Competence and Performance Grammar in Incremental Processing

Vincenzo Lombardo
Dipartimento di Informatica
Università di Torino
c.so Svizzera, 185
10149, Torino, Italy
vincenzo@di.unito.it

Alessandro Mazzei
Dipartimento di Informatica
Università di Torino
c.so Svizzera, 185
10149, Torino, Italy
mazzei@di.unito.it

Patrick Sturt
Department of Psychology
University of Glasgow
58 Hillhead Street
Glasgow, G12 8QB, UK
patrick@psy.gla.ac.uk

Abstract

The goal of this paper is to explore some consequences of the dichotomy between competence and performance from the point of view of incrementality. We introduce a TAG–based formalism that encodes a strong notion of incrementality directly into the operations of the formal system. A left-associative operation is used to build a lexicon of extended elementary trees. Extended elementary trees allow derivations in which a single fully connected structure is mantained through the course of a left-to-right word-by-word derivation. In the paper, we describe the consequences of this view for semantic interpretation, and we also evaluate some of the computational consequences of enlarging the lexicon in this way.

1 Introduction

Incremental processing can be achieved with a combination of grammar formalism and derivation/parsing strategy. In this paper we explore some of the computational consequences of deriving the incremental character of the human language processor from the competence grammar. In the following paragraphs, we assume that incremental processing proceeds through a sequence of processing steps. Each step consists of a configuration of partial syntactic structures (possibly connected into only one structure) and a configuration of semantic structures (again, possibly connected into one single expression). These semantic structures result from the application of the semantic interpreter to the syntactic structures in the same processing step. Depending on the semantic rules, some syntactic structures may not be interpretable—that is, some processing steps do not involve an updating of the semantic representation. In the view we present here, competence grammar is responsible for the definition of both the set of well-formed sentences of the language and the set of possible partial structures that are yielded by the derivation process. According to this view, the performance component is responsible for other aspects of language processing, including ambiguity handling and error handling. The latter issues are not addressed in this paper.

In the psycholinguistic and computational literature, many models for incremental processing have been discussed. These models can be characterized in terms of the location of the border between competence and performance. In particular, we discuss the relative responsibility of the competence and performance components on three key areas of syntactic processing: a) the space of well-formed partial syntactic structures; b) the space of the possible configurations of partial syntactic structures at each processing step; c) the sub-space of partial structures that can actually be interpreted.

The definition of well-formedness is almost universally assigned to the competence component, whether in a direct implementation of the grammar formalism (cf. the Type Transparency hypothesis (Berwick and Weinberg, 1984)) or a compiled version of the competence grammar (e.g. LR parsing (Shieber and Johnson, 1993)).

The space of the possible configurations of partial structures refers to those partial syntactic structures that are built and stored during parsing or derivation. Different algorithms result in different possibilities for the configurations of partial structures that the parser builds. For example, a bottom–up algorithm will never build a partial structure with non–terminal leaf nodes. The standard approach is to assign this responsibility to the parsing algorithm, whether the grammar is based on standard context-free formalisms (Roark, 2001), on generative syntactic theories based on a context-free backbone (Crocker, 1992), or on categorial approaches, like e.g. Combinatory Categorial Grammar (CCG – (Steedman, 2000)). A different method is to assign this responsibility to the competence component. In this case the space of

possible configurations of partial structures is constrained by the grammatical derivation process itself, and the parsing algorithm needs to be aligned with these requirements. This approach is exemplified by the works of Kempson et al. (2000) and Phillips (2003), who argue that many problems in theoretical syntax, like the definition of constituency, can be solved by extending this responsability to the competence grammar.

This issue of constituency is also relevant in the third key area, which is the definition of the space of interpretable structures. The assignment of responsibility with respect to current approaches usually depends on the implementation of the incremental technique. Approaches based on a coupling of syntactic and semantic rules in the competence grammar (Steedman, 2000; Kempson et al., 2000) adhere to the so-called *Strict Competence Hypothesis* (Steedman, 2000), which constrains the interpreter to deal only with grammatical constituents, so the responsibility for deciding the interpretable partial structures is assigned to competence[1]. In contrast, approaches that are based on competence grammars that do not include semantic rules, like CFG, implement semantic interpreters that mimic such semantic rules (Stabler, 1991), and so they assign the responsibility for deciding the interpretable partial structures to performance.

In this paper we explore the empirical consequences of building a realistic grammar when the formalism constrains all these three areas, as is the case with Kempson et al. (2000) and Phillips (2003). The work relies upon the Dynamic Version of Lexicalized Tree Adjoining Grammar (DV–TAG), introduced in (Lombardo and Sturt, 2002b), a formalism that encodes a *dynamic grammar* (cf. (Milward, 1994)) in LTAG terms (Joshi and Schabes, 1997). The consequence of encoding a dynamic grammar is that the configurations of partial structures discussed above are limited to fully connected structures, that is no disconnected structures are allowed in a configuration. In particular, the paper focuses on the problem of building a realistic DV–TAG grammar through a conversion from an LTAG, in order to maintain the linguistic significance of elementary trees while extending them to allow the full connectivity.

[1]Notice that these approaches may, however, differ in the time-course with which semantic rules are applied in the interpreter, and this issue depends directly on the space of configurations of partial structures discussed above

2 Dynamic Version of Tree Adjoining Grammar

This section reviews the major aspects of the Dynamic Version of Tree Adjoining Grammar (DV–TAG), with special reference to similarities and differences with respect to LTAG.

Dynamic grammars define well-formedness in terms of states and transitions between states. They allow a natural formulation of incremental processing, where each word w_i defines a transition from State_{i-1}, also called the *left context*, to State_i (Milward, 1994). The states can be defined as partial syntactic or semantic structures that are "updated" as each word is recognized; roughly speaking, two adjacent states can be thought of as two parse trees before and after the attachment of a word, respectively. The derivation process proceeds from left to right by extending a fully connected left context to include the next input word.

Like an LTAG (Joshi and Schabes, 1997), a Dynamic Version of Tree Adjoining Grammar (DV–TAG) consists of a set of elementary trees, divided into initial trees and auxiliary trees, and attachment operations for combining them. Lexicalization is expressed through the association of a lexical *anchor* with each elementary tree. The anchor defines the semantic content of the elementary tree: the whole elementary tree can be seen as an *extended projection* of the anchor (Frank, 2000). LTAG is said to define an *extended domain of locality* –unlike context-free grammars, which use rules that describe one–branch deep fragments of trees, TAG elementary trees can describe larger structures (e.g. a verb, its maximal S node and subject NP node).

In figures 1(a) and 2(a) we can see the elementary trees for a derivation of the sentence *Bill often pleases Sue* for LTAG and DV–TAG respectively. Auxiliary trees in DV–TAG are split into *left auxiliary trees*, where the lexical anchor is on the left of the foot node, and *right auxiliary trees*, where the lexical anchor is on the right of the foot node. The tree anchored by *often* in fig. 2(a) is a left auxiliary tree.

Non-terminal nodes have a distinguished head daughter, which provides the lexical head of the mother node: unlike in LTAG, each node in the elementary trees is augmented with a feature indicating the lexical head that projects the node. This feature is needed for the no-

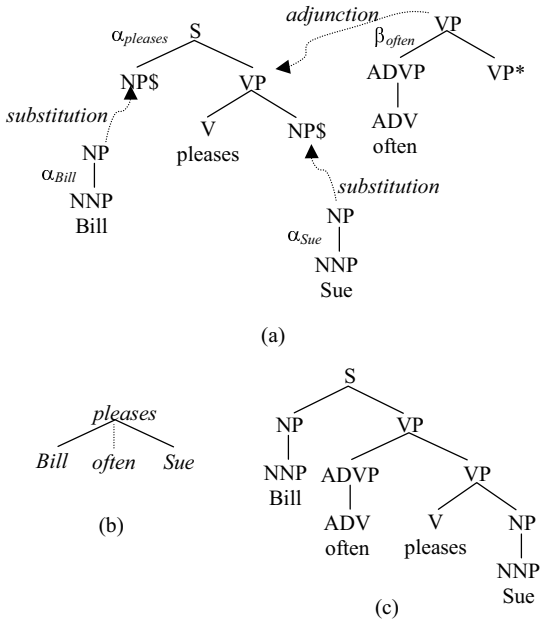

Figure 1: The LTAG derivation of the sentence *Bill often pleases Sue*.

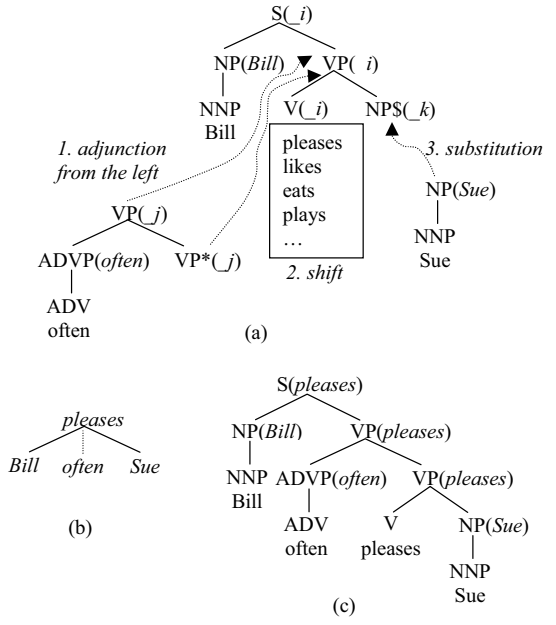

Figure 2: The DVTAG derivation of the sentence *Bill often pleases Sue*.

tion of *derivation–dependency tree* (see below). If several unheaded nodes share the same lexical head, they are all co-indexed with a head variable (e.g. _*i* in the elementary tree anchored by *Bill* in figure 2(a)); the head variable is a variable in logic terms: _*i* will be unified with the constant ("lexical head") *pleases*.

In both LTAG and DV–TAG the lexical anchor does not necessarily provide the head feature of the root of the elementary tree. This is trivially true for auxiliary trees (e.g. the tree anchored *often* in figure 1(a) and figure 2(a)). However, in DV–TAG this can also occur with initial trees (e.g. the tree anchored by *Bill* in figure 2(a)), because initial trees can include not only the head projection of the anchor, but also other higher projections that are required to account for the full connectedness of the partial parse tree. The elementary tree anchored by *Bill* is linguistically motivated up to the NP projection; the rest of the structure depends on connectivity. These extra nodes are called *predicted nodes*. A predicted preterminal node is referred by a set of lexical items. In the section 3 we illustrate a method for building such extended elementary trees.

The derivation process in LTAG and DV–TAG builds a *derived tree* by combining the elementary trees via some operations that are illustrated below. DV–TAG implements the incremental process by constraining the *derivation process* to be a series of steps in which an elementary tree is combined with the partial tree spanning the left fragment of the sentence. The result of a step is an updated partial structure. Specifically, at the processing step i, the elementary tree anchored by the i-th word in the sentence is combined with the partial structure spanning the words from 1 to $i-1$ positions; the result is a partial structure spanning the words from 1 to i. In contrast, LTAG does not pose any order constraint on the derivation process, and the combinatorial operations are defined over pairs of elementary trees. In DV–TAG the derivation process starts from an elementary tree anchored by the first word in the sentence and that does not require any attachment that would introduce lexical material on the left of the anchor (such as in the case that a Substitution node is on the left of the anchor). This elementary tree becomes the first left context that has to be combined with some elementary tree on the right.

Since in DV–TAG we always combine a left context with an elementary tree, the number of attachment operations increases from two in LTAG to six in DV–TAG. Three operations (substitution, adjunction from the left and adjunction from the right) are called *forward operations* because they insert the current elemen-

tary tree into the left context; two other operations (inverse substitution and inverse adjunction) are called *inverse operations* because they insert the left context into the current elementary tree; the sixth operation (shift) does not involve any insertion of new structural material.

The first operation in DV–TAG is the standard LTAG *substitution*, where some elementary tree replaces a substitution node in another tree structure (see fig. 2(a)).

Standard LTAG adjunction is split into two operations: *adjunction from the left* and *adjunction from the right*. The type of adjunction depends on the position of the lexical material introduced by the auxiliary tree with respect to the material currently dominated by the adjoined node (which is in the left context). In figure 2(a) we have an adjunction from the left in the case of the left auxiliary tree anchored by *often*.

Inverse operations account for the insertion of the left context into the elementary tree. In the case of *inverse substitution* the left context replaces a substitution node in the elementary tree; in the case of *inverse adjunction*, the left context acts like an auxiliary tree, and the elementary tree is split because of the adjoining of the left context at some node. In (Lombardo and Sturt, 2002b) there is shown the importance of the latter operation to obtain the correct dependencies for cross-serial Dutch dependencies in DV–TAG.

Finally, the *shift* operation either scans a lexical item which has been already introduced in the structure or derives a lexical item from some predicted preterminal node.

It is important to notice that, during the derivation process, not all the nodes in the left context and the elementary tree are accessible for performing some operation: given the $i-1$-th word in the sentence we can compute a set of accessible nodes in the left context (the *right fringe*); also, given the lexical anchor of the elementary tree, that in the derivation process matches the i-th word in the sentence, we can compute a set of accessible nodes in the elementary tree (the *left fringe*).

At the end of the derivation process the left context structure spans the whole sentence, and is called the *derived tree*: in the figures 1(c) and 2(c) there are the derived trees for *Bill often pleases Sue* in LTAG and DV–TAG respectively.

A key device in LTAG is the *derivation tree* (fig. 1(b)). The derivation tree represents the history of the derivation of the sentence: it describes the substitutions and the adjoinings that occur in a sentence derivation through a tree structure. The nodes of the derivation tree are identifiers of the elementary trees, and one edge represents the operation that combines two elementary trees. Given an edge, the mother node identifies the elementary tree where the elementary tree identified by the daughter node is substituted in or adjoined to, respectively. The derivation tree provides a factorized representation of the derived tree. Since each elementary is anchored by a lexical item, the derivation tree also describes the syntactic dependencies in the sentence in the terms of a *dependency–style* representation (Rambow and Joshi, 1999) (Dras et al., 2003).

The notion of derivation tree is not adequate for DV–TAG, since the elementary trees contain unheaded predicted nodes. For example, the elementary tree anchored by *Bill* actually involves two anchors, *Bill* and *pleases*, even if the latter anchor remains unspecified until it is scanned/derived in the linear order. We introduce a new word–based structure that represents syntactic dependencies, namely a *derivation-dependency tree*.

A derivation-dependency tree is a *head-based* version of the derivation tree. Each node in an elementary tree is augmented with the lexical head that projects that node. The derivation-dependency tree contains one node per lexical head, and a lexical head dominates another when the corresponding projections in the derived tree stand in a dominance relation. Each elementary tree can contain only one overtly marked lexical head, that represents the semantic unit, but the presence of predicted nodes in the partial derived tree corresponds to *predicted heads* in the derivation-dependency tree. In figure 3 is depicted the evolution of the derivation–dependency tree for the sentence *Bill often pleases Sue*.

The DV–TAG derivation process requires the full connectivity of the left context at all times. The extended domain of locality provided by LTAG elementary trees appears to be a desirable feature for implementing full connectivity. However, each new word in a string has to be connected with the preceding left context, and there is no *a priori* limit on the amount of structure that may intervene between that word and the preceding context. For example, in a DV–TAG derivation of *John said that tasty apples*

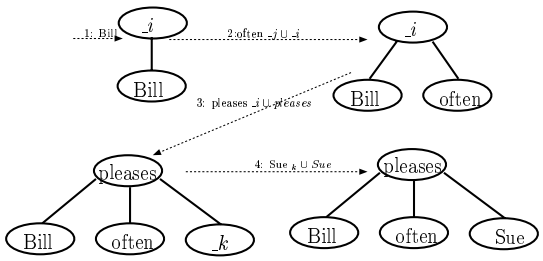

Figure 3: The DVTAG derivation of the sentence *Bill often pleases Sue*.

were on sale, the adjective *tasty* cannot be directly connected with the S node introduced by *that*; there is an intervening NP symbol that has not yet been predited in the structure. Another example is the case of an intervening modifier between an argument and its predicative head, like in the example *Bill often pleases Sue* (see figure 2), where in order to scan *often* we need a VP adjunction node that the NP projection cannot introduce. So, the extended domain of locality available in LTAG has to be further extended. In particular, some structures have to be predicted as soon as there is some evidence from arguments or modifiers on the left. In other approaches this extension is implemented via top-down predictions (see e.g. Roark (2001)) during the parsing process. This can lead to a high number of combinations that raise the degree of local ambiguity in the derivation process. In fact, in the case of Roark (2001), the method to reduce this problem has been to use underspecification in the right part of the cf rules.

In the remainder of this paper we address the issue of building a wide coverage DV–TAG grammar where elementary trees extend the domain of locality given by the argumental structure, and we provide an empirical evaluation of the possible combinatorial problems that can raise with such extended structures.

3 Building a DV-TAG lexicon

The method used to build a wide coverage DV–TAG grammar is to start with an existing LTAG grammar, and to extend the elementary trees through a closure of a left–associative operation.

First, the LTAG elementary tree nodes have to be augmented with the lexical head information through a percolation procedure that takes into account the syntactic projections.

Then, the elementary trees must be extended to account for the full connectivity. Given that one step of the derivation process is a combination of a left context and an elementary tree, we have that the rightmost symbol of the left context and the leftmost anchor of the elementary tree (the current input word) must be adjacent in the sentence. However, it is possible (as we have illustrated above) that the left context and the elementary tree cannot be combined through none of the five DV–TAG operations. But if the combination between the left context and the elementary tree can occur once we assume some intervening structure, we can build a *superstructure* that includes the elementary tree and extends it until either the left context can be inserted in the left fringe of the new superstructure or the new superstructure can be inserted in the right fringe of the left context. In building the superstructures, we require that the linguistic dependencies posed by the LTAG elementary trees over the lexical heads in the derivation/dependency tree are maintained, in order not to disrupt the semantic interpretation process.

Since no new symbol can intervene between the rightmost symbol of the left context and the leftmost anchor of the elementary tree (the current input word), the elementary tree must be extended in ways that do not alter such linear order of the terminal symbols. This means that the elementary tree must be extended without introducing any further structure that can in turn derive terminal symbols on the left of the leftmost anchor. In order to satisfy such a constraint, the elementary tree has to be *left–anchored*, i.e. the leftmost symbol of the elementary tree, but possibly the foot node in case of a right auxiliary tree, is an anchor. Then, the operation that extends the left–anchored elementary trees is the *left association*. The left association starts from the root of the elementary tree and combines it with another elementary tree on the right through through either inverse operation (see above)[2]; this combination is iterated as far as possible through a *transitive closure* of left association (see below). All the combinations are stored in the extended lexicon.

Since the individual elementary trees that form a superstructure through left association are not altered in this process, linguistic depen-

[2]There are some similarities between left association and the CCG type raising operation (Steedman, 2000), because in both cases some (root) category X is "raised" to some higher category Y.

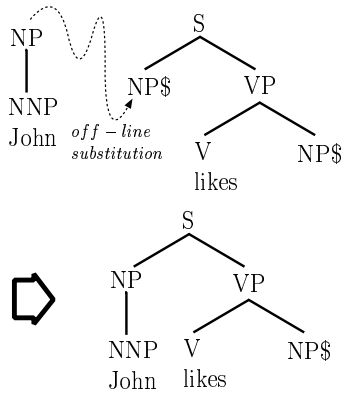

Figure 4: Example of the left association operation: the trees on the top are, respectively, the *Base tree* and the *Raising tree*; the tree on the bottom is the *Raised tree*

dencies are kept unchanged. Left association can be performed during the parsing/derivation process (i.e. on–line) or with the goal to extend the lexicon (i.e. off–line). Since we are exploring the consequences of increasing the role of the competence grammar, we perform this operation off–line (see the next section).

Each left association operation takes in input two trees: a left–anchored *Base tree* and a *Raising tree*, and produces in output a new left–anchored *Raised tree*. A Base tree α can be any left–anchored elementary tree or a Raised tree. A Raising tree is any elementary tree β that allows α to combine on its left via either inverse substitution or adjunction. A Raised tree is a tree such that α has been attached to β according to inverse substitution or inverse adjunction.

The application of the transitive closure of left association occurs with the termination condition of minimal recursive structure, that is the non repetition of the root category in the sequence of Raising trees (henceforth *root sequence*). So, if the original Base tree or some Raising tree already employed have a root X, we cannot use a Raising tree rooted X anymore for the same superstructure.

Considering that LTAG is a lexicalized formalism, we immediately realize that a superstructure is multiply anchored. As an example, consider the left association illustrated in figure 4: we substitute the tree anchored by *John* into the tree anchored by *likes*, yielding a larger elementary structure multiply anchored by *John* and *likes* at the same time (the lexical head information for each node has been omitted).

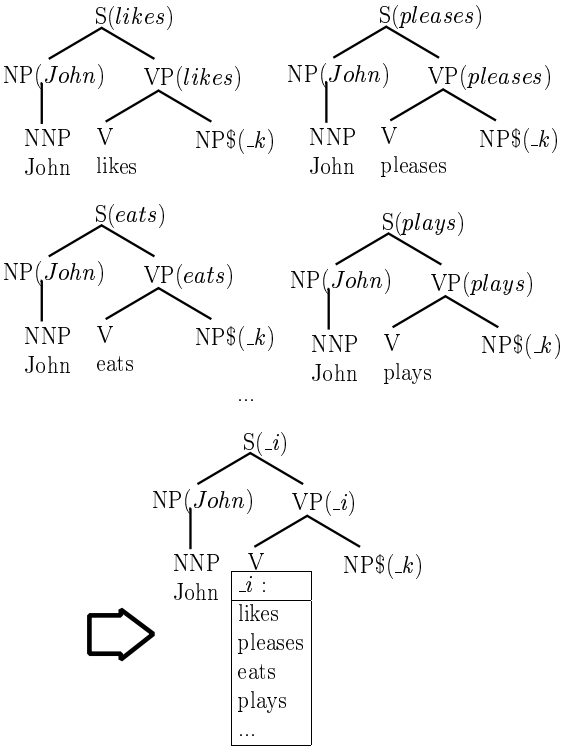

Figure 5: Schema of the left association operation, followed by the factorization in template trees.

Multiple anchoring, when not linguistically motivated like in the case of idioms or specific subcategorization constraints, leads to some potential problems. The first is the theoretical issue of semantic compositionality, because the superstructures do not reflect the incremental process in the semantic composition once words are not the minimal semantic units anymore (as assumed in LTAG). The second is a practical issue of duplicating the stored information for all the verbs sharing the predicate–argument structure. For example, in the previous example, all the transitive verbs have a tree structure identical to the elementary tree of *likes* (see fig. 5).

These two problems can be solved by introducing the notion of template, already present in the practical implementations of wide coverage LTAG systems (Doran et al., 2000). A *tree template* is a single elementary tree that represents the set of elementary trees sharing the same structure except for the lexical anchor: one single structure is referred to by pointers from the word list. All the equal tree structures that have the same leftmost anchor after are represented as a single template, where only

the leftmost anchor is lexically realized; all the other anchors are replaced by variables with *referring word lists* that explicitly state the range of variation of the variables themselves. A variable replaces each occurrence of the lexical item in the original elementary structure: once a shift operation matches the current input word with one of the words in the associated list, we also have to unify the lexical head variables that augment non terminal symbols with the current input word. For instance, on the bottom of figure 5, there is the template obtained by the left association of the elementary tree anchored by *John* with all the equal elementary trees of transitive verbs.

A further problem is a possible combinatorial explosion of the size of the lexicon: this problem has to be tackled in an empirical way on a wide coverage grammar (it could be that a large number of the theoretically possible combinations do not occur in practice; in fact, empirical work by (Lombardo and Sturt, 2002a) indicates that there is an empirical bound on the size of expanded elementary trees necessary to maintain connectedness).

4 Empirical tests

In order to estimate whether the combinatorial explosion has a dramatic effect on the lexicon we have run two tests, implementing the transitive closure of the left association. The first test was performed on a realistic grammar from the XTAG system (Doran et al., 2000), and the second test was performed on an automatically extracted grammar from an Italian treebank (Mazzei and Lombardo, 2004).

In the implementation of the recursive procedure, the left-association operation takes as input two templates: a left-anchored template, that we call *base template*, and another template, that cannot be a left–anchored template, that we call *raising template*. In every step of the algorithm, the base template is taken from the subset of left-anchored templates and the raising template is picked from the whole lexicon. Since the algorithm builds only left–anchored templates, the output template is inserted in the left-anchored subset.

The grammar used in the first test has 628 tree templates representing one half of the handwritten XTAG grammar, with the same distribution of template families as the overall XTAG grammar. This size is a realistic grammar size (consider that the XTAG lexicon, the widest LTAG grammar existing for English, 1227 templates). 140 out of 628 were left–anchored templates, and the transitive closure from these base templates produces 176,190 raised templates, with a maximum of 7 left associations, and a distribution of trees that reaches its maximum at 4 left associations (140 base templates, 3,033 twice raised templates, 24,855 three-times raised, 62,970 four times raised, 59,908 five times, 22,454 six times, 2,970 seven times). Similar distributional results have been produced for subsets of the grammar and with restrictions on root categories. The number of raised templates drastically reduces when we forbid raising to verbal projections (S, VP and V), thus cutting one of major sources of the explosion. In this case we go from 717 non verbal base templates in the XTAG grammar to only 24,468 raised templates, again with a maximum of 7 raisings (notice that the base lexicon is larger than the test above).

In the second test we used a LTAG grammr extracted from the 45,000 word TUT (Turin University Treebank). The number of extracted tree templates was 1283. In this case, as the grammar was relatively large, we decided to impose an extra condition on the closure for left association procedure. We estimated the maximum number of trees that need to be composed to create any one left associated tree. This was done by inspecting the derivation trees for each sentence of the treebank, and looking at the leftmost child of each level of the derivation tree. It was found that no left-associated tree needed to be composed of more than three elementary trees, in order to create a covering DV–TAG for the treebank. This replicates a previous result of Lombardo and Sturt (2002a). Moreover, of the 800 mathematically possible root sequences[3] only 67 were present in the treebank. We decided to allow left association only for root sequences that actually appeared in the treebank. This resulted in a total of 706,866 left associated trees. Of these, 988 were base templates, 87,245 were raised twice, and 618,654 were raised three times.

The combinatorial explosion seen in the two experiments suggests the use of underspecification techniques before applying DV–TAG in a realistic setting (see Roark (2001) for one method applied to Context Free Grammar). However, in order to estimate the amount of

[3] In the TUT treebank we have 27 non-terminal symbols, and 20 possible root categories.

ambiguity that can arise in a parsing process we need to refer to some specific parsing model. Also selective strategies on categories and restriction to empirically observed root sequences can be effective. However, in order to make a full evaluation of these strategies, it is desirable to perform further coverage tests.

5 Conclusion

This paper has explored some consequences of building an incremental grammar that constrains the possible configurations of partial structures and the possible interpretable partial structures.

We have introduced a TAG–based formalism that encodes a strong notion of incrementality directly into the operations of the formal system.

A left-associative type-raising operation has been used to build a DV–TAG realistic lexicon from a LTAG lexicon. The left–association operation can be viewed as a sort of chunking of linguistic knowledge: this chunking could be useful in the definition of a language model in which specific combinations of elementary trees that have been successful in the past experience become part of the lexicon.

As shown in the empirical tests ofn English and Italian, the left–association closure can lead to severe computational problems that suggest the adoption of some form of underspecification.

References

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

M. W. Crocker. 1992. *A Logical Model of Competence and Performance in the Human Sentence Processor*. Ph.D. thesis, Dept. of Artificial Intelligence, University of Edinburgh, UK.

C. Doran, B. Hockey, A. Sarkar, B. Srinivas, and F. Xia. 2000. Evolution of the xtag system. In A. Abeillé and O. Rambow, editors, *Tree Adjoining Grammars*, pages 371–405. Chicago Press.

M. Dras, D. Chiang, and W. Schuler. 2003. On relations of constituency and dependency grammars. *Language and Computation*, in press.

R. Frank. 2000. Phrase structure composition and syntactic dependencies. Unpublished Manuscript.

A. Joshi and Y. Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, pages 69–123. Springer.

R. Kempson, W. Meyer-Viol, and D. Gabbay. 2000. *Dynamic Syntax: the Flow of Language Understanding*. Blackwell, Oxford, UK.

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

V. Lombardo and P. Sturt. 2002b. Towards a dynamic version of tag. In *TAG+6*, pages 30–39.

A. Mazzei and V. Lombardo. 2004. Building a large grammar for italian. In *LREC04*. in press.

David Milward. 1994. Dynamic dependency grammar. *Linguistics and Philosophy*, 17:561–604.

C. Phillips. 2003. Linear order and constituency. *Linguistic Inquiry*, 34:37–90.

O. Rambow and A. Joshi. 1999. A formal look at dependency grammars and phrase structure grammars, with special consideration of word-order phenomena. In *Recent Trends in Meaning-Text Theory*, pages 167–190. John Benjamins, Amsterdam and Philadelphia.

B. Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

S. M. Shieber and M. Johnson. 1993. Variations on incremental interpretation. *Journal of Psycholinguistic Research*, 22(2):287–318.

E. P. Stabler. 1991. Avoid the pedestrians' paradox. In R. C. Berwick, S. P. Abney, and C. Tenny, editors, *Principle-based parsing: computation and psycholinguistics*, pages 199–237. Kluwer, Dordrecht: Holland.

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