Preserving Semantic Dependencies in Synchronous Tree Adjoining Grammar*

William Schuler
University of Pennsylvania
200 South 33rd Street
Philadelphia, PA 19104 USA
schuler@linc.cis.upenn.edu

Abstract

Rambow, Wier and Vijay-Shanker (Rambow et al., 1995) point out the differences between TAG derivation structures and semantic or predicate-argument dependencies, and Joshi and Vijay-Shanker (Joshi and Vijay-Shanker, 1999) describe a monotonic compositional semantics based on attachment order that represents the desired dependencies of a derivation without underspecifying predicate-argument relationships at any stage. In this paper, we apply the Joshi and Vijay-Shanker conception of compositional semantics to the problem of preserving semantic dependencies in Synchronous TAG translation (Shieber and Schabes, 1990; Abeillé et al., 1990). In particular, we describe an algorithm to obtain the semantic dependencies on a TAG parse forest and construct a target derivation forest with isomorphic or locally non-isomorphic dependencies in $\mathcal{O}(n^7)$ time.

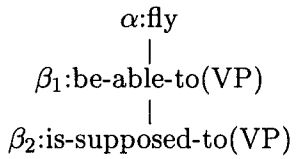
1 Introduction

The primary goal of this paper is to solve the problem of preserving semantic dependencies in Isomorphic Synchronous Tree Adjoining Grammar (ISTAG) (Shieber, 1994; Shieber and Schabes, 1990), a variant of Tree Adjoining Grammar (Joshi, 1985) in which source and target elementary trees are assembled into isomorphic derivations. The problem, first described in Rambow, Wier and Vijay-Shanker (Rambow et al., 1995), stems from the fact that the TAG derivation structure – even using a flat adjunction of modifiers (Schabes and Shieber, 1994) – deviates from the appropriate dependency structure in certain cases. This can result in translation errors.

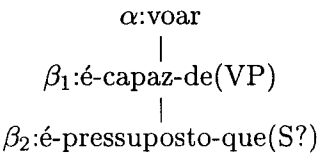
For example, if we parse sentence (1),

(1) X is supposed to be able to fly.

using the trees in Figure 1, we get the following derivation:[1]

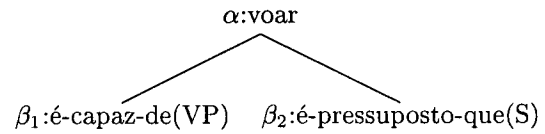


with the auxiliary *is-supposed-to* adjoining at the VP to predicate over *be-able-to* and the auxiliary *be-able-to* adjoining at the VP to predicate over *fly*. If we then try to assemble an isomorphic tree in a language such as Portuguese (which makes less use of raising verbs) using the ISTAG transfer rules in Figure 2, we will be forced into an ill-formed derivation:



because the raising construction *is-supposed-to* translates to a bridge construction *é-pressuposto-que* and cannot adjoin anywhere in the tree for *é-capaz-de* (the translation of *be-able-to*) because there is no *S*-labeled adjunction site.

The correct target derivation:



*The author would like to thank Karin Kipper, Aravind Joshi, Martha Palmer, Norm Badler, and the anonymous reviewers for their valuable comments. This work was partially supported by NSF Grant SBR8920230 and ARO Grant DAAH0404-94-GE-0426.

[1]The subject is omitted to simplify the diagram.

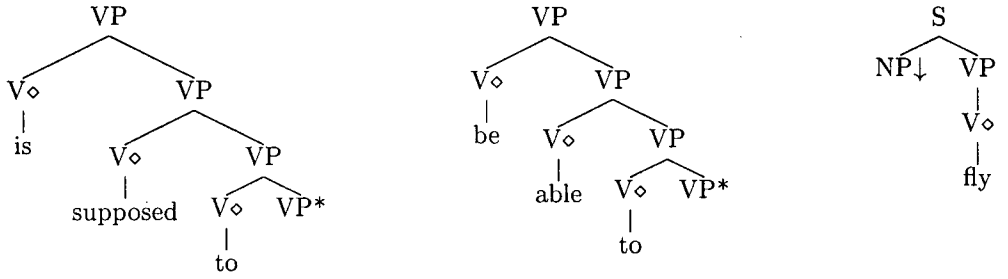
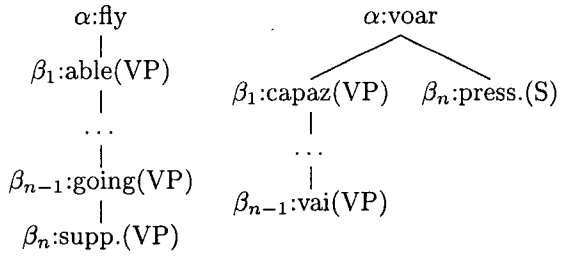


Figure 1: Sample elementary trees for "supposed to be able to fly"

which yields the translation in sentence (2),

(2) É pressuposto que X é capaz de voar.

is not isomorphic to the source. Worse, this non-isomorphism is unbounded, because the bridge verb *pressuposto* may have to migrate across any number of intervening raising verbs to find an ancestor that contains an appropriate adjunction site:



This sort of non-local non-isomorphic transfer cannot be handled in a synchronous TAG that has an isomorphism restriction on derivation trees. On the other hand, we do not wish to return to the original non-local formulation of synchronous TAG (Shieber and Schabes, 1990) because the non-local inheritance of links on the derived tree is difficult to implement, and because the non-local formulation can recognize languages beyond the generative power of TAG. Rambow, Wier and Vijay-Shanker themselves introduce D-Tree Grammar (Rambow et al., 1995) and Candito and Kahane introduce the DTG variant Graph Adjunction Grammar (Candito and Kahane, 1998b) in order to solve this problem using a derivation process that mirrors composition more directly, but both involve potentially significantly greater recognition complexity than TAG.

2 Overview

Our solution is to retain ISTAG, but move the isomorphism restriction from the derivation structure to the predicate-argument attachment structure described in (Joshi and Vijay-Shanker, 1999).

This structure represents the composition of semantic predicates for lexicalized elementary trees, each of which contains a 'predicate' variable associated with the situation or entity that the predicate introduces, and a set of 'argument' variables associated with the foot node and substitution sites in the original elementary tree. The predicates are composed by identifying the predicate variable in one predicate with an argument variable in another, so that the two variables refer to the same situation or entity.

Composition proceeds from the bottom up on the derivation tree, with adjuncts traversed in order from the lowest to the highest adjunction site in each elementary tree, in much the same way that a parser produces a derivation. Whenever an initial tree is substituted, its predicate variable is identified in the composed structure with an argument variable of the tree it substitutes into. Whenever an auxiliary tree is adjoined, the predicate variable of the tree it adjoins into is identified in the composed structure with one of its own argument variables. In cases of adjunction, an auxiliary tree's semantics can also specify which variable will become the predicate variable of the composed structure for use in subsequent adjunctions at higher adjunction sites: a *modifier* auxiliary will return the host tree's original predicate variable, and a *predicative* auxiliary will return its own predicate variable.[2] Since the traversal must

[2]See (Schabes and Shieber, 1994) for definitions of modifier and predicative auxiliaries.

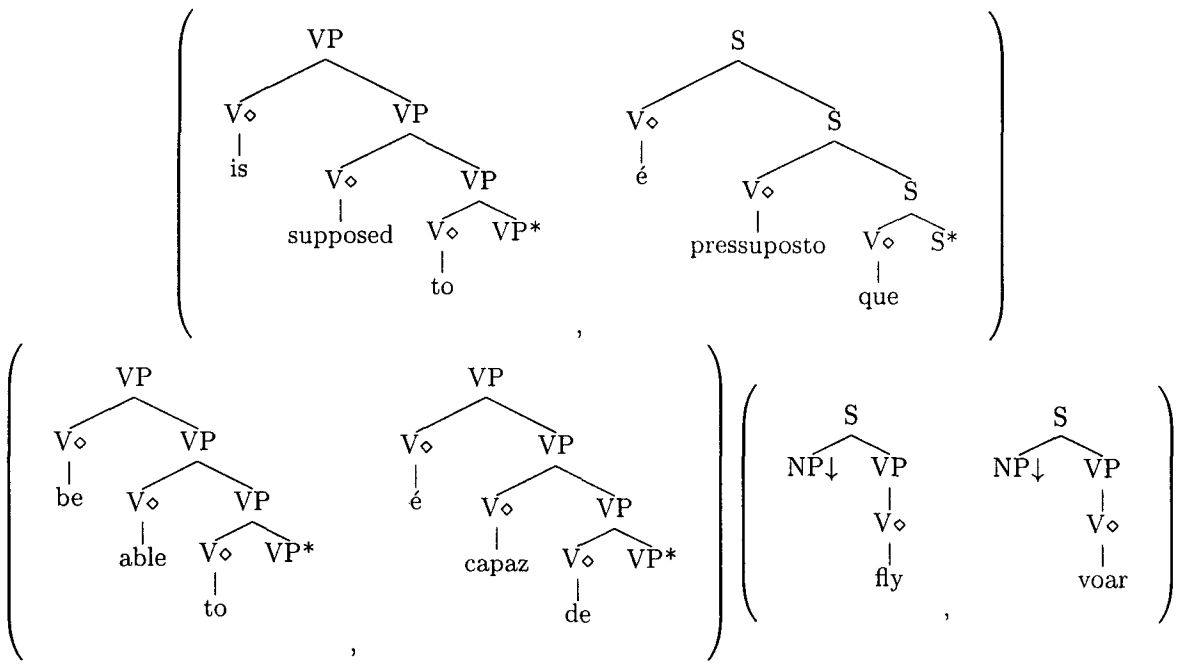
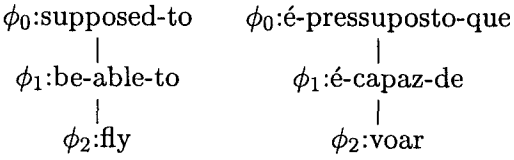


Figure 2: Synchronous tree pairs for "supposed to be able to fly"

proceed from the bottom up, the attachment of predicates to arguments is neither destructive nor underspecified at any stage in the interpretation.

For example, assume the initial tree α*:fly* has a predicate variable s_1, representing the situation of something flying, and an argument variable x_1, representing the thing that is flying; and assume the predicative auxiliary tree β_1*:be-able-to* has a predicate variable s_2, representing the situation of something being possible, and an argument variable s_3, representing the thing that is possible. If β_1 is now adjoined into α, the composed structure would have s_1 identified with s_3 (since the situation of flying is the thing that is possible), and s_2 as an overall predicate variable, so if another tree later adjoins into this composed structure rooted on α, it will predicate over s_2 (the situation that flying is possible) rather than over α's original predicate variable s_1 (the situation of flying by itself). Note that Joshi and Vijay-Shanker do not require the predicate and modifier distinctions, because they can explicitly specify the fates of any number of predicate variables in a tree's semantic representation. For simplicity, we will limit our discussion to only the two possibilities of predicative and modifier auxiliaries, using one predicate variable per tree.

If we represent each such predicate-argument attachment as an arc in a directed graph, we can view the predicate-argument attachment structure of a derivation as a dependency graph, in much the same way as Candito and Kahane interpret the original derivation trees (Candito and Kahane, 1998a). More importantly, we can see that this definition predicts the predicate-argument dependencies for sentences (1) and (2) to be isomorphic:



even though their derivation trees are not.

This is because the predicative auxiliary for *é-capaz-de* returns its predicate variable to the host tree for subsequent adjunctions, so the auxiliary tree for *é-pressuposto-que* can attach it as one of its arguments, just as if it had adjoined directly to the auxiliary, as *supposed-to* does in English.

It is also important to note that Joshi and Vijay-Shanker's definition of TAG compositional semantics differs from that of Shieber

and Schabes (Shieber and Schabes, 1990) using Synchronous TAG, in that the former preserves the scope ordering of predicative adjunctions, which may be permuted in the latter, altering the meaning of the sentence.[3] It is precisely this scope-preserving property we hope to exploit in our formulation of a dependency-based isomorphic synchronous TAG in the next two sections. However, as Joshi and Vijay-Shanker suggest, the proper treatment of synchronous translation to logical form may require a multi-component Synchronous TAG analysis in order to handle quantifiers, which is beyond the scope of this paper. For this reason, we will focus on examples in machine translation.

3 Obtaining Source Dependencies

If we assume that this attachment structure captures a sentence's semantic dependencies, then in order to preserve semantic dependencies in synchronous TAG translation, we will need to obtain this structure from a source derivation and then construct a target derivation with an isomorphic structure.

The first algorithm we present obtains semantic dependencies for derivations by keeping track of an additional field in each chart item during parsing, corresponding to the predicate variable from Section 2. Other than the additional field, the algorithm remains essentially the same as the parsing algorithm described in (Schabes and Shieber, 1994), so it can be applied as a transducer during recognition, or as a post-process on a derivation forest (Vijay-Shanker and Weir, 1993). Once the desired dependencies are obtained, the forest may be filtered to select a single most-preferred tree using statistics or rule-based selectional restrictions on those dependencies.[4]

For calculating dependencies, we define a function $arg(\eta_\gamma)$ to return the argument position associated with a substitution site or foot node η in elementary tree γ. Let a dependency be defined as a labeled arc $\langle \phi, l, \psi \rangle$, from predicate ϕ to predicate ψ with label l.

- For each tree selected by ϕ, set the predicate variable of each anchor item to ϕ.
- For each **substitution** of initial tree α_ψ with predicate variable ω into γ_ϕ at node address η, emit $\langle \phi, arg(\gamma, \eta), \omega \rangle$
- For each **modifier adjunction** of auxiliary tree β_ψ into tree γ_ϕ with predicate variable χ, emit $\langle \psi, arg(\beta, FOOT), \chi \rangle$ and set the predicate variable of the composed item to χ.
- For each **predicative adjunction** of auxiliary tree β_ψ with predicate variable ω into tree γ_ϕ with predicate variable χ, emit $\langle \psi, arg(\beta, FOOT), \chi \rangle$ and set the predicate variable of the composed item to ω.
- For all other productions, propagate the predicate variable up along the path from the main anchor to the root.

Since the number of possible values for the additional predicate variable field is bounded by n, where n is the number of lexical items in the input sentence, and none of the productions combine more than one predicate variable, the complexity of the dependency transducing algorithm is $\mathcal{O}(n^7)$.

This algorithm can be applied to the example derivation tree in Section 1,

α:fly
|
β_1:be-able-to(VP)
|
β_2:is-supposed-to(VP)

which resembles the stacked derivation tree for Candito and Kahane's example 5a, "Paul claims Mary said Peter left."

First, we adjoin β_2*:is-supposed-to* at node VP of β_1*:be-able-to*, which produces the dependency ⟨is-supposed-to, 0,be-able-to⟩. Then we adjoin β_1*:be-able-to* at node VP of α*:fly*, which produces the dependency ⟨be-able-to, 0,fly⟩. The resulting dependencies are represented graphically in the dependency structure below:

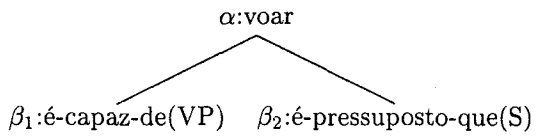
ϕ_0:supposed-to
|
ϕ_1:be-able-to(0)
|
ϕ_2:fly(0)

This example is relatively straightforward, simply reversing the direction of adjunction dependencies as described in (Candito and Kahane, 1998a), but this algorithm can transduce

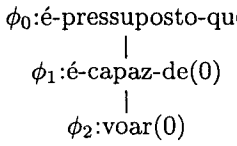
[3] See (Joshi and Vijay-Shanker, 1999) for a complete description.

[4] See (Schuler, 1998) for a discussion of statistically filtering TAG forests using semantic dependencies.

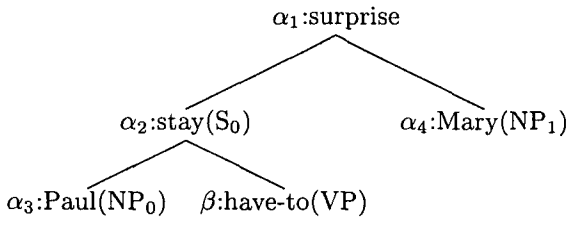
the correct isomorphic dependency structure for the Portuguese derivation as well, similar to the distributed derivation tree in Candito and Kahane's example 5b, "Paul claims Mary seems to adore hot dogs," (Rambow et al., 1995), where there is no edge corresponding to the dependency between the raising and bridge verbs:



We begin by adjoining β_1*:é-capaz-de* at node VP of α*:voar*, which produces the dependency ⟨é-capaz-de, 0,voar⟩, just as before. Then we adjoin β_2*:é-pressuposto-que* at node S of α*:voar*. This time, however, we must observe the predicate variable of the chart item for α*:voar* which was updated in the previous adjunction, and now references *é-capaz-de* instead of *voar*. Because the transduction rule for adjunction uses the predicate variable of the parent instead of just the predicate, the dependency produced by the adjunction of β_2 is ⟨é-pressuposto-que, 0,é-capaz-de⟩, yielding the graph:

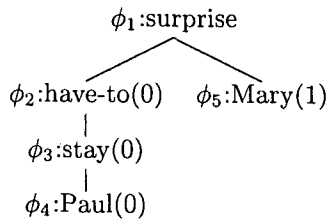


The derivation examples above only address the preservation of dependencies through adjunction. Let us now attempt to preserve both substitution and adjunction dependencies in transducing a sentence based on Candito and Kahane's example 5c, "That Paul has to stay surprised Mary," in order to demonstrate how they interact.[5] We begin with the derivation tree:



[5]We have replaced *want to* in the original example with *have to* in order to highlight the dependency structure and set aside any translation issues related to PRO control.

As Candito and Kahane point out, this derivation tree does not match the dependency structure of the sentence as described in Meaning Text Theory (Mel'cuk, 1988), because there is no edge in the derivation corresponding to the dependency between *surprise* and *have-to* (the *necessity* of Paul's staying is what surprises Mary, not his staying in itself). Using the above algorithm, however, we can still produce the desired dependency structure:



by adjoining β*:have-to* at node VP of α_2*:stay* to produce a composed item with *have-to* as its predicate variable, as well as the dependency ⟨have-to, 0,stay⟩. When α_2*:stay* substitutes at node S_0 of α_1*:surprise*, the resulting dependency also uses the predicate variable of the argument, yielding ⟨surprise, 0,have-to⟩.

4 Obtaining Target Derivations

Once a source derivation is selected from the parse forest, the predicate-argument dependencies can be read off from the items in the forest that constitute the selected derivation. The resulting dependency graph can then be mapped to a forest of target derivations, where each predicate node in the source dependency graph is linked to a set of possible elementary trees in the target grammar, each of which is instantiated with substitution or adjunction edges leading to other linked sets in the forest. The elementary trees in the target forest are determined by the predicate pairs in the transfer lexicon, and by the elementary trees that can realize the translated targets. The substitution and adjunction edges in the target forest are determined by the argument links in the transfer lexicon, and by the substitution and adjunction configurations that can realize the translated targets' dependencies.

Mapping dependencies into substitutions is relatively straightforward, but we have seen in Section 2 that different adjunction configurations (such as the raising and bridge verb ad-

junctions in sentences (1) and (2)) can correspond to the same dependency graph, so we should expect that some dependencies in our target graph may correspond to more than one adjunction configuration in the target derivation tree. Since a dependency may be realized by adjunctions at up to n different sites, an unconstrained algorithm would require exponential time to find a target derivation in the worst case. In order to reduce this complexity, we present a dynamic programming algorithm for constructing a target derivation forest in time proportional to $\mathcal{O}(n^4)$ which relies on a restriction that the target derivations must preserve the relative scope ordering of the predicates in the source dependency graph.

This restriction carries the linguistic implication that the scope ordering of adjuncts is part of the meaning of a sentence and should not be re-arranged in translation. Since we exploit a notion of locality similar to that of Isomorphic Synchronous TAG, we should not expect the generative power of our definition to exceed the generative power of TAG, as well.

First, we define an ordering of predicates on the source dependency graph corresponding to a depth-first traversal of the graph, originating at the predicate variable of the root of the source derivation, and visiting arguments and modifiers in order from lowest to highest scope. In other words, arguments and modifiers will be ordered from the bottom up on the elementary tree structure of the parent, such that the foot node argument of an elementary tree has the lowest scope among the arguments, and the first adjunct on the main (trunk) anchor has the lowest scope among the modifiers.

Arguments, which can safely be permuted in translation because their number is finitely bounded, are traversed entirely before the parent; and modifiers, which should not be permuted because they may be arbitrarily numerous, are traversed entirely after the parent. This enumeration will roughly correspond to the scoping order for the adjuncts in the source derivation, while preventing substituted trees from interrupting possible scoping configurations. We can now identify all the descendants of any elementary tree in a derivation because they will form a consecutive series in the enumeration described above. It therefore provides a convenient way to generate a target derivation forest that preserves the scoping information in the source, by 'parsing' the scope-ordered string of elementary trees, using indices on this enumeration instead of on a string yield.

It is important to note that in defining this algorithm, we assume that all trees associated with a particular predicate will use the same argument structure as that predicate.[6] We also assume that the set of trees associated with a particular predicate may be filtered by transferring information such as mood and voice from source to target predicates.

Apart from the different use of indices, the algorithm we describe is exactly the reverse of the transducer described in Section 3, taking a dependency graph $\mathcal{D}$ and producing a TAG derivation forest containing exactly the set of derivation trees for which those dependencies hold. Here, as in a parsing algorithm, we define forest items as tuples of $\langle \gamma_\phi, \eta, \bot, i, j, \chi \rangle$ where α, β, and γ are elementary trees with node η, ϕ and ψ are predicates, χ and ω be predicate variables, and $\top$ and $\bot$ are delimiters for opening and closing adjunction, but now let i, j, and k refer to the indices on the scoping enumeration described above, instead of on an input string. In order to reconcile scoping ranges for substitution, we must also define a function $first(\phi)$ to return the leftmost (lowest) edge of the ϕ's range in the scope enumeration, and $last(\phi)$ to return the rightmost (highest) edge of the ϕ's range in the scope enumeration.

- For each tree γ mapped from predicate ϕ at scope i, introduce $\langle \gamma_\phi, first(\phi), i+1, \phi \rangle$.
- If $\langle \phi, arg(\gamma, \eta), \omega \rangle \in \mathcal{D}$, try **substitution** of α into γ:

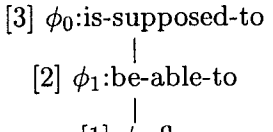
$$\frac{\langle \alpha_\psi, ROOT, \top, first(\omega), last(\omega), \omega \rangle}{\langle \gamma_\phi, \eta, \bot, -, -, - \rangle}$$

[6] Although this does not hold for certain relative clause elementary trees with wh-extractions as substitutions sites (since the wh-site is an argument of the main verb of the clause instead of the foot node), Candito and Kahane (Candito and Kahane, 1998b) suggest an alternative analysis which can be extended to TAG by adjoining the relative clause into its wh-word as a predicative adjunct, and adjoining the wh-word into the parent noun phrase as a modifier, so the noun phrase is treated as an argument of the wh-word rather than of the relative clause.

- If $\langle \psi, arg(\beta, FOOT), \chi \rangle \in \mathcal{D}$, try **modifier adjunction** of β into γ:
$$\frac{\langle \gamma_\phi, \eta, \bot, i, j, \chi \rangle \quad \langle \beta_\psi, ROOT, \top, j, k, \omega \rangle}{\langle \gamma_\phi, \eta, \bot, i, k, \chi \rangle}$$
- If $\langle \psi, arg(\beta, FOOT), \chi \rangle \in \mathcal{D}$, try **predicative adjunction** of β into γ:
$$\frac{\langle \gamma_\phi, \eta, \bot, i, j, \chi \rangle \quad \langle \beta_\psi, ROOT, \top, j, k, \omega \rangle}{\langle \gamma_\phi, \eta, \top, i, k, \omega \rangle}$$
- Apply productions for nonterminal projection as in the transducer algorithm, propagating index ranges and predicative variables up along the path from the main anchor to the root.

Since none of the productions combine more than three indices and one predicate variable, and since the indices and predicate variable may have no more than n distinct values, the algorithm runs in $\mathcal{O}(n^4)$ time. Note that one of the indices may be redundant with the predicate variable, so a more efficient implementation might be possible in $\mathcal{O}(n^3)$.

We can demonstrate this algorithm by translating the English dependency graph from Section 1 into a derivation tree for Portuguese. First, we enumerate the predicates with their relative scoping positions:



Then we construct a derivation forest based on the translated elementary trees *α:voar*, *β1:é-capaz-de*, and *β2:é-pressuposto-que*. Beginning at the bottom, we assign to these constituents the relative scoping ranges of 1-2, 2-3, and 3-$, respectively, where $ is a terminal symbol.

$$\langle \alpha_{voar}, 1, 2, .. \rangle \quad \langle \beta_{capaz}, 2, 3, .. \rangle \quad \langle \beta_{press}, 3, \$, .. \rangle$$

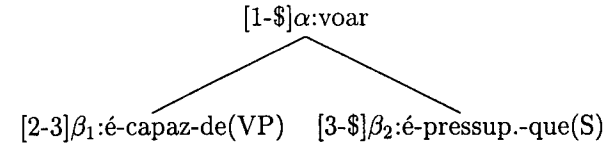
Since there is a dependency from *be-able-to* to *fly*, we can adjoin *β1:é-capaz-de* to *α:voar* such that it covers the range of scopes from 1 to 3 (from *voar* to *é-capaz-de*), so we add this possibility to the forest.

$$\frac{\langle \alpha_{voar}, 1, 2, .. \rangle \quad \langle \beta_{capaz}, 2, 3, .. \rangle}{\langle \alpha_{voar}, 1, 3, capaz \rangle} \quad \langle \beta_{press}, 3, \$, .. \rangle$$

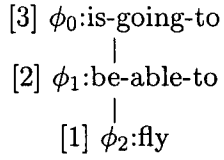
There is also a dependency from *is-supposed-to* to *be-able-to* allowing us to adjoin *β2:é-pressuposto-que* to *β1:é-capaz-de* to make it cover the range from 2 to $, but there would be no S node to host its adjunction, so this possibility can not be added to the forest. We can, however, adjoin *β2:é-pressuposto-que* to the instance of *α:voar* extending to *β1:é-capaz-de* that covers the range from 1 to 3, resulting in a complete analysis of the entire scope from 1 to $, (from *α:voar* to *β2:pressuposto*) rooted on *voar*:

$$\frac{\dfrac{\langle \alpha_{voar}, 1, 2, .. \rangle \quad \langle \beta_{capaz}, 2, 3, .. \rangle}{\langle \alpha_{voar}, 1, 3, capaz \rangle} \quad \langle \beta_{press}, 3, \$, .. \rangle}{\langle \alpha_{voar}, 1, \$, press \rangle}$$

which matches the distributed derivation tree where both auxiliary trees adjoin to *voar*.



Let us compare this to a translation using the same dependency structure, but different words:



Once again we select trees in the target language, and enumerate them with scoping ranges in a pre-order traversal, but this time the construction at scope position 3 must be translated as a raising verb (*vai*) instead of as a bridge construction (*é-pressuposto-que*):

$$\langle \alpha_{voar}, 1, 2, .. \rangle \quad \langle \beta_{capaz}, 2, 3, .. \rangle \quad \langle \beta_{vai}, 3, \$, .. \rangle$$

Although we can still adjoin *β1:ser-capaz-de* at the VP node of *α:voar*, we will have nowhere to adjoin *β2:vai*, since the VP node of *α:voar* is now occupied, and only one predicative tree may adjoin at any node.[7]

$$\frac{\langle \alpha_{voar}, 1, 2, .. \rangle \quad \langle \beta_{capaz}, 2, 3, .. \rangle}{\langle \alpha_{voar}, 1, 3, capaz \rangle} \quad \langle \beta_{vai}, 3, \$, .. \rangle$$

[7] See (Schabes and Shieber, 1994) for the motivations of this restriction.

Fortunately, we can also realize the dependency between *vai* and *ser-capaz-de* by adjoining β_2*:vai* at the VP.

$$\langle \alpha_{voar}, 1, 2, .. \rangle \; \frac{\langle \beta_{capaz}, 2, 3, .. \rangle \;\; \langle \beta_{vai}, 3, \$, .. \rangle}{\langle \beta_{capaz}, 2, \$, vai \rangle}$$

The new instance spanning from 2 to \$ (from β_1*:capaz* to β_2*:vai*) can then be adjoined at the VP node of *voar*, to complete the derivation.

$$\frac{\langle \alpha_{voar}, 1, 2, .. \rangle \;\; \dfrac{\langle \beta_{capaz}, 2, 3, .. \rangle \;\; \langle \beta_{vai}, 3, \$, .. \rangle}{\langle \beta_{capaz}, 2, \$, vai \rangle}}{\langle \alpha_{voar}, 1, \$, vai \rangle}$$

This corresponds to the stacked derivation, with β_2*:vai* adjoined to β_1*:ser-capaz-de* and β_1*:ser-capaz-de* adjoined to α*:voar*:

[1-\$] α:voar
|
[2-\$] β_1:ser-capaz-de(VP)
|
[3-\$] β_2:vai(VP)

5 Conclusion

We have presented two algorithms – one for interpreting a derivation forest as a semantic dependency graph, and the other for realizing a semantic dependency graph as a derivation forest – that make use of semantic dependencies as adapted from the notion of predicate-argument attachment in (Joshi and Vijay-Shanker, 1999), and we have described how these algorithms can be run together in a synchronous TAG translation system, in $\mathcal{O}(n^7)$ time, using transfer rules predicated on isomorphic or locally non-isomorphic dependency graphs rather than isomorphic or locally non-isomorphic derivation trees. We have also demonstrated how such a system would be necessary in translating a real-world example that is isomorphic on dependency graphs but globally non-isomorphic on derivation trees. This system is currently being implemented as part of the Xtag project at the University of Pennsylvania, and as natural language interface in the Human Modeling and Simulation project, also at Penn.

References

Anne Abeillé, Yves Schabes, and Aravind K. Joshi. 1990. Using lexicalized tree adjoining grammars for machine translation. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING '90)*, Helsinki, Finland, August.

Marie-Helene Candito and Sylvain Kahane. 1998a. Can the TAG derivation tree represent a semantic graph? In *Proceedings of the TAG+4 Workshop*, University of Pennsylvania, August.

Marie-Helene Candito and Sylvain Kahane. 1998b. Defining DTG derivations to get semantic graphs. In *Proceedings of the TAG+4 Workshop*, University of Pennsylvania, August.

Aravind Joshi and K. Vijay-Shanker. 1999. Compositional Semantics with Lexicalized Tree-Adjoining Grammar (LTAG): How Much Underspecification is Necessary? In *Proceedings of the 2nd International Workshop on Computational Semantics.*

Aravind K. Joshi. 1985. How much context sensitivity is necessary for characterizing structural descriptions: Tree adjoining grammars. In L. Karttunen D. Dowty and A. Zwicky, editors, *Natural language parsing: Psychological, computational and theoretical perspectives*, pages 206–250. Cambridge University Press, Cambridge, U.K.

Anthony S. Kroch. 1989. Asymmetries in long distance extraction in a TAG grammar. In M. Baltin and A. Kroch, editors, *Alternative Conceptions of Phrase Structure*, pages 66–98. University of Chicago Press.

Igor Mel'cuk. 1988. *Dependency syntax: theory and practice* . State University of NY Press, Albany.

Owen Rambow and Giorgio Satta. 1996. Synchronous Models of Language. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL '96).*

Owen Rambow, David Weir, and K. Vijay-Shanker. 1995. D-tree grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL '95).*

Yves Schabes and Stuart M. Shieber. 1994. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124.

William Schuler. 1998. Expoiting semantic dependencies in parsing. *Proceedings of the TAG+4 Workshop.*

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING '90)*, Helsinki, Finland, August.

Stuart M. Shieber. 1994. Restricting the weak-generative capability of synchronous tree adjoining grammars. *Computational Intelligence*, 10(4).

K. Vijay-Shanker and D.J. Weir. 1993. The use of shared forests in tree adjoining grammar parsing. In *Proceedings of EACL '93*, pages 384–393.