1 Introduction

The notion that quantifiers take scope has led researchers to devise a number of mechanisms to account for quantifiers in natural language. The additional fact that structural position does not uniquely determine the relative scope of scope-taking operators has caused these mechanisms to allow quantifiers to scope freely with respect to one another (within a clause). Yet, it has been observed that there are many instances where the relative scope of quantifiers is determined: while $\forall$-quantifiers in object position are able to take scope over subject quantifiers (ex. (1a)), they are unable to take scope over clausal negation (ex. (1b)) (see [1]). $\exists$-quantifiers, such as more than 2, are unable to scope over either subject quantifiers (ex. (1c)) or negation\(^1\) (ex. (1d)). I propose an analysis that can easily account for such cases.

(1) (a) A student read every paper. ($\forall > \exists, \exists > \forall$)  
(b) John didn’t read every/each paper. ($\forall \not> \neg, \neg > \forall$)  
(c) Every student read more than 2 papers. ($\forall > 2+, 2+ \not> \forall$)  
(d) John didn’t read more that 2 papers. ($2+ \not> \neg, \neg > 2+$)

In this paper, I explore a semantically-oriented approach to the data in (1) and related data, with the goal to better understand why the restrictions occur where they do. Specifically, I show that two general principles can account for the data in (1) and extensions thereof. First, I argue that classes of quantifiers differ in the flexibility of their derivationally-constructed syntax-semantics mapping. Second, I argue that interpretive possibilities can be made unavailable through competition from alternative derivations that provide the same interpretation, but are more simply derived.

I implement these principles in synchronous tree adjoining grammar [12], a version of TAG (defined in §2) where the derivation of semantic trees parallels the syntactic derivation. The extended domain of locality that is expressible in TAG plays a key role in this analysis. The elementary objects in TAG are larger than words, and includes structure which corresponds to the domain over which

\(^1\) I put aside here other existential quantifiers like bare numerals, which are able to scope over a subject quantifier or negation. I assume that these types of quantifiers acquire scope via a pragmatic mechanism, following [11].
a word imposes grammatical restrictions [4]. This extended domain of locality allows the two principles that are proposed in this paper to be stated in a natural and computationally feasible manner.

The remainder of the paper is arranged as follows: In §2, I give an introduction to tree-adjoining grammar. In §3, I show how to generate surface scope and inverse scope using this system. I invoke mechanisms that make inverse scope a more costly operation derivationally. In §4, I give an analysis where transderivational economy can account for certain instances of frozen scope. I show how locality principles inherent to TAG limit the range of derivational competition in an empirically supported manner in §5. In §6, I prove that the variety of TAG I propose in this paper is no more powerful than a STAG with a finite number of features.

2 Introduction to Tree-adjoining Grammars

A tree-adjoining grammar (TAG) consists of a set of trees, called elementary trees. Elementary trees are combined to produce larger trees through two operations: substitution and adjunction. Substitution is the replacing of a leaf of a tree with a new tree. Adjunction is the replacing of an internal node of a tree with a new tree. The tree that adjoins into a tree is called an auxiliary tree. Auxiliary trees must have their root node and one of their leaf nodes match in type. The leaf node is called a foot node. The location of adjunction takes place at a node that matches the root and foot nodes. TAGs are a variety of mildly context-sensitive grammars [8]. An example of substitution and adjunction is in figure 1. The far left tree in figure 1 is the derivation tree. Daughter nodes of the derivation tree represent trees that combine into their parent node through either substitution or adjunction.

Following Shieber and Schabes [12], I assume that two or more auxiliary trees can adjoin into the same node (multiple adjunction). I adopt the assumption
that the later introduced tree adjoins on top of the earlier adjoined tree [12].
An example of multiple adjunction is in figure 2. In the derivation tree on the
left, the numerical subscripts represent the order of combination. Because stupid
adjoins first, it is below big in the final derived tree.

Fig. 2. This is an example of multiple adjunction. On the far left is the derivation
tree. The numerical subscripts represent the order of combination. In the middle is the
elementary trees for big, stupid, and phone. The arrows demonstrate where big and
stupid adjoin into phone. On the right is the final derived tree.

Synchronous TAG (STAG) extends the elementary structures of a TAG
(initial and auxiliary trees) into pairs of trees whose nodes are connected by
links [12, 9]. For this paper, one tree represents the syntax and the other tree
represents the logical form. The derivation proceeds synchronously, where an
operation in the syntax corresponds to an operation in the semantics. Linkages
in an elementary tree set make correspondences between how trees combine into
the set in both the syntax and the semantics. An example is in figure 3: the syntax
and semantics trees for john and mary combine into the syntax and semantics
trees for loves. The numerical linkages between the syntax and semantics trees
for loves guarantees the correct connection between the syntax and semantics.

An Multi-component TAG (MC-TAG) consists of sets of trees that
can combine into another tree [7]. Intuitively, MCTAGs represent the idea that
a lexical item can consist of more than one part. We assume a Tree-local MC-
TAG: a tree set must combine into the same elementary tree.

Within the TAG tradition, quantifiers have often been construed as being
a multi-component set [7, 12]. An example is in figure 4: One tree represents
the scope of the quantifier and the other tree fills the argument position. They
both combine into the same tree. The derivation of (1a), depicted in figure 5,
provides an example of quantification with two nominal quantifiers: First, the
object-tree (every book) combines with the verb-tree (read) via the substitution
operation. In the semantic derivation, the scope tree and the argument tree of the
Fig. 3. This is an example of STAG. On the left, is the derivation tree; on the right, the elementary trees; and on the bottom is the final derived trees.

quantifier combine into the verb tree via substitution and adjoining, respectively. Then, the subject-tree (a student) substitutes into the verb-tree. In the semantic derivation, the scopal tree and the argument tree combine into the verb-tree again via substitution and adjoining. The properties of multiple adjunction [12] result in the subject scope tree appearing above the scope tree for the object quantifier. This produces the desired surface scope reading. The inverse scope reading could be derived if the subject combines first with the verb and then the object.

3 Deriving surface and inverse scope

3.1 Deriving surface scope through restrictions on derivation

One generalization from the data in (1) is that surface scope is generally available, while inverse scope is more limited in its occurrence. In this section, I devise a mechanism to restrict the TAG derivation so that only surface scope interpretations are available. I propose the following restriction on the ordering of the derivation (which is traditionally thought to be unordered):

Definition 1. Prominence Restriction on Derivation (PRoD): If node a in syntactic tree T is targeted prior to node b in T, then a must not irreflexively dominate or asymmetrically c-command b.

PRoD enforces the idea that a derivation proceeds bottom up. This makes the derivation procedure similar to other frameworks like minimalist grammars.
Fig. 4. This is an example of MC-TAG. The semantic portion of the quantifier consists of two parts: the scope-tree adjoins to the top t-node and the argument-tree substitutes into the argument slot.

where bottom-up derivation is mandatory. By PRoD, one could not substitute into the subject position before the object position. Then, because of our assumptions on multiple adjunction, the subject quantifier will always scope over the object quantifier. But, given PRoD, inverse scope should be unavailable generally.

3.2 Deriving inverse scope through delayed combination

Of course we need to allow for cases of inverse scope in cases like (1a). To handle such cases of inverse scope, I propose that a tree set can specify derivational restrictions (contra [7]) on its use during a derivation. I assume that the substitution of the variable trees in the semantic derivation must take place at the time of the substitution of the syntactic NPs, following the hierarchical order. The scopal portion of the semantic tree set may follow one of the following regimens:

Definition 2. Simultaneous combination (SC): the integration of the trees within a tree-set must take a single point in a derivation (SC is the assumed method of derivation in most TAG analyses.).

Definition 3. Delayed combination (DC): the integration of the trees within a tree-set may take place at different points during the derivation but must occur tree-locally.

Without multiple adjunction, this difference has no effect. With multiple adjunction however, there is an effect. Delay allows a less prominent quantifier to scope over a more prominent quantifier by coming into the derivation later.

I argue that the different classes of quantifiers differ along the SC/DC dimension: the tree-set of ∀-quantifiers are DC, while the other QP classes are SC. So, in order to get inverse scope in cases like (1a), the DC property of the ∀-quantifier tree is exploited. The scope portion of the ∀-quantifier is not adjoined immediately. Instead, the derivation proceeds through the integration of the subject quantifier, including its scope, and then the scope of the object quantifier is
Fig. 5. This is an example of quantification in MC-STAG. On the left, there are the two possible derivation trees, with either quantifier combining first. On the right are the elementary trees. The top portions show how the semantic portions combine in with both DP-trees substituting into the verb-tree. On the bottom is the semantics. The scope portion of the quantifiers adjoin at the t-node of the read-tree. The argument-trees substitute into the read-tree as well.

finally adjoined. This yields wide scope for the object quantifier. The derivation tree for this can be seen in figure 8 where the derivation tree shows the three steps of the derivation.

4 Transderivational Competition

At this point there seems to be a puzzle with respect to the data in (1): why is the object $\forall$-quantifier unable to scope over clausal negation if it is DC?

Notice that the example in (2) has an interpretation that is logically equivalent to the unavailable reading in (1b), as $\forall\neg$ is equivalent to $\neg\exists$.

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

The examples in (2) under this interpretation are more simply derived than (1b) in that they take fewer steps. Explicitly, as depicted in the derivation trees in figure 7, (1b) takes four steps: (i) the object combines with the verb-tree (without the scope tree for the quantifier combining in the semantic derivation); (ii) the negation combines into the verb-tree; (iii) the subject combines into the
verb tree; (iv) the scope-tree for the object quantifier combines into the verb-tree. The derivation for (2) only takes three steps because the scope tree does not combine in separately. Following this observation, I propose the following:

**Definition 4. Derivational Complexity Constraint on Semantic Interpretation**: A derivation \( d \) producing syntax \( s \) and meaning \( m \) is ruled out if: (1) another derivation \( d' \), which takes fewer steps, also produces \( m \) and (2) there is a derivation \( d'' \) such that \( d'' \) has syntax \( s \) but has a meaning \( m' \) distinct from \( m \).

Here we tentatively define meaning \( m \) as being truth-conditional meaning and information structure content. So, given the definition in the DCCSI, we can see how the blocking works: the derivation for (2) is shorter than the derivation for (1b), the meanings for (2) and (1b) are the same, and (1b) has a different derivation that produces the same syntax but a different meaning (\( \neg \forall / \neg \exists \)).

Given the definition of the DCCSI, it is not always the case that a more complex derivation is blocked by a simpler one: Many apparent comparisons between sentences lead to wrong predictions. For instance, a passive voiced sentence should not compete with its active counterpart because they do not match in information structure; they do not produce the same meaning \( m \). Also, a sentence like “no one didn’t arrive” is not blocked by “everyone arrived” because of the former’s lack of semantic ambiguity; it does not have a meaning \( m' \) distinct from \( m \).

Additional evidence for the DCCSI stems from the availability of the \( \forall \exists \neg \) reading for the sentence “someone didn’t read every book”. No blocking occurs because there is no suitable competition: (i) another quantifier cannot replace *every* and produce the intended meaning and (ii) there are no quantifiers that
lexically encode two of the adjacent quantifiers/operators. This is surprising because of the cases where the universal quantifier seems to be unable to scope over negation. This provides an argument for why the grammar should not just disallow universal quantifiers scoping over negation.

An apparent problem for the analysis we are exploring is that the surface scope seems to be unavailable for the sentence in (3). Surface scope should always be available given PRoD, surface scope is not available in this case.

(3) Every student didn’t read Huck Finn.

The puzzle is straightforwardly solved by the DCCSI however: (3) cannot get the $\forall \neg$ reading because it is blocked by (4), as depicted in figure 8.

(4) No student read the book. ($\neg \exists$)

![Fig. 8. Derivation trees for “Every student didn’t read Huck Finn” (top) and “No student read the book” (bottom) both with the $\forall \neg$/$\neg \exists$ reading.]

Again, the derivation producing the meaning $\neg \exists$ for (4) is shorter than the derivation producing the meaning $\neg \forall$ for (3), the meaning of both sentences is the same, and (3) has another meaning ($\neg \forall$) that can be derived from the same syntax. Specifically, the derivation for (3) takes three steps: (i) the object combines into the read-tree; (ii) the negation combines into the read-tree; (iii) the subject combines into the read-tree. The derivation for (4) takes two steps: (i) the object combines into the read-tree; (ii) the subject combines into the read-tree. The shortened derivation for (4) is due to the negative quantifier no providing both scopal elements; both $\neg$ and $\exists$ are part of one elementary tree.

Thus, the mode of comparison employed by the DCCSI is able to account for the puzzling data from (1) given the analysis in §3; namely why surface scope isn’t available in Every...not sentences and inverse scope isn’t available in not...every sentences.

5 Locality and Comparison

The notion of comparison presented in this paper is a very old one [5] but it has resisted a precise formulation. Additionally, there are computational reasons to think that economy constraints are too powerful to begin with [6]; and if
they are available, the comparison class has to be highly restricted \[2, 3\]. This section shows that we can limit the comparison class in a way that will not be problematic with respect to the expressive power of the grammar.

The DCCSI can be understood as a local constraint on TAG derivations in conjunction with the specific mechanics of the TAG formalism. I define a locality of comparison constraint in (5).

**Definition 5. Locality of Comparison Constraint (LCC)**: A derivation \( d \) is only compared to distinct derivation \( d' \) if the derivation trees are identical except for the leaf nodes of the same parent node.

The LCC consists of two parts: (1) the quantificational elements being configured are sisters in the derivation tree; and (2) that the sisters have no dependencies in the derivation tree. The first part is necessary because it allows all of the quantificational information to be localized on a single elementary tree (see §6). The second part adheres with some modification to the TAG notion of context freeness of derivation; i.e. each derivational step must be insensitive to the previous derivational history \[4\]. I slightly diverge from the spirit of that requirement to allow that at a particular derivational step it can be known whether or not the trees combining in are complex or elementary. Otherwise, the derivation is blind to the content of previous steps, as is standard (this can be done without formally making the derivation context sensitive). The result of the blindness is that any tree that adjoins into a DP will not be visible when the DP combines into the verb-tree (as schematized in figure 9d,e,f), and since the comparison mechanism cannot know if the modified material matches between any two derivations, a comparison cannot be made. The definition of locality used also makes a number of empirical predictions; the DCCSI should only hold in certain configurations and in other configurations “blocking” should not be in effect. For instance, sentences with nominal modification should render the DCCSI unusable, as in figure 9a compared to 9c. The modifying PP, which adjoins into the DP, makes it so that the relevant derivations for “every dinosaur that flew didn’t survive the meteor shower” and “no dinosaur that flew survived the meteor shower” (on the wide scope universal reading) incomparable because the quantifier-trees have dependents. This predicts a wide scope universal reading for “every dinosaur that flew didn’t survive the meteor shower” and this does seem to be the case. Of course, more empirical work needs to be done in this area.

6 Complexity of Formalism

This section attempts to show that the system proposed in this paper is no more computationally powerful than standard tree-adjoining grammar through a sketch of a proof. I do this through a method for converting the formalism described in this paper into a standard TAG with features that is known to be mildly-context sensitive. The translation takes three steps: (1) recreate the effects of multiple adjunction using the formalism outlined in this paper but without
(a) every dinosaur that flew didn’t... (b) No dinosaur that swam... (c) No dinosaur that flew didn’t; and what contact can be seen at one step of the derivation in (d), (e), (f).

Multiple adjunction; (2) recreate the effects of PRoD and delayed combination using the grammar created in step 1 without PRoD and delayed combination and with the overt addition of features; (3) recreate the effects of the DCCSI with the grammar created in step 2 and no DCCSI. This will show that the grammar created in this paper is no more powerful than a STAG with a finite number of features.

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

Proposition 1. Every STAG with multiple adjunction, delayed combination, and the DCCSI, can be transformed into an equivalent synchronous multi-component feature tag without multiple adjunction, delayed combination, and DCCSI (See [10] for the properties of these types of grammars).

Proof. Construction of a STAG $G'$ from a STAG with multiple adjunction, delayed combination, and DCCSI $G$. We start out with the elementary trees of $G$ and proceed to convert them systematically to STAG trees.

Step 1: Consider all of the elementary trees in the grammar. In order to get rid of the relevant multiple adjunction, we do the following: For each node with $n$ links $(n > 1)$ (this corresponds to the nodes in the semantics trees where multiple adjunction can take place) make an unordered set $\{\alpha_1...\alpha_n\}$. Then, make totally ordered sets for each permutation of the sets’ members. For each ordered set, represent it as a single-branching tree where order is represented by the dominance relation. Replace nodes with multiple links with the representations of the ordered sets. This would, for instance take a node $t_{1,2}$ and convert it into two different structures where a t-node dominates a t-node. One where the 1
link dominates 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.

Next, 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 $t_2-t_1$. 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 “delay”.

**Fig. 10.** Tree set that is to be removed to convert a grammar with delayed combination to one without it. Without removing these types of tree sets, any quantifier could in essence “delay”.

**Step 2:** Now we have to convert the grammar from one which in essence allows any quantifier to “delay” to one that replicates *delayed combination* only for the cases where it is warranted. 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 +/- modified feature. Then, we do the following to make an equivalent grammar that does not have unselective “delay”: we remove a tree-set from the grammar if:

1. a syntactic node $\beta$ asymmetrically c-commands or dominates a syntactic node $\alpha$;
2. $\alpha$ does not have a feature that allows delay ([dist]) and
3. the semantic node linked to $\alpha$ dominates the semantic node linked to $\beta$.

The type of trees that are to be removed are in figure 10. At this point, we have constructed a STAG that is equivalent to a STAG + multiple adjunction + PRoD + DC. All that needs to be done now is recreate the DCCSI to eliminate the generation of the unwanted strings.

**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. In figure 11, an example is given: it corresponds to the 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.
Fig. 11. Example of tree sets to be removed to replicate the effects of the DCCSI. This tree-pair would allow a non-distributive quantifier in object position to take wide scope over another quantifier. Without removing these types of tree pairs, any quantifier could in essence “delay”.

7 Conclusion

In this paper, I have shown that we can account for certain scope asymmetries by lexically differentiating the combinatorial possibilities of different classes of quantifiers. And that other asymmetries can be handled by transderivational competition. More work needs to be done to both expand the empirical coverage and determine the computational power of the proposal.

References