Cosubstitution, derivational locality, and quantifier scope*

Chris Barker
New York University
chris.barker@nyu.edu

Abstract

Quantifier scope challenges the mantra of Tree Adjoining Grammar (TAG) that all syntactic dependencies are local once syntactic recursion has been factored out. The reason is that on current TAG analyses, a quantifier and the furthest reaches of its scope domain are in general not part of any (unicomponent) elementary tree. In this paper, I consider a novel basic TAG operation called COSUBSTITUTION. In normal substitution, the root of one tree (the argument) replaces a matching non-terminal on the frontier of another tree (the functor). In cosubstitution, the syntactic result is the same, leaving weak and strong generative capacity unchanged, but the derivational and semantic roles are reversed: the embedded subtree is viewed as the functor, and the embedding matrix is viewed as its semantic argument, i.e., as its nuclear scope. On this view, a quantifier taking scope amounts to entering a derivation at the exact moment that its nuclear scope has been constructed. Thus the relationship of a quantifier and its scope is constrained by DERIVATIONAL LOCALITY rather than by elementary-tree locality.

1 Introduction

The main claim of the present paper is that among the major grammatical frameworks, Tree Adjoining Grammar (TAG) offers a uniquely simple and direct way to understand the relationship between a quantifier and its scope.

*Thanks to Robert Frank and Chung-chieh Shan.

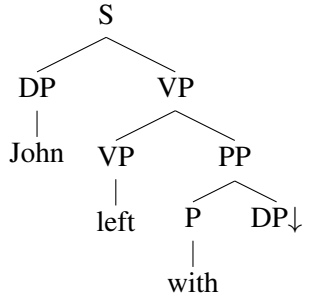
There are at least two well-developed approaches to scope in TAG. One is due to Kallmeyer and Romero and their collaborators (e.g., Joshi, Kallmeyer and Romero 2008). They use Multi-Component TAG (MC-TAG), and emphasize semantic underspecification, so that a single derivation corresponds to multiple quantifier scope construals. Another approach due to Nesson and Shieber (e.g., 2006) uses a Synchronous TAG to relegate the multi-component part to the semantic side of the derivation.

The view here is not intended as a competing approach, so much as a competing perspective on what other accounts are already doing—an alternative conceptualization. However, there are some differences between the analyses that I will mention below.

The basic idea here relies on flexibility in the order in which the components of a TAG derivation combine. That is, substitutions and adjunctions can be interleaved and reordered with considerable freedom. This flexibility makes it possible to build a partial derivation that exactly corresponds to the material over which a quantifier takes scope:

(1) a. John left with no one.

b.



Normally, we might choose to substitute *no one* into

the auxiliary tree projected by *with*, and then adjoin the derived tree *with no one* at the VP node dominating *left*. But we might just as well adjoin the incomplete auxiliary tree first, resulting in the (still incomplete) tree in (1b). The point of interest is that this derivational constituent corresponds in a natural way to the nuclear scope of *no one*: just abstract over the substitution location to get the property λx.John-left-with x.

Thus the reason that managing scope in TAG is a challenge is that quantifiers and their scope domain are not local in the usual TAG sense. That is, it is not possible to factor out recursion in such a way that the quantifier and its scope are safely included within a single elementary tree. For instance, in (1), the quantifier never shares an elementary tree with the S node it take scope over.

Yet although quantifiers and their scope are not elementary-tree local, quantifiers and their scope are never discontinuous. At the end of a derivation, if we shade in the portion of the tree that corresponds to the material a quantifier takes scope over, it will always be a contiguous portion of the tree, and in addition, it will also immediately dominate (in general, surround) the quantifier.

Making sense out of the derivational approach considered here requires rethinking the tree-merging operation that combines the quantificational DP *no one* with its nuclear scope. Instead of regarding the quantifier DP as plugging a hole in the argument structure of *with*, we would like to reverse the roles, and think of the incomplete tree in (1b) as the semantic argument of the quantifier. Call this desired operation COSUBSTITUTION (details below).

If we allow cosubstitution as a basic TAG operation, we recognize quantificational scope as an example of a different kind of local dependency, namely, the dependence of a functor on its (co)substitution argument. The result is that we need to recognize two kinds of locality: structural locality, i.e., sharing the same elementary tree, and derivational locality, participating in the same derivational step.

The late substitution contemplated in (1) would not be innocent in a Multi-Component TAG. Allowing one component of a tree set to substitute into the lower DP position in (1) at the same time that another element (think: the scope-taking part) adjoins into the original initial tree is non-local, and allowing such non-local operations in MC-TAG increases its generative capacity. Therefore it's important that I'm considering ordinary TAG here, not MC-TAG. In some sense, of course, all analyses of quantifier scope are an attempt to simulate just this kind of non-local operation, as discussed further below.

Treating scope-taking as cosubstitution is a version of the continuation-based approaches to scope-taking of Barker 2002, de Groote 2001, and Bernardi and Moortgat 2010, among others. A continuation is (a portion of) the computational future of an expression. In (1), the computational future of the quantifier *no one* is that it will serve as the argument of the preposition *with*, and the result of that computation will serve to modify the verb phrase *left*, and so on. The central insight I'm aiming for in this paper is that in TAG, the computational future of a DP can be viewed as the same thing as its derivational past.

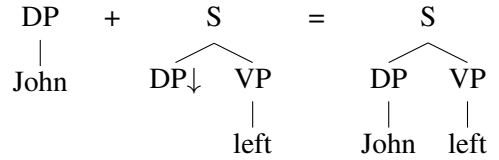
2 Preliminaries

2.1 Syntax

A Tree Adjoining Grammar is a finite set of elementary trees closed under two derivational operations: substitution and adjunction.

Elementary trees are finite ordered labeled trees. Nonterminals on the frontier of an elementary tree are substitution targets, and are decorated with a downarrow. Some elementary trees have a distinguished node on their frontiers called the FOOT (marked with a star) that match the root node in syntactic category. Such trees are auxiliary trees, and participate in adjunction.

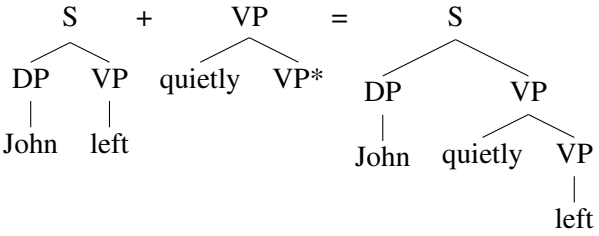
Substitution: Nodes with downarrows on their labels can be replaced via substitution with any non-auxiliary tree whose root node has a matching label. The substitution operation amounts to replacing the target node with the root of the substitution tree.

(2)


Adjunction: Interior nodes whose labels match the root label of an auxiliary tree can be adjunction targets. Adjunction is accomplished by replacing the adjunction target node with the root of the auxiliary

tree, at the same time that the foot of the adjunction tree is replaced by the subtree rooted in the adjunction target node. In effect, the auxiliary tree is inserted into the tree at the adjunction target node.

(3)



This is the familiar TAG story, simple and elegant. Technical details are available in may places, e.g., Joshi and Schabes 1997.

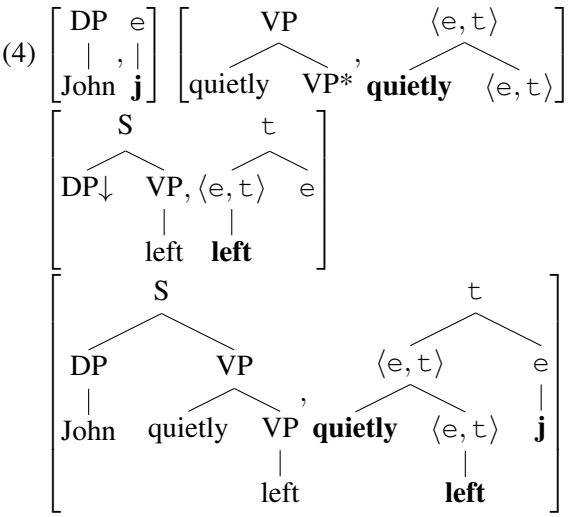
2.2 Semantics

I will use a Synchronous TAG (Shieber and Schabes 1990) to specify semantic representations. Instead of elementary trees, STAG uses pairs of elementary trees connected by a linking relation. Any operation targeting a node in the left element of a pair must be matched by a parallel operation targeting the linked node in the right element of the pair.

In general, then, STAG is a tree transduction system. Here, as in Nesson and Shieber 2006, each pair will be interpreted as the syntax and the corresponding semantics for an expression. The syntactic component will use syntactic categories for labels, and the semantic component will use semantic types for labels.

So for [syntax, semantics] pairs we might have:

(4)



Not much happens in this transduction, except that the compositional order of the VP and the subject are reversed to conform to the conventions for function/argument order in the lambda calculus. Throughout the paper, I've left the linking relation between syntactic nodes and semantic nodes implicit, since the intended relation is particularly simple and, I hope, obvious.

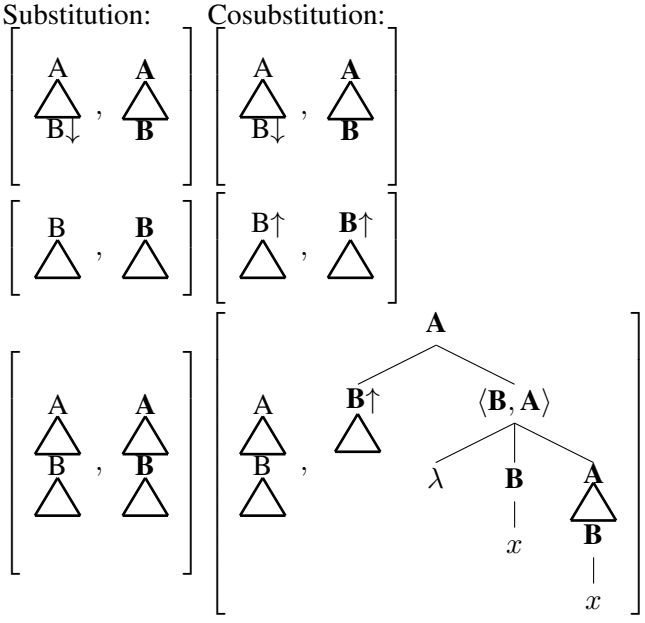
3 Cosubstitution

The basic idea of using cosubstitution to handle scope is that we can build the nuclear scope of a quantifier before the quantifier enters the derivation.

In the normal substitution case, we have a tree t_1 containing a substitution target, that is, a node x whose label B is decorated with a downarrow. We also have a separate tree t_2 whose root r has a matching label, B. We replace x with r, and the tree rooted in r becomes a subtree of t_1 (first column of (5)).

In cosubstitution, we reverse the roles: now t_2 contains the (co)substitution target, (which can only be) the root node r. In recognition that the root is now a cosubstitution target, we annotate its label with an uparrow. As long as t_1 contains a frontier node x with a matching label (matching except that it is still decorated with a downarrow rather than an uparrow), cosubstitution may occur. Conceptually, we replace (only!) the target node r with x, and the tree footed in x becomes a supertree of t_2. (So the operation probably should be called "superstitution".)

(5)



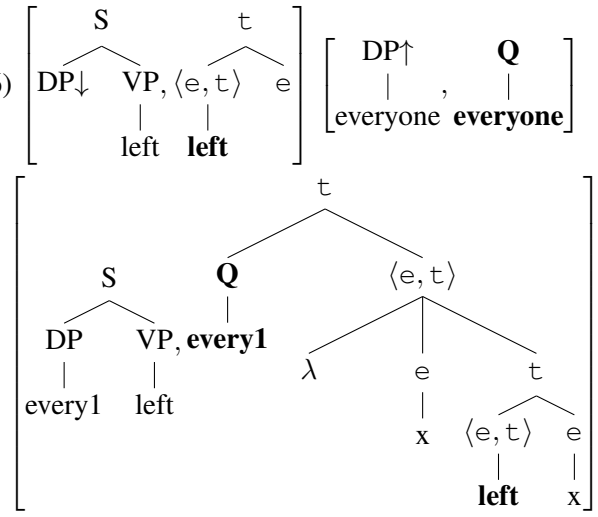
In the diagrams, bolded symbols (e.g., **A**) represent the semantic type of the corresponding syntactic category (A). In general, **A**↑ will be $\langle\langle \mathbf{A}, \mathtt{t}\rangle, \mathtt{t}\rangle$; in particular, **DP**↑ = $\langle\langle \mathtt{e}, \mathtt{t}\rangle, \mathtt{t}\rangle$, the type of a generalized quantifier. I'll use **Q** as shorthand for $\langle\langle \mathtt{e}, \mathtt{t}\rangle, \mathtt{t}\rangle$.

Note that the syntactic trees after substitution and after cosubstitution are identical. There is a difference in the semantics, however, since we must abstract over the substitution argument expression.

As usual (e.g., Nesson and Shieber 2006:5) the abstraction variable (in the diagram, x) should be chosen fresh with respect to all previous choices of abstraction variables in the derivation, though it may be identical with the translation of pronouns in order to accomplish quantificational binding.

In the simplest case, the cosubstitution argument is the lexical projection of a predicate.

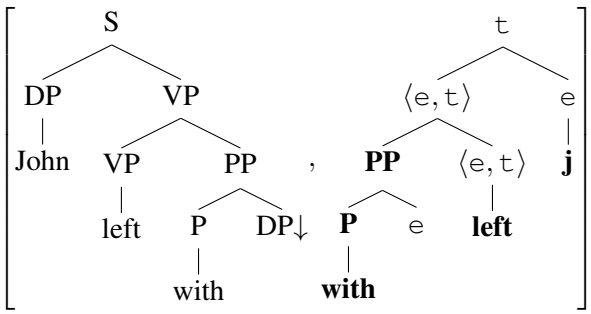
(6)



If **everyone** $= \lambda P \forall x.(Px)$, then the semantics for *everyone left* beta-reduces to $\forall x.(\mathbf{left}\ x)$.

In general, however, nuclear scopes can be complex, as in (1):

(7)



If we cosubstitute (7) onto the quantifier *no one*, we get **no one**$(\lambda x.(\mathbf{with}\ x\ \mathbf{left})\ \mathbf{j})$.

Thus the dependence of *no one* on its nuclear scope is not local in the usual sense of forming part of the same elementary tree. However, it is derivationally local: the quantifier and its nuclear scope are the two participants in a single derivational step, a cosubstitution step.

4 Inverse scope

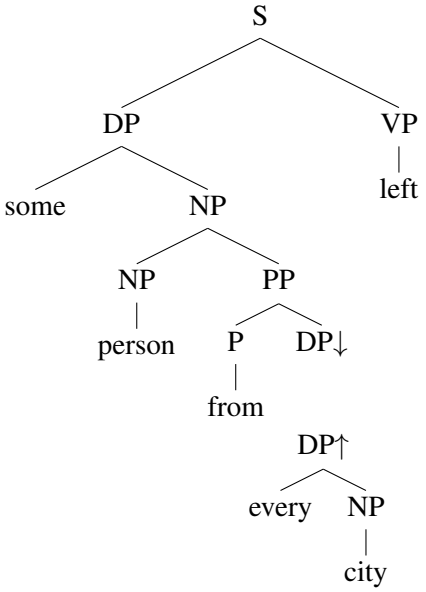
As in previous TAG work, inverse scope makes a good test case for illustrating how the system works.

(8) Some person from every city left.

Near the beginning of the derivation, *some person* undergoes cosubstitution with *left*, essentially as in (6). After adjunction with *from* at the NP node, we have:

(9) a. Some person from every city left.

b.



At this point, we have a cosubstitution opportunity for *every city* in which its scope corresponds to *Some person from __ left*.

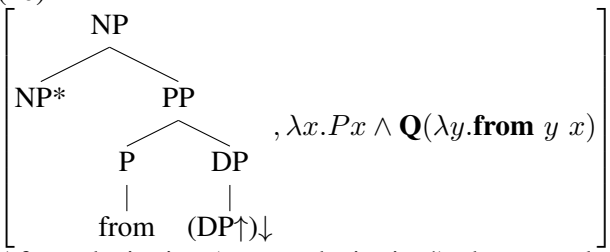
In other words, the nuclear scope of a quantifier is quite simply and quite literally its syntactic and semantic argument.

The fact that the embedded quantifier has the entire verb phrase in its scope explains how it can bind a pronoun in the verb phrase, as in *Someone from every$_i$ city loves it$_i$*.

Getting the opposite scoping requires supposing that prepositions project structure at which a quantifier can take scope internal to the enclosing DP. This strategy is used by May 1985, Heim and Kratzer 1998, Barker and Shan 2006, among others, and, in the TAG literature, by Nesson and Shieber 2006. For

the sake of concreteness, we can approximate this analysis (more work is needed to make it fully general) by allowing *from* to take a quantificational DP as its argument:

(10)



After substitution (not cosubstitution!), the generalized quantifier denoted by *every city* substitutes for the semantic non-terminal **Q**.

One advantage of this strategy is that when *some* takes scope over *every*, *every* has only the prepositional phrase in its scope, so it can't bind a pronoun in the verb phrase (this agrees with the facts).

A second advantage of this strategy is that whenever the scope of the embedded quantifier is trapped inside of the larger DP, there is no way for quantifiers external to the DP to intervene in scope between *some* and *every*.

(11) Two politicians spied on some person from every city.

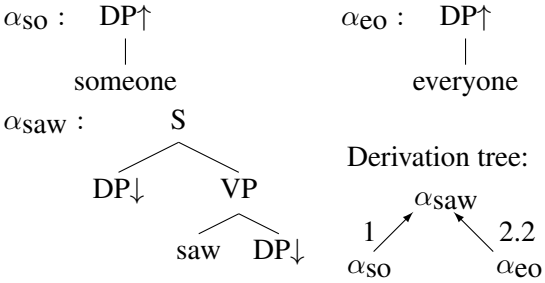
The truth conditions of such a reading for (11) would require that there be some specific person from every city, a single pair of politicians, and at least one spying event for each city. As discussed in detail in Joshi, Kallmeyer and Romero 2008 and in Nesson and Shieber, there is a general consensus that this scoping should be ruled out by the grammar.

However, the system here does allow external quantifiers to intervene in scope between *some* and *every* on the inverse linking reading. That is, it is possible to cosubstitute the *spied on* tree onto *some person*, then cosubstitute the result onto *two politicians*, then cosubstitute the result of that operation onto *every city*, giving as a scoping *every* > *two* > *some*. As Nesson and Shieber note, there is less of a consensus on whether this reading should be considered ungrammatical. My position is that it is grammatical, but unusually hard to process.

5 Scope ambiguity and derivation trees

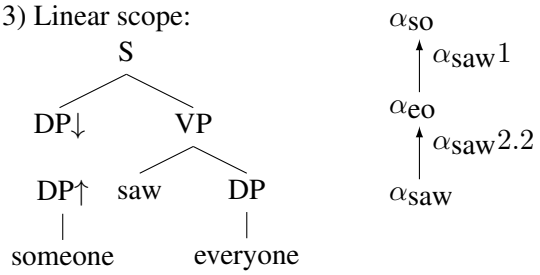
Thus on the cosubstitution approach, quantifier scope ambiguity is a matter of timing: quantifiers that enter later in the derivation take wider scope.

(12) Someone saw everyone.

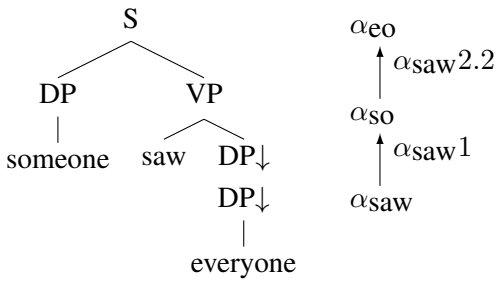


As Nesson and Shieber point out, if we use the usual kind of derivation tree, we end up with a derivation tree that does not disambiguate scope. In the derivation tree in (12), I have even added arrowheads to distinguish substitution (downward-pointing arrowhead) from cosubstitution (upward), but the derivation tree still does not reveal which quantifier takes wide scope.

(13) Linear scope:



(14) Inverse scope:



But if we take the conceptualization of cosubstitution seriously, we have to treat the derived scope as the argument of the quantifier. That means that the quantifier conceptually is the target for the cosubstitution. The correct derivation trees, then, have the quantifier dominating their nuclear scope.

These derivation trees contain full information about the order in which the quantifiers were added to the derivation, and which takes wider scope.

Usually, derivation links are labelled with the Gorn address of the substitution location. This is uninformative here, since the only possible cosubstitution location is the root of the cosubstitution target,

whose Gorn address is always 0. Therefore I have labeled the cosubstitution links with the address of the node that cosubstitutes onto the cosubstitution target.

6 Comparison with other approaches

6.1 TAG approaches

In the MC-TAG approaches, quantifiers contribute a pair of trees to the derivation. One part corresponds to the visible DP, and the other corresponds to the scope-taking part. The scope-taking component floats upwards in the derivation until it finds the top of the scope domain. In the version of Kallmeyer and Romero and collaborators, the multiple components are part of the syntax: the scope part is a degenerate, single-node S tree that eventually attaches to the top of the scope domain. In the STAG version of Nesson and Shieber, the syntax contains a simple DP, and the multiple components are part of the semantics.

In both accounts, the scope-taking part of a quantifier must potentially remain unattached (i.e., part of one component in a tree set) over an unbounded number of TAG operations until the full scope domain is in view. To see this, note that in *A raindrop fell on the tire near the right corner of the hood of every car* there is a reading that entails that there is at least one raindrop-falling event per car. That means that it is possible for *every car* to take scope over the entire clause. Similar examples show that adjunctions (*on*, *near*) and substitutions (*of*) can be added between the quantifier and its scope without any grammatical limit.

The claim of the MC-TAG approach is that quantifiers and their scope domains are at most only tree-set local. Another way to interpret the MC-TAG analyses is to say that the relationship between a quantifier and its scope is local, but they are (at least temporarily) discontinuous, a kind of long-distance scrambling.

On the cosubstitution view here, the entire scope (*a raindrop fell on the tire near to the right corner of the hood of* __) is composed before the quantifier enters the derivation. There is no need to resort to multicomponent tree sets. The relationship between the quantifier and its scope is entirely local, since it is nothing more than the relationship between a predicate and its cosubstitution coargument.

6.2 Delimited continuations

In systems with delimited continuations (Felleisen 1988, Shan 2005, etc.), there are typically two elements: shift and reset (prompt). Shift is analogous to the uparrow used here to mark cosubstitution targets; it corresponds to the bottom of a scope domain. The reset marks the top of the scope domain. One common challenge in shift/reset systems is indexing occurrences of shift with matching occurrences of reset. In the TAG system here, there is no need for a separate reset element. Instead, the state of the derivation implicitly delimits the continuation that is captured by the uparrow (shift). The continuation will always contain the derivation up to the point at which the cosubstitution occurs, and we achieve delimitation without needing an explicit reset operator.

In other systems with delimited continuations such as Barker and Shan 2008, delimitation is accomplished by a system of optional typeshifting operators. Once again, the interleaving of the cosubstitution with the derivation makes typeshifting unnecessary here.

7 Generalized scope-taking

Most work in TAG semantics, including this paper so far, assumes that scope-taking elements all take scope over a sentence (S) and produce as a result a (quantified) sentence. More general scoping mechanisms allow these parameters to vary independently. Thus in Moortgat 1997, q(A,B,C) is a type that functions syntactically as an expression of type A, takes scope over a constituent of type B, and returns a result of type C, so the quantifiers discussed above would all be type q(DP, S, S). Barker and Shan 2008, Morrill et al. 2007, and others provide directly analogous general scope-taking categories.

This more general approach allows new kinds of linguistic analyses. For instance, I argue in Barker 2008 that in *two men with the same name*, the word *same* is a scope-taking quantifier that functions locally as an adjective and takes scope over a nominal, in this case, *men with the __ name*. Thus *same* has category q(Adj, NP, NP).

For a second instance due to Moortgat (in teaching materials circa 2000), in order to handle the

bracketed phrase in *a book [the author of which] I know*, we can analyze *which* as having category q(DP, DP, RelPn): it behaves syntactically as a DP, it takes scope over a DP (the bracketed phrase), and the result functions as a relative pronoun.

It would probably be straightforward to allow existing TAG accounts to accommodate non-S scope targets, though perhaps at the cost of increased generative power. It is far from clear, however, how these accounts would allow scope-takers to return arbitrary result categories. This would require allowing a deeply embedded constituent to change the syntactic category of the constituent in which it is embedded.

The cosubstitution account here, however, was created with the general case in mind. We simply refine the definition of cosubstitution to require that if a cosubstitution target has category q(A,B,C), the surrounding cosubstitution argument must have category A on its foot, B on its root, and the result of the cosubstitution operation is treated as a tree of category C. Perhaps in the TAG tradition of split categories, the root of the resulting tree would have B as its lower category and C as its upper category.

8 Conclusions

A fully general system for scope-taking requires providing a scope-taking element with its delimited continuation. Derivational flexibility in TAG allows construction of delimited continuations on the fly. Adding the operation of cosubstitution as dual to substitution makes for a strikingly simple but fully general scope-taking system.

Still, after all is said and done, there is something multicomponent-ish about the cosubstitution approach. The semantics of cosubstitution expands the DP node on the frontier of the tree at the same time that it also adds semantic material at the top of the tree. As I mentioned, I don't view cosubstitution as a competitor to the MC-TAG approaches, but rather as a reconceptualization. Given that scope-taking requires associating distant locations in a tree, just what is the nature of the required dependency? Exactly what sort of multicomponent-ness is required? What expressive power is needed, and what extra generative capacity?

In the system as described above, the answer to the generative capacity question is simple. For every co-TAG grammar, there is an ordinary TAG grammar in which each root labeled X↑ is replaced with the label X. For every derivation in the co-TAG grammar, there is a derivation in the ordinary TAG grammar that generates the same tree with the same string. Furthermore, cosubstitution is defined in every situation in which ordinary substitution would be defined. As a result, syntactically, co-TAG is both weakly and strongly equivalent to TAG.

Of course, even though cosubstitution is defined when the root of the cosubstitution argument is not of type $\mathtt{t}$ (e.g., an S node), the lambda term constructed by the semantics may be ill-typed. In effect, we've only been interested so far in a subclass of derivations, the ones in which uparrowed constituents are combined with arguments rooted in S.

Building ill-typed lambda terms is clearly not satisfactory. However, it is easily fixed. We restrict the version of cosubstitution defined above to cases in which the cosubstitution argument is rooted in S, and we add two new cosubstitution definitions to cover cases in which the argument is rooted in some category A where A is distinct from S.

For the first new version, the syntax will be the same, except that now the category of the resulting derived tree will be A↑ instead of A. The semantics will be $\lambda\kappa\lambda\gamma.\mathbf{Q}(\lambda x.\gamma(\kappa x))$, where κ is a function of type $\langle \mathbf{B}, \mathbf{A}\rangle$, γ is a function of type $\langle \mathbf{A}, \mathtt{t}\rangle$, $\mathbf{Q}$ is the semantics of the B↑ tree (and so has type $\langle\langle \mathbf{B}, \mathtt{t}\rangle, \mathtt{t}\rangle$), and x is a variable of type $\mathbf{B}$. This rule says that if a scope-taking element does not find its scope, it turns the constituent it combines with into a new scope-taking element. When this new, complex scope-taker finally finds its scope, meanings compose in such a way that the original scope-taker takes scope over the entire domain.

This rule in effect allows a scope-taking expression to combine bottom-up if desired, much in the way that the MC-TAG scope analyses work. In the full system, then, we can either accumulate a scope domain piecemeal, layer by layer, in the style of the MC-TAG analysis, or we can jump directly to the top layer in one swoop. See Barker 2007 for a discussion of a type-logical system in which these two conceptions of scope-taking coexist in a single grammar.

The second new version of the cosubstitution

rule covers cases in which the cosubstitution argument is rooted in a scope-taking category A↑ to begin with. This will happen if there are already scope-taking elements incorporated into the cosubstitution argument. In this case, the syntactic result is unchanged (A↑ again), and the semantics is $\lambda\kappa\lambda\gamma.\mathbf{Q}(\lambda x.(\kappa x)\,\gamma)$, where κ now has type $\langle \mathbf{B}, \mathbf{A}\uparrow\rangle$. This rule gives the newest scope-taker scope over all of the other scope-taking elements already present in the (partial) domain.

As long as there aren't any initial trees rooted in S↑, or internal nodes with arrows, it is easy to see that the final co-TAG tree will have no arrows in it. Clearly, for every co-TAG derivation, there is a derivation in the corresponding de-arrowed TAG in which the final tree is identical, and vice-versa. Furthermore, with the new cosubstitution rules in place, every derviation in the co-TAG has a well-typed (and sensible) semantic interpretation. Finally, note that all TAG grammars are also co-TAG grammars.

In other words, co-TAG has exactly the same weak and strong generative capacity as TAG.

References

Barker, Chris. 2002. Continuations and the nature of quantification. *Natural Language Semantics* **10.3**: 211–242.

Barker, Chris. 2007. Direct Compositionality on Demand. In Chris Barker and Pauline Jacobson (eds). *Direct Compositionality*. Oxford University Press. 102–131.

Barker, Chris. 2008. Parasitic Scope. *Linguistics and Philosophy*. **30.4**: 407–444.

Barker, Chris, and Chung-chieh Shan. 2006. Types as graphs: continuations in Type Logical Grammar. *Journal of Logic, Language and Information* **15.4**: 331–370.

Barker, Chris, and Chung-chieh Shan. 2008. Donkey anaphora is in-scope binding. *Semantics and Pragmatics* **1.1**: 1–42.

Bernardi, Raffaella and Michael Moortgat. 2010. Continuation semantics for the Lambek-Grishin calculus. *Information and Computation* **208.5**: 397–416.

Felleisen, Mattias. 1988. The theory and practice of first-class prompts. In *Popl 88: Proceedings of the 15th acm sigplan-sigact symposium on principles of programming languages*. 180–190.

de Groote, P.: 2001, Continuations, type raising, and classical logic, in R. van Rooy and M. Stokhof (eds.), *Thirteenth Amsterdam Colloquium*, pp. 97–101. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Heim, Irene and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Blackwell.

Joshi, Aravind, Laura Kallmeyer, and Maribel Romero. 2008. Flexible composition in LTAG: quantifier scope and inverse linking. In H. Bunt and R. Muskens (eds). *Computing Meaning*, vol. 3, Springer. 233–256.

Joshi, Aravind and Yves Schabes. 1997. Tree-Adjoining Grammars. In *Handbook of Formal Languages, Beyond Words*, vol. 3. 69–123.

May, R.: 1985, *Logical Form: Its Structure and Derivation*, MIT Press, Cambridge, MA.

Morrill, Glyn, Mario Fadda and Oriol Valentn. 2007. Nondeterministic Discontinuous Lambek Calculus. In *Proceedings of the Seventh International Workshop on Computational Semantics, IWCS7*. Tilburg.

Nesson, Rebecca and Stuart Shieber. 2006. Simpler TAG semantics through synchronization. In *Proceedings of the 11th Conference on Formal Grammar*. CSLI.

Shan, Chung-chieh. 2005. *Linguistic side effects*. Harvard PhD.

Shieber, Stuart and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, vol. 3, 253–258.