Towards a Dynamic Version of TAG

Vincenzo Lombardo and Patrick Sturt
Università di Torino and University of Glasgow

1. Introduction

This paper proposes a syntactic framework that is appropriate for modeling human language processing. The work moves from the largely held assumption that human language processing is incremental, and aims at exploring the consequences of incrementality when the processing strategy is encoded in the operations of a formal system. This makes a syntactic framework consistent with the Transparency Hypothesis (Berwick and Weinberg, 1984), in that a syntactic theory must reflect the behaviour of the human syntactic processor in its formal operations.

Incrementality is a strategy that constrains the processor to analyse the input words from left to right, and to carry out a semantic interpretation of the partial structures (Marslen-Wilson, 1973). A parsimonious version of incrementality is what we call *strong incrementality*, which restricts incrementality to the case in which the parser maintains a fully connected tree at each state (cf. (Stabler, 1994)). There have been some proposals in the literature that claim that structure building occurs incrementally both in derivation and in parsing. Many aspects of the CCG formalism are grounded on incrementality (Steedman, 2000), and Phillips has provided a large amount of evidence that incrementality in the derivation process solves many problems in the definition of constituency (Phillips, 1998). The incremental nature of the formalism is a major issue in Milward's proposal of a *dynamic dependency grammar* (Milward, 1994). In this case, the syntactic formalism is expressed in the terms of a dynamic system, that is a system evolving in time through a number of steps. A *dynamic grammar* views the syntactic process as a sequence of transitions between adjacent states S_{i-1} and S_i while moving from left to right in the input string [1]. Thus, it naturally implements a strongly incremental strategy. The state S_i is a partial tree that spans the input string from the beginning to the i-th word, and at each step the parser tries to attach the word w_i into the current partial tree S_{i-1}, also called the *left context*. So, the definition of a formal dynamic grammar involves the definition of the shape of the partial trees and the formal operations that extend these partial trees.

Tree Adjoining Grammar (Joshi, Levy and Takahashi, 1975) is a well studied framework, that can realize a range of syntactic theories, provided a few constraints on trees and formal operations are satisfied. The goal of this paper is to devise a dynamic version of TAG, that retains the formal characteristics of the framework in terms of basic machinery, while constraining derivation and parsing to be strongly incremental. Of great interest for implementing incrementality are the wide domain of locality introduced by TAG rules, the adjunction operation, and the linguistic motivation for TAG elementary trees. Some of these considerations are shared by (Frank and Badecker, 2001), as points of strength for TAG in language production. Let us consider them in turn.

1) Wide domain of locality
In an empirical study on the Penn Treebank, aimed to discover the amount of non-lexical information which is necessary to implement a fully connected incremental processing strategy, we simulated the sequence of processing steps of an incremental parser on the treebank (Lombardo and Sturt, 2002). Given full connectedness, each new word must be attached to the preceding left context via some syntactic structure, called "connection path". In the simulation, connection paths often needed to be multiply levelled trees. See, e.g., the structure that connects the word "the" to the left context in:

[$_{\mathrm{S}}$ [$_{\mathrm{NP}}$ John] [$_{\mathrm{VP}}$ [$_{\mathrm{V}}$ thinks] [$_{\mathrm{S}}$ [$_{\mathrm{NP}}$ [$_{\mathrm{D}}$ the]]]]]

However, the form of these connection paths is quite predictable from linguistic observations. The use of multiply levelled elementary trees in TAG is an immediate encoding of this requirement. We also found that the introduction and co-indexation of traces needed to be carefully designed for the incremental setting. In TAG, given the wide domain of locality, filler and trace are both present in the same elementary tree, and then they may be separated through a number of adjunctions. This solution is particularly interesting for incrementality, provided that adjunctions are constrained to occur through the insertion of lexical material only to the right of the word whose elementary tree introduced the filler-trace pair.

1. Here we have used the generic term "syntactic process" to indicate both derivation and parsing. In fact, in a dynamic grammar both processes share the same mechanisms, and we will use the two terms interchangeably.

2) Adjunction
The adjunction operation provided by the TAG formalism is vital for the incremental processing of modifiers. In fact, in many cases the correct attachment of a word to the current partial tree requires the guess of an unknown number of left recursive structures. The adjunction operation permits the progressive extension of recursive structures, as they are required to process the input string. A typical use of left recursive structures in English is in possessive constructions. In a sentence like “Mary hated the salesman’s assistant’s hairstyle”, during incremental processing it is impossible to guess in advance the depth of the NP immediately dominating “the salesman”. One solution is to attach this NP immediately as the object of “hated”, and then to embed this NP via adjunction when the possessive is processed. This operation can be repeated an arbitrary number of times to produce an arbitrarily embedded structure (cf. (Thompson, Dixon and Lamping, 1991)).

3) Linguistically motivated elementary trees
TAG, and in particular Lexicalized TAG (Schabes, Abeillé and Joshi, 1988), constrain the form of the elementary trees to be linguistically motivated according to the number and the type of arguments required by some lexical anchor. This is a desirable property also for incremental processing, since in head-initial languages, items that are to occur on the right are often predicted from anchors on the left. Here we are not referring to any processing model, but to a basic requirement for a syntactic formalism that supports incrementality. We must also notice that linguistic constraints are not always enough to guarantee the full connectedness of the partial tree. It can be that the argument structures of two adjacent lexical anchors cannot be directly combined, and so we need some extra structure to guarantee full connectedness. Take again the previous example “John thinks the ...”: the lexical entry for “thinks” is an initial tree that predicts an S node on its right, marked for substitution; the lexical entry for “the” is an auxiliary tree rooted in NP; the extra structure provides the edge between S and NP, and must be included in one of the two elementary trees, even though this edge is not directly licensed by any of the lexical anchors in the partial tree. As a consequence, an elementary tree of a dynamic TAG may be larger than the argument domain of that elementary tree’s anchor. This is because extra structure is needed to guarantee connectivity (in the next section we will see a few examples of this case).

This paper describes the major issues of a *dynamic version of TAG* (called *DV-TAG*), which is suitable for incremental processing. We first describe the consequences that incrementality bears upon the shape of elementary trees and on the form of the attachment operations, together with some definitions. Then we describe the derivation process that produces the so-called derived tree. Finally, in order to illustrate the functioning of the formalism, we provide a few meaningful linguistic examples.

2. Dynamic Version of TAG (DV-TAG)

In this section we describe the basic machinery of DV-TAG. We start with elementary trees, and then we move to the attachment operations, Substitution and Adjunction.

2.1. Elementary trees

A DV-TAG grammar consists of elementary trees, divided into initial trees and auxiliary trees. Most of the definitions provided by TAG are also applicable to elementary trees in DV-TAG. The structure of elementary trees in DV-TAG is constrained by a number of syntactic modules (X-bar theory, case theory, thematic relations theory, some empty category principle(s), see (Kroch and Joshi, 1985)). In particular, we assume that each node has a distinguished head daughter, that each elementary tree is associated with a lexical anchor, and that the elementary tree includes the argument structure of the anchor. Long-distance dependencies have to be stated locally, in a single elementary tree. Auxiliary trees are minimal recursive structures with root and foot nodes identically labelled.

The major difference in comparison with TAG is that the shapes of the elementary trees must be constrained in order to permit the left-to-right derivation (and parsing) of all and only the legal structures. The linguistic motivations mentioned above may not be enough to constrain the elementary trees in a way that guarantees the construction of a fully connected structure [2]. This motivates the addition of extra information to elementary trees. Here we are not claiming that additional information is in some form, and is added on-the-fly while processing, but that elementary trees in DV-TAG are larger than the corresponding elementary trees in TAG.

Even if we ground this requirement on the empirical observations addressed in the Penn Treebank (Lombardo and Sturt, 2002), we can think of some principled way to constrain the extension of elementary trees beyond the modules above. A similar solution, devised again in the context of incremental processing, is the *type raising*

2. Remember that a fully connected structure is required by incremental processing.

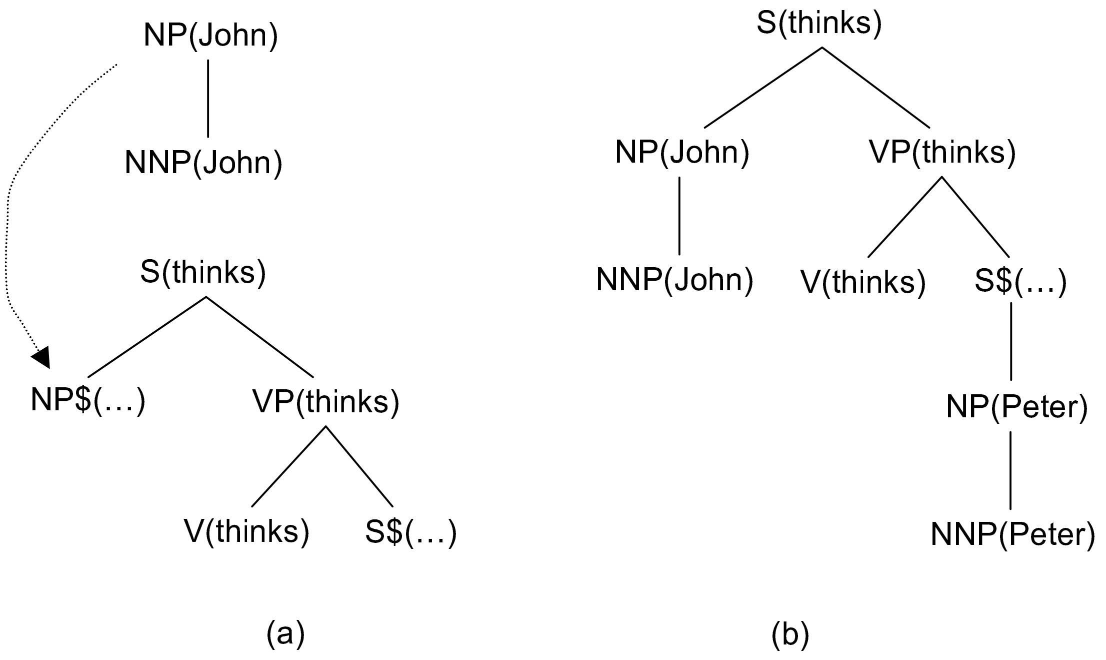


Figure 1: Initial trees and derivation for "John thinks Peter ...". Node labels include the head word. The symbol "$" indicates "marked for Substitution". (a) The initial trees for "John" and "thinks". (b) The derived tree for "John thinks Peter".

operation provided in the CCG theory (Steedman, 2000). We have not yet addressed such a problem at this stage of development. What may happen is that some portions of this extra information can overlap with the arguments of some lexical anchor which is yet to come (on the right of the current word). So, a DV-TAG grammar can include some redundant information, since the same portions of structure can belong to several elementary trees for different reasons.

In order to illustrate the case for redundant information, consider the sentence "John thinks Peter laughs". In Figure 1a there are the initial trees for "John" and "thinks", respectively. "Thinks" subcategorizes for a subject NP and a sentential clause S. The initial tree for "Peter" should be the same as the one for "John". However, given the substitution indicated by the dotted arrow, we have that the NP "Peter" has to be incorporated into the subject position of the complement clause licensed by "thinks" (Figure 1b). This means that the connection between S and NP must be realized by the elementary tree for "Peter". Certainly, the edge between S and NP is not projected by the lexical anchor "Peter", but is part of the initial tree for "Peter". It is linguistically licensed by "laughs", and so it must be part of the initial tree for "laughs" as well. As we see below, the attachment operations need to be more sophisticated in DV-TAG, because they also include some checking of structures previously built for connectivity reasons. We call this process *redundancy checking*. The edge existing between the complement S and the subject NP ("Peter") is redundancy checked when the initial tree for "laughs" is Substituted.

The structure in Figure 2 is more complex. It is an initial tree for a NP that is fronted as a consequence of an object extraction (in the example, the word "beans" in "Beans_1 John likes e_1"). This structure is common to all cases where the filler precedes the trace. It provides both the filler and the trace positions in the same elementary tree, and the two positions are co-indexed (index i). The assumption underlying this structure is that the fronting phenomenon must predict the embedded verb that licenses the fronted NP as an object. There are two nodes (S and NP) marked for Substitution: this means that, in order to have an object extraction we need an actual transitive verbal subcategorization frame and an actual subject. The S node shares the lexical head with the VP and the V nodes (see the index j).

Another underlying assumption that is not obvious in standard TAG is that maximal projections that are marked for substitution can also be internal nodes, and not leaves. This is the case for the internal S node in Figure 2. The nodes marked for substitutions are required to be roots of initial trees. The substitution that will take place here is the initial tree of a transitive verb, e.g. "like", whose initial tree [S [NP VP [V NP]]] is already part of the structure

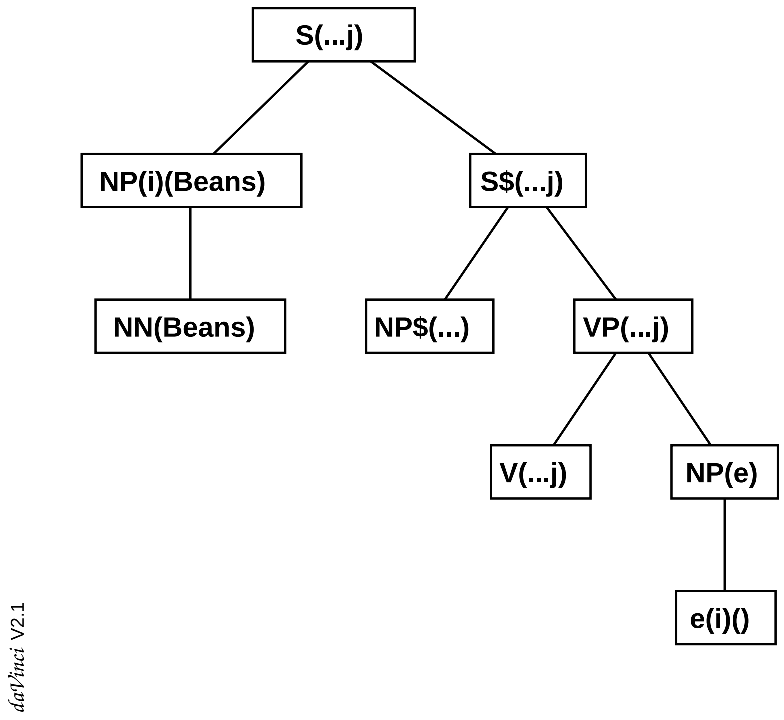


Figure 2: An elementary tree for an extracted object. Notice that it includes the constraints on the structure of the verbal subcategorization frame. Filler and trace nodes are co-indexed with i. The nodes including the j-index in parentheses are constrained to bear the same head daughter.

and will be rooted in “S$(...j)”. This substitution will be a redundancy checking only, since the whole substructure is already in the left context.

Finally, notice that adjunctions to the internal S node permit the insertion of other clauses, that take the filler and the trace further apart (“Beans_1 I know John likes e_1”). Notice that in this case, the elementary tree whose head is “likes” will be discontinuous in the derivation tree; clearly, the actual implementation of the redundancy checking mechanism would require some care to deal with such cases.

It can also be the case that some elementary tree introduces some other lexical element beyond the lexical head, as may occur, for example, when a verb selects the lexical form of the head of an argument PP (e.g., “go to”). In derivation and parsing, we assume that the pointer on the input string goes to the leftmost lexical head of the elementary tree, and it is possible to adjoin words in between (“go slowly to”). We stipulate that only one of the lexical elements is the lexical anchor of the elementary tree. Redundancy checking must also take into account lexical items.

2.2. Attachment operations

The attachment operations in DV-TAG are substitution and adjunction, with a number of differences on their applicability and functioning in comparison with standard TAG. In particular, their functioning has to implement some form of redundancy checking as mentioned above[3]. Their applicability depends on the legal attachment sites, with respect to left-to-right processing.

Since we proceed incrementally, every time we apply an operation there will be a *left context*, given by the tree spanning all the words from the beginning of the sentence to the word w_{i-1}, and a current elementary tree for the word w_i.

3. In actual implementations redundancy checking should be accompanied with feature upgrading.

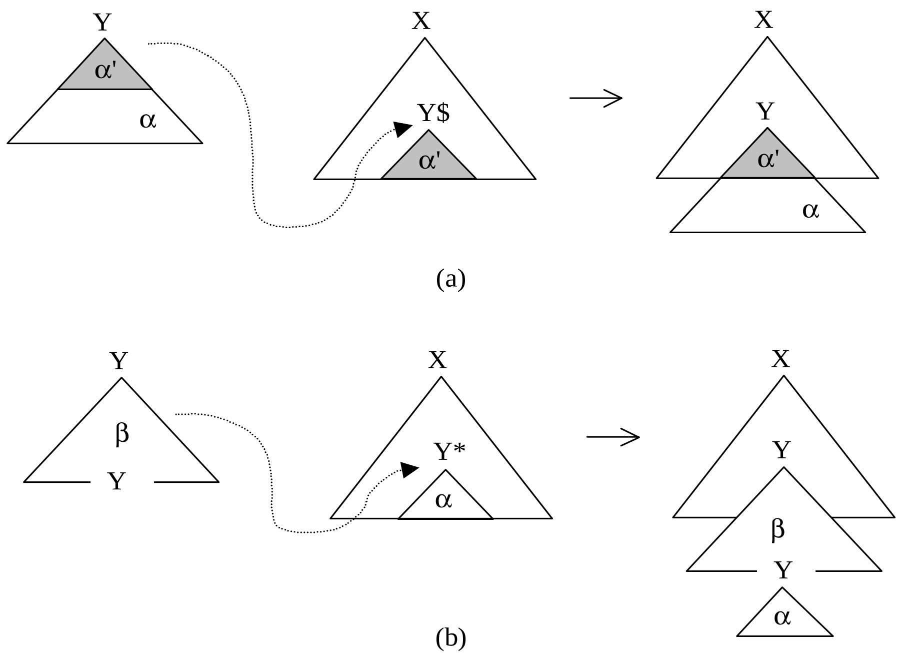


Figure 3: Substitution (a) and Adjunction (b).

- Substitution (see Figure 3(a)):
  In the Substitution operation, either the root node of the current initial tree is merged into the leftmost non-terminal node marked for substitution in the left context, or the root node of the left context is merged into the leftmost non-terminal node marked for substitution in the current initial tree, producing a new tree.

The root node and the substitution node must have the same name. Nodes marked for Substitution are indicated with a $ in the elementary trees. The leftmost relation on nodes marked for Substitution is calculated on the lowest nodes of their respective head projections[4].

- Adjunction (see Figure 3(b)):
  In the Adjunction operation, either the left context, which has the form of an auxiliary tree, is grafted into a non-terminal node marked for adjunction in the current elementary tree, or the current elementary tree, which is an auxiliary tree, is grafted into a non-terminal node marked for adjunction in the left context.

The root and foot nodes of the auxiliary tree, whether it is the left context or the current elementary tree, must match the node at which the auxiliary tree adjoins. It is important to notice that in DV-TAG both the left context and the current elementary tree can act as the auxiliary tree in adjunction. This is to say that either the left context is split because of the adjoining of some auxiliary tree, or some elementary tree is split because of the adjoining of the left context at some node. In the left context, nodes are marked for adjunction according to the definition of the accessible *fringe* (see below for the definition); in the elementary trees, nodes are overtly marked for adjunction.

The foot node of the auxiliary tree is matched with the node which is the site of adjunction. If this node has daughters marked for substitution, then it is necessary to move these daughters to the root node, leaving the co-indexed traces at the foot node. Traces allow us to keep the locality of the subcategorization constraints, while moved elements account for the structural constraints. An example of when this is necessary is in the processing of expressions where a modifier intervenes between a head and its argument, as in “George worried severely about the storm.[5] The Figure 4 depicts this case.

4. The precedence relation is trivially defined when the substitution nodes are not in a dominance relation (remember that we assume a full connectedness of the left context); in case one substitution node dominates another, the leftmost substitution node is defined in relation to nodes on the respective head projections of all the substitution nodes, which by definition must be different.
5. Note that adverbs very frequently intervene between a verb and its object in Romance languages like Italian.

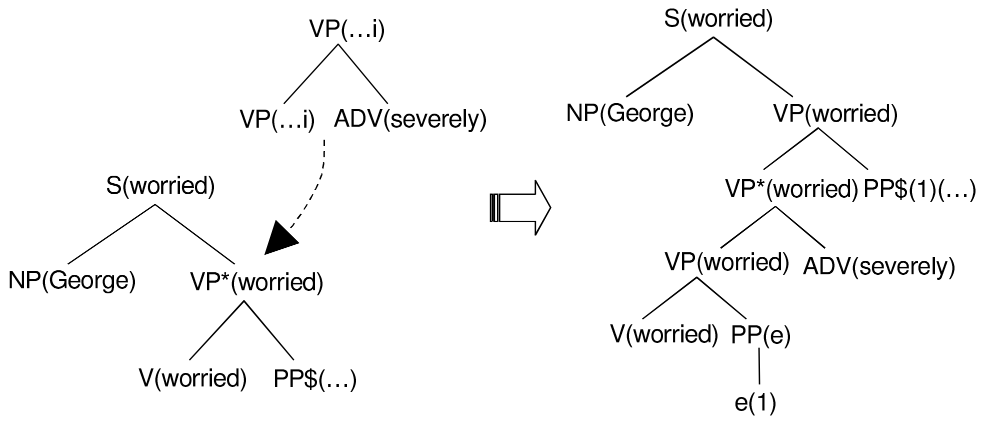


Figure 4: The modifier “severely” occurs between the head “worried” and its argument PP “about the storm”.

The attachment operations involve the redundancy checking procedure. Redundancy checking involves those substructures that are already part of the left context. Starting from the root node that matched, the redundancy checking procedure overlaps the nodes of the elementary tree onto the left context, and verifies that nodes that match are consistent. Nodes that are the results of adjoining operations are neglected during this process. Our solution is to maintain a data structure for each word inserted through an initial tree, that keeps track of the correspondences between the nodes of the original structure, and the actual instantiations of nodes in the current left context.

3. Derivation in DV-TAG

The derivation, as well as the parsing, of syntactic structures, proceeds from left to right, by combining the elementary trees anchored to the words in the string, using the operations of Substitution and Adjoining.

The left-to-right constraint implies that the derivation process proceeds through intermediate states $\mathrm{S_i}$, that exactly advance on the words of the terminal string. At the processing step i, we have a *current word* w_i and a *current state* S_{i-1}, the left context. The derivation process combines an elementary tree for w_i with S_{i-1} to produce S_i, which becomes the current state at the next step.

Each elementary tree typically introduces a few nodes headed by the lexical anchor, and a few nodes that are marked for substitution, and that need to be actually substituted. In derivation, these nodes are called *predicted nodes*. Predicted nodes are introduced both for connectivity requirements and for linguistic motivations. They are typically unheaded; in fact, usually, during the course of a derivation, they are pending non terminals that expect to be substituted by some initial tree, possibly after a number of adjoinings. It can also be the case that a predicted node is headed: such a situation occurs when the node marked for substitution already has a lexical anchor (that is, the word is part of the prediction, like in the case of subcategorized prepositions, see above). This can cause a situation where the derivation of some word at position i occurs before the derivation of some word at position j (<i), which seems in contrast with the incremental assumption. For example, when a verb subcategorizes for a preposition, the elementary tree for the verb can include the preposition too. Verb and preposition can possibly be pushed apart by the adjunction of some adverbial to the VP node. This adjunction occurs in the linear order between the verb and the preposition. Thus, the derivation of the preposition precedes the derivation of the adverbial; the reverse of the linear string order. In case the predicted nodes are headed, the terminal symbol (i.e. the lexical anchor) is checked for redundancy, when it is the current word. A node such that there are no predicted nodes in its subtree is said to be *complete*. In the final tree all nodes must be complete.

The left-to-right order constraint, combined with the standard conditions on trees implies that only a part of the left context is accessible for continuation; that is, not all the nodes can be sites of substitutions or adjunctions. Specifically, elementary trees can be attached only in the strict sequence given by the terminal string of the lexical anchors. The accessible nodes form the so-called *fringe*. In context-free incremental processing, such as left-to-right top-down parsing, the fringe corresponds to the right frontier. However, because of the presence of predicted nodes and the possibility of the adjoining operation, the fringe needs to be a bit more articulated, with nodes that can be the site of adjunction and nodes that can be site of substitution.

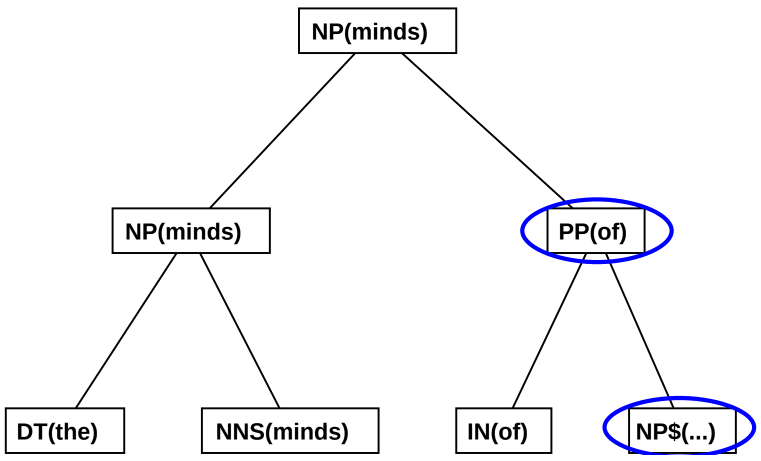


Figure 5: A fringe exemplifying the second item in the definition.

Here is the definition of the fringe in the left context S_{i-1}:

- at the beginning, the derivation process postulates a left context given by the start symbol;
- at the step i, we take the path leading from the preterminal node for the word w_{i-1} to the root:
  - if there are no nodes marked for substitution, all the nodes on this path are marked for adjunction;
  - if there are nodes marked for substitution, take the lowest common ancestor of w_{i-1} and the leftmost node marked for substitution; all the nodes on the two paths to the common ancestor are marked for adjunction.

These constraints on the fringe preserve the linear order of tree nodes with respect to word order. In the figures 5 and 6 there are two examples of fringes. Figure 5 shows the fringe at a point where a node is marked for substitution. Figure 6 shows the fringe after the substitution node has been filled. It can be seen that the number of nodes available for adjunction increases dramatically after the substitution operation.

4. Derivation of cross-serial dependencies

In the following paragraphs we will give a sketch of how the formalism can be used for creating an incremental derivation of a Dutch verb-raising sentence, which is an example of a cross-serial dependency construction. This construction is well-known as a test-bed for TAG-based systems, because it requires greater than context-free expressive power.

The example sentence is below:

(1) Jan Piet Marie zag helpen zwemmen.
Jan Piet Marie saw help swim.
(i.e. *Jan saw Piet help Marie to swim.*)

Assume that the elementary tree for a noun phrase can also include elements of the sentential structure in which the noun phrase appears (this could be seen as an analogue of *type raising* in Combinatorial Categorial Grammar (Steedman, 2000)) so that the elementary tree for "Jan" will be as in Figure 7.

The tree includes information that the noun phrase appears as the subject of a sentence whose verb subcategorizes for a clause, and the verb is extraposed (Kroch and Santorini, 1991).[6] The next word, "Piet" has an elementary tree with an identical format to that of "Jan". The left context (that is the elementary tree of "Jan") is adjoined into that of "Piet"[7], creating a new partial derived tree, which is in turn adjoined to the elementary tree for *Marie*, yielding the new partial derived tree shown in Figure 8.

6. Whether so much predicted information would be included in a psychologically plausible elementary tree is debatable. For the purposes of parsing, some forms of underspecification could be used, in which case the relation between parsing and grammatical derivations would be weakened.
7. This is non standard in TAG, the left context behaves like an auxiliary tree, and adjoins into the elementary tree of "Piet".

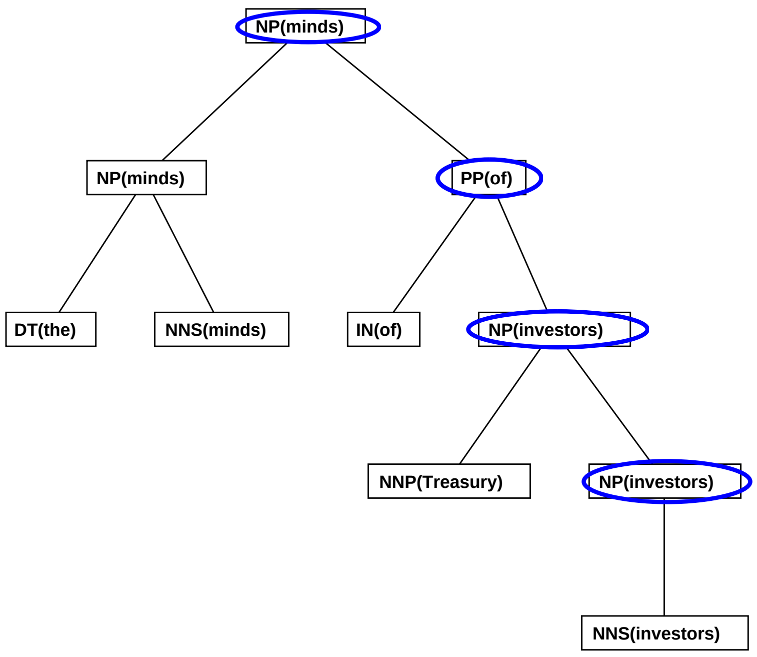


Figure 6: A fringe exemplifying the first item in the definition.

Notice that at this point, the lowest substitution node is close to the root of the tree, so, according to the definition of the fringe, adjunctions could occur anywhere in the dotted line shown on the figure.

The final three verbs of the sentence are in the respective extraposed head positions at the end of the sentence.

5. Conclusions and future extensions

In this paper, we have illustrated informally an approach that makes TAG a dynamic grammar, that is a formalism suitable for the incremental processing of natural language. The dynamic version of the TAG (DV-TAG) involves elementary trees and attachment operations that obey the left-to-right full-connectedness constraint due to incrementality.

The linguistic examples we have shown reveal the major issues of the approach. Beyond the formalization of the approach, two extensions are planned in the near future. On the theoretical side, we are going to model the incremental attachment of extraposed modifiers (such as the phrase “with blond hair” in “A man came in with blond hair”). This type of phenomenon is challenging, since it requires adjunction in two places in the left context simultaneously; one adjunction will represent the surface position of the extraposed element, and the other will represent its trace. If the trace and the extraposed element must be in the same elementary tree, then it may be necessary to add a regular expression to the elementary tree, which can match an arbitrary number of intervening nodes between the surface attachment point and the trace. An alternative solution could be the use of Synchronous TAGs. On the practical side, we aim to extract a (large scale) grammar from a treebank. The extractor will be based on the algorithms presented in (Xia, Palmer and Joshi, 2000), especially in the initial stage, where treebank trees are transformed into derived trees. The resulting grammar can then be used to build models of human language processing on a realistically large scale.

References

Berwick, R. and A. Weinberg. 1984. *The grammatical basis of linguistic performance: language use and acquisition.* MIT Press.

Frank, R. and W. Badecker. 2001. Modeling syntactic encoding with Tree Adjoining Grammar. In *Talk presentd at the CUNY conference on sentence processing.*

Joshi, A., L. Levy and M. Takahashi. 1975. Tree adjunct grammars. *Journal of the Computer and System Sciences*, 10(1):136–163.

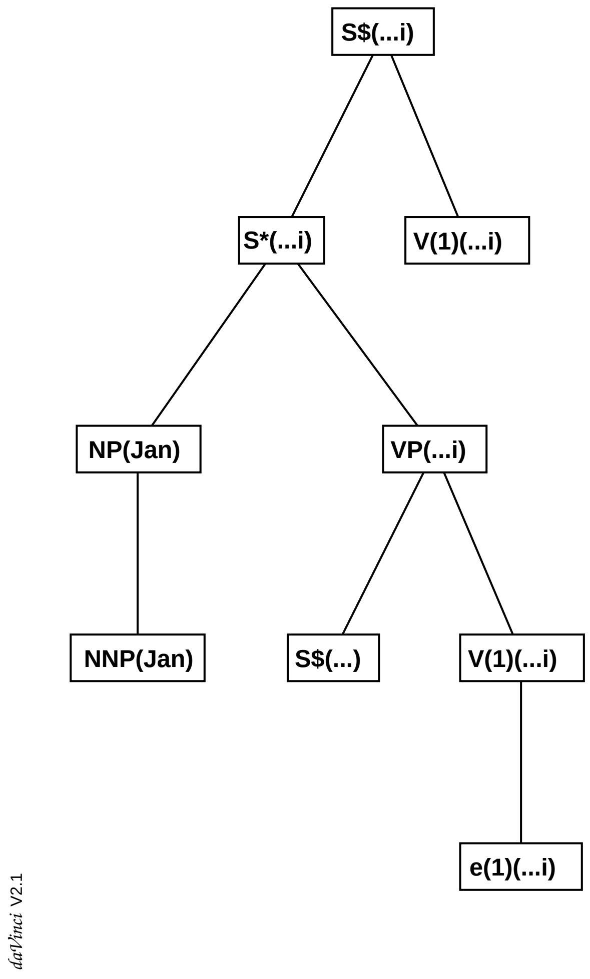


Figure 7: Elementary tree for *Jan*. Notice that the internal S node is marked for Adjunction. In the derivation of the example sentence, it is this node to be adjoined in “Piet” elementary tree.

Kroch, A. and A. Joshi. 1985. The linguistic relevance of Tree Adjoining Grammar. *CIS Technical Report MS-CIS-85-16, University of Pennsylvania.*

Kroch, A. and B. Santorini. 1991. The derived constituent structure of the West Germanic verb-raising construction. In R. Freidin, editor, *Principles and Parameters in comparative grammar*. MIT Press, Cambridge: MA, pages 269–338.

Lombardo, V. and P. Sturt. 2002. Incrementality and lexicalism: A treebank study. In S. Stevenson and P. Merlo, editors, *Lexical Representations in Sentence Processing.* John Benjamins.

Marslen-Wilson, W. 1973. Linguistic structure and speech shadowing at very short latencies. *Nature*, 244:522–533.

Milward, D. 1994. Dynamic Dependency Grammar. *Linguistics and Philosophy*, 17(6).

Phillips, C. 1998. Linear Order and Constituency. *Linguistic Inquiry*, in press.

Schabes, Y., A. Abeillé and A. K. Joshi. 1988. Parsing strategies with “lexicalized” grammars: Applications to tree adjoining grammars. In *12th International Conference in Computational Linguistics*, pages 578–583, Budapest, August.

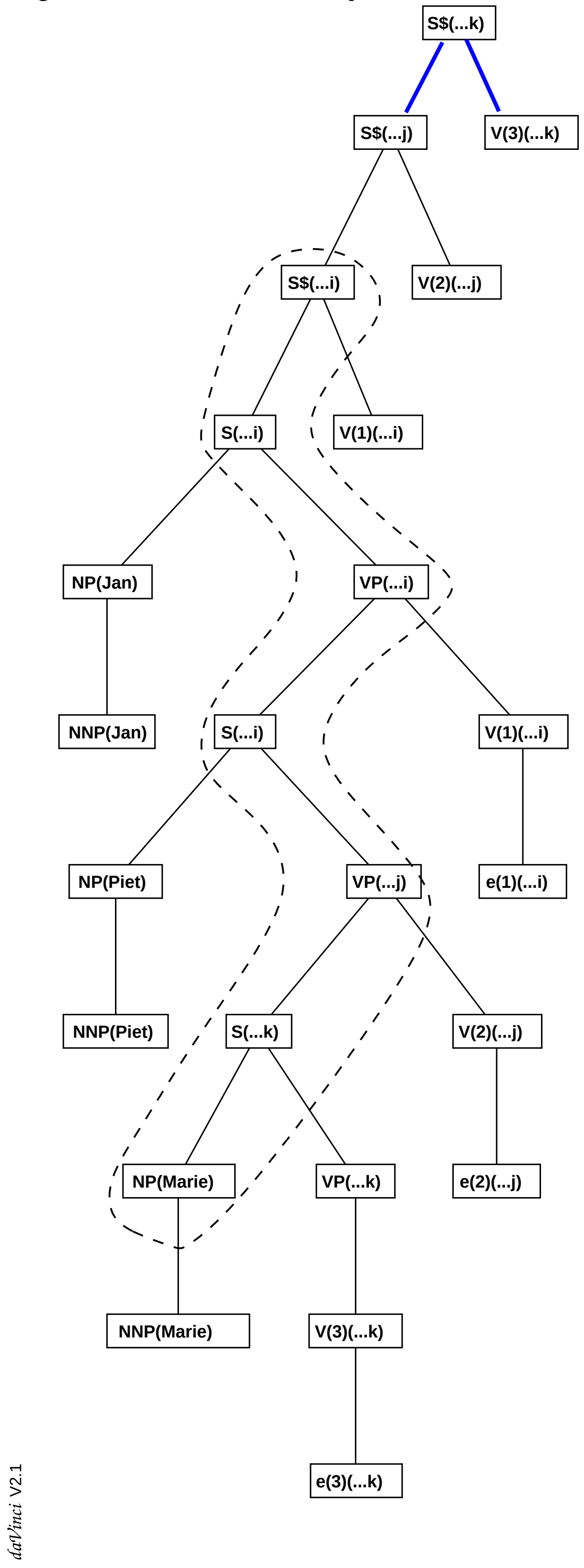
Stabler, E. P. 1994. The finite connectivity of linguistic structure. In C. Clifton, L. Frazier and K. Reyner, editors, *Perspectives on Sentence Processing.* Lawrence Erlbaum Associates, pages 303–336.

Steedman, M. J. 2000. *The syntactic process.* A Bradford Book, The MIT Press.

Thompson, H. S., M. Dixon and J. Lamping. 1991. Compose-reduce parsing. In *Proc. of 29th ACL*, pages 87–97.

Xia, F., M. Palmer and A. Joshi. 2000. A Uniform Method of Grammar Extraction and Its Applications. In *Proc. of the EMNLP 2000.*

Figure 8: Derived tree for the sequence *Jan Piet Marie*


S$(...k)
S$(...j)
V(3)(...k)
S$(...i)
V(2)(...j)
S(...i)
V(1)(...i)
NP(Jan)
VP(...i)
NNP(Jan)
S(...i)
V(1)(...i)
NP(Piet)
VP(...j)
e(1)(...i)
NNP(Piet)
S(...k)
V(2)(...j)
NP(Marie)
VP(...k)
e(2)(...j)
NNP(Marie)
V(3)(...k)
e(3)(...k)
daVinci V2.1