compete with.

(11) a. A teacher gave every student every book. ($\forall_{book} > \exists > \forall_{stu}$)

b. Peter bet a friend every nickel (he had). ($\forall > \exists$, $\exists > \forall$)

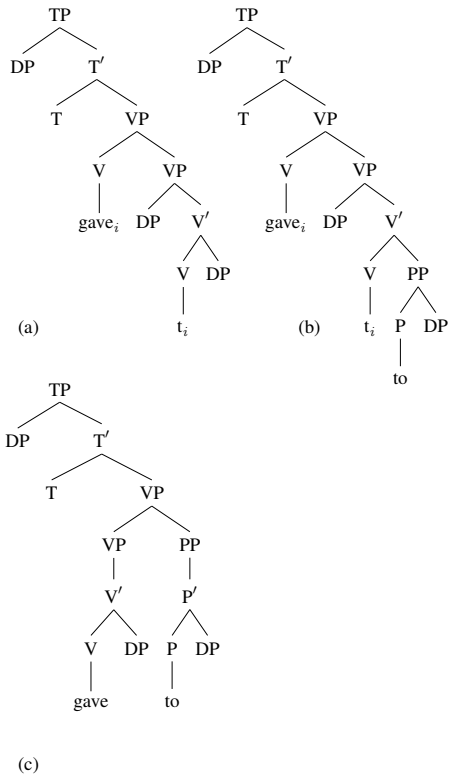

Figure 9: The three elementary trees for ditransitive constructions: (a) double object; (b)dative complement (low attachment); dative complement (high attachment)

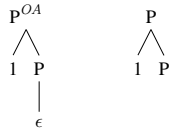

Figure 10: This figure shows TAG G'; it consists of trees that can build up the natural numbers in unary. The left and right trees construct numbers by successively "adding" 1.

6 Complexity

6.1 TAG with TDC is not a TAL

I show in this section how a TAG with a TDC can generate a language that contains exactly the prime numbers. Since the language that contains only the prime numbers is not in the class of mildly context sensitive languages, the language that is generated by this grammar is not a TAL. This shows that adding a transderivational "economy" constraint can increase the generative capacity of a grammar. We can construct this language in the following way:

Construct a TAG G that composes the natural numbers. Figure 10 shows a TAG G' that generates unary strings that can be interpreted as the natural numbers beginning with 2. The initial tree has an obligatory adjoining constraint; the grammar only generates unary strings of length > 1. Each adjoining operation increases the interpreted value of the string by 1.

An additional tree is added (in figure 11) to the grammar G' (let's call it G) in order to be able to represent all of the natural numbers greater than 1 as products of other natural numbers. The tree contains two nodes where adjunction can occur: there is one OA site where at least one 1-tree must obligatorily adjoin and one SA site where a tree may optionally adjoin. Adjoining into this tree can produce any natural number except 1.

The grammar G consisting of the trees in figures 10 and 11 generates the language that consists of the natural numbers in unary and the natural numbers as products of other natural numbers (represented in unary). Now, we introduce a transderivational constraint that will remove from the grammar any derived tree whose value can be

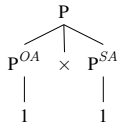

Figure 11: Additional tree for TAG G; this allows natural numbers to be represented by the product of other natural numbers.

more simply derived (fewer derivational steps) by an alternative derivation. We will call the resultant grammar G^c. This constraint is similar to the constraint used in the first half of the paper for grammatical constructions; equivalence in number and equivalence in meaning are both instances of logical equivalence where it is necessary for the truth values to be the same in any possible model (although meaning as defined requires a bit more than logical equivalence). A more formal definition follows:

(12) **TDC on Number Generation** A derivation d taking k steps and generating a string that is interpreted as natural number n is ruled out if a derivation d' takes fewer than k steps and also produces n.

The length of derivation for the purely unary method of combination is equal to the value of the number minus one $(n-1)$. The length of derivation with the product-tree is the sum of the value of each product minus two $(p+m-2)$.

For any prime number no blocking occurs because the length of derivation for both the unary-string and the product-string is the same. The proof by contradiction follows: Suppose that for any prime $n-1 \neq p+m-2$. For any prime we know that (1) one of the factors will be 1 $(m=1)$ and that the other factor will be equal to the number $(n=p)$. Replacing m with 1 and p with n we get the equation $n-1 \neq n-1$. Thus, $n-1 = p+m-2$ for any prime number n.

For non-prime numbers there will always be a product-tree derivation that is shorter than the unary-tree derivation. To do this we want to prove that for all n there is a p and an m such that $n-1 > p+m-2$ which reduces to $pm > p+m-1$ when replacing n with pm. This can be proven directly:

First solve for m: $pm > p+m-1 \Rightarrow pm-m > p-1 \Rightarrow m(p-1) > p-1 \Rightarrow m > 1$.

Then solve for p: $pm > p+m-1 \Rightarrow pm-p > m-1 \Rightarrow p(m-1) > m-1 \Rightarrow p > 1$.

This shows that the inequality holds as long as m,$p > 1$. Since the problem is defined on non-prime factors, the equality holds in all the relevant cases and the TDC will remove all purely unary representations of non-prime numbers.

Thus with the transderivational constraint all unary representations of non-prime numbers will not be members of G^c.

Next, we will intersect the constructed TAL (U) (from TAG G^c) with the complement of the language that contains the product representation of the natural numbers in unary (L'). L' is a regular language because L is a regular language and regular languages are closed under complementation. Since TALs are closed under intersection with regular languages $U \cap L'$ is a TAL if U is. But, the strings that comprise $U \cap L'$ are only the prime numbers: Since, this language does not have the constant growth property it cannot be a TAL. This result shows that the transderivational constraint takes the grammar out of the class of TAL because the properties of all of the other elements of the construction are known.

6.2 TAG with TDC is beyond NP

This section shows that a TAG grammar with the addition of a TDC is beyond NP. This is shown by generating the language MINIMAL.

The minimization problem for propositional formulas seeks to find a minimum equivalent formula for a given boolean formula; the language MINIMAL consists of all well-formed boolean expression for which there is no shorter equivalent formula (Meyer and Stockmeyer, 1972). Minimality ('shortness') can be defined in a number of ways and for the purposes of this proof it will be defined by the number of connectives in the formula. The complexity class of the minimization problem is unknown but it is known to be beyond NP.

The first step of the proof is to make a TAG that generates the set of propositional formulas. The syntax for PL can be defined in the following manner:

(13) **Syntax of PL**

1. any statement letter α is a well-formed formula (*wff*);
2. if α is a *wff* then $\ulcorner\neg\alpha\urcorner$ is a *wff*;

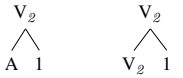

Figure 12: This figure shows how atomic propositions are constructed. Atomic propositions would be the following: A1, A11, A111, etc.

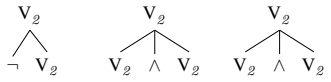

Figure 13: This figure shows trees that with the atomic proposition trees construct (from left to right) negation, conjunction, and disjunction.

3. if α and β are *wff* then $\ulcorner(\alpha \wedge \beta)\urcorner$ is a *wff*;
4. if α and β are *wff* then $\ulcorner(\alpha \vee \beta)\urcorner$ is a *wff*;
5. Nothing else is a *wff*.

A TAG version of (13) follows (which we will call G): The first step creates the atomic propositions in PL. Because of the finite limit to the number of trees in a TAG, the syntax has to recursively construct the atomic propositions. The construction includes an initial tree and an auxiliary tree where the initial tree encodes a single proposition. Each instance of adjunction is a new atomic proposition. The two trees are shown in figure 12. The trees that encode the logical connectives are in figure 13. These trees can combine with one another through substitution to make the full set of possible well-formed formulas for PL.

These trees form G. We can add a transderivational constraint to the grammar (making grammar G^c):

(14) A derivation d taking k steps and producing PL formula m is ruled out if a derivation d' takes fewer than k steps and also produces m.

The addition of the TDC to the grammar would define the language of propositional formulas that do not have alternative semantically equivalent formulas that can be constructed by the TAG in the same or fewer derivational steps. This is equivalent to the minimization problem; if we had an oracle that could determine for any formula ϕ if it was a member of the TAL then we would know the answer to the membership problem for MINIMAL or if we had an oracle that could determine for any formula ϕ if it was a member of MINIMAL, we would know whether the corresponding tree was in G^c. Since a solution for the minimization problem is beyond NP, the member ship problem for the language generated by G^c is as well.

Given the results that the language generated by a TAG with a TDC is not a TAL and is beyond NP for propositional logic, it is clear that an unrestricted TAG with TDCs is unwanted. I will present a construction in the next section that shows that TAG with the DCCSI and with the addition of the locality constraint (LCC) described in the previous chapter is a TAL.

7 The Expressive Power of the DCCSI

In this section, I will show that the expressive power of TAG is not increased by adding TDCs into the grammar: A TAG with the DCCSI constraint does not exceed the expressive power of a similar TAG variant that does not have the DCCSI. I do this through a compilation: the formalism described in this paper is algorithmically transformed into a standard STAG that is known to be in the mildly-context sensitive class of grammar formalisms. The translation takes three steps: (1) The effects of multiply linked semantic nodes (MLSNs) will be recreated using the formalism outlined in this paper but without having any MLSNs. This step will also make sure that the relevant links are completely ordered with respect to one another. This will allow there to be a single tree for every possible scope configuration. (2) The effects of PRoD and *split combination* will be recreated using the grammar created in step 1 without PRoD and *split combination* and with the overt addition of features to recreate their effects. The addition of features will allow the removal of trees that violate PRoD. (3) The effects of the DCCSI with the grammar created in step 2 will be recreated without the use of the DCCSI. This will be done by eliminating structures that have unwanted scope configurations. Through each step I will show that the increase in the number of trees in the grammar is bounded in a non-problematic way given some restrictions on the properties of natural language grammars. This will show that the grammar created in this paper is no more powerful than a STAG.

This type of proof is possible because the LCC localizes comparison to a single elementary tree.

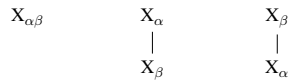

Figure 14: This figure shows the conversion from a multiply linked semantic node to nodes corresponding to the possible orderings of the links. On the left is a node with links α and β. The two trees on the right correspond to the two possible orderings.

It does so by making the compared nodes of the derivation tree necessarily sisters. The comparison can then be represented in a single elementary tree using a finite number of features. The proof goes as follows:

Step 1: Construction of a STAG *G'* (where there are no MLSNs) from a STAG with MLSNs *G*: Consider all of the elementary trees in the grammar. In order to convert all trees with MLSNs into trees without them, the following step can be taken: For each node with n links (n>1) create a tree for each possible ordering of the links. The ordering is represented by the dominance relation on a single-branching tree. Replace nodes with multiple links with the representations of these ordered trees. This would, for instance take a node $t_{1,2}$ and convert it into two different structures where a t-node dominates another t-node. One tree would have the 1 link dominate the 2 link and another where the 2 link dominates the 1 link. Trees with these structures in them replace the original trees where there are nodes with multiple links.

For n links the number of additional trees is $n!$. For a tree with multiple instances of MLSNs ($m_1 ... m_n$) the number of additional trees is the product $m_1!... \times ...m_n!$. This, at first glance, is problematic because as n grows, the factorial growth of $n!$ exceeds even the exponential growth rate. This is not problematic for the compilation because of the natural bound of links for a given node. The maximum number of links for a semantic node is the number of nodes in the syntax and since this is finitely bounded, the increase in trees is at worst still manageable. In any actual case, the results will be easier: it is reasonable to think that the maximum number of links for a t node is the number of arguments of the verb plus 1 (for sentence level modifiers.) In conclusion, since there is a finite bound, there is no particular problem with the factorial growth rate.

Step 2: Another necessary step in the conversion is to do the following: Take (scopal) nodes (t-nodes) and make an ordering of their links. If the order is a partial order take the total order extensions of the ordering. Replace the partial ordered trees with their total ordered extensions. For instance, if we have a series of nodes t_1-t_2-t_1, we would end up with t_1-t_2 and t_2-t_1. This step is necessary for future steps where trees are eliminated. Trees where a link is associated with a node that both dominates and is dominated by another link underspecifies the scope relation between different scope taking operators. Since it is necessary to make the scope unambiguous for each elementary tree (in order to remove scope configurations that are unwanted) these trees must be removed from the grammar. At this point, we have constructed separate trees that corresponds to every scope ordering that the linkages allow; in essence this grammar will allow any quantifier to use split combination.

Step 3: Now we have to replicate the effects of the DCCSI. Take the grammar created by step 2 and then remove the tree set types that correspond to the readings that are made unavailable by the DCCSI.

First, we add features that constrain what type of quantifier can adjoin to what DP position of a verbal tree to relevant nodes of trees. In order to get the results described in this paper, for instance, it suffices to only have a +/- quantifier feature, a distributive quantifier feature, and a +/- negative feature. But no matter the actual number of features needed for a complete analysis, it will be finitely bounded.

If the number of features were not stipulated to be finite, the addition of features would exponentially increase the number of trees in the grammar. The addition of trees in general (while also only considering features on arguments) adds $(2^n)^k$ trees for each tree in the grammar, where n is the number of arguments and k is the number of features.

The nodes of a tree in the worst case would all have the maximal set of features and there would be all of the possible combinations. Since the number of nodes causes exponential growth in the size of the grammar this would be problematic if the number of nodes in the grammar weren't also bounded. Since all trees in TAG are bounded, they must also have a finite number of nodes. Thus,

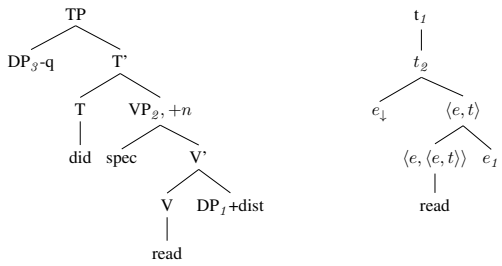

Figure 15: Example of a tree set to be removed to replicate the effects of the DCCSI. This tree-pair would allow a distributive quantifier in object position to take wide scope over clausal negation.

the finite number of features and nodes allows the added number of trees to be manageable.

In figure 15, an example is given; it corresponds to the *not...every* sentences that can not have a wide scope universal reading. Once all of the appropriate tree sets are removed, the resultant grammar is *G'* which is equivalent to G.

This construction has shown how a grammar with a TDC can be a TAL when the comparison class for the transderivational constraint is constrained. Additionally, it was suggested that the growth during the compilation does not increase the number of trees disastrously if some reasonable assumptions are made. For these reasons, it seems that the use of semantic TDCs are not infeasible in grammar when properly constrained.

References

F. Beghelli and T. Stowell. *The Syntax of Each and Every*, pages 71–107. Springer, 1997.

B. Bruening. Qr obeys superiority: Frozen scope and acd. *Linguistic Inquiry*, 32(2):233–273, 2001.

D. Fox. Economy and Scope. *Natural Language Semantics*, 3(3):283–341, 1995. ISSN 0925-854X.

D. Fox. *Economy and Semantic Interpretation*. The MIT Press, 2000. ISBN 0262561212.

R. Frank. *Phrase Structure Composition and Syntactic Dependencies*. The MIT Press, 2002.

M. Freedman and R. Frank. Restricting inverse scope in synchronous tree-adjoining grammar. In *Proceedings of the Tenth International Workshop on Tree Adjoining Grammars and Related Formalisms*, 2010.

P. Grice. *Studies in the Way of Words*. Harvard Univ Pr, 1989.

C.H. Han. Pied-piping in relative clauses: Syntax and compositional semantics based on synchronous tree adjoining grammar. In *Proceedings of the Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms*, pages 41–48. Association for Computational Linguistics, 2006.

I. Heim. Degree operators and scope. *audiatur vox sapientiae. a festschrift for arnim von stechow*, pages 214–239, 2001.

L.R. Horn. *A natural history of negation*. University of Chicago Press, 1989.

R.K. Larson. On the double object construction. *Linguistic inquiry*, 19(3):335–391, 1988.

A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *13th Annual Symposium on Switching and Automata Theory*, pages 125–129. IEEE, 1972.

R. Nesson and S. Shieber. Extraction phenomena in synchronous TAG syntax and semantics. In *Proceedings of the NAACL-HLT 2007/AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 9–16. Association for Computational Linguistics, 2007.

Y. Schabes and S. Shieber. An alternative conception of tree adjoining derivation. *Computational Linguistics*, 20:91–124, 1994.

S. Shieber and Y. Schabes. Synchronous Tree-adjoining Grammars. In *Proceedings of the 13th conference on Computational linguistics-Volume 3*, pages 253–258. Association for Computational Linguistics Morristown, NJ, USA, 1990.