Research on Language and Computation **00:** 1–28, 2007.

Dynamic TAG and lexical dependencies

A. MAZZEI (mazzei@di.unito.it)
Dip. Informatica, Università di Torino

V. LOMBARDO (vincenzo@di.unito.it)
Dip. Informatica, Università di Torino

P. STURT (patrick.sturt@ed.ac.uk)
Psychology, School of Philosophy, Psychology and Language Sciences, University of Edinburgh

September 26, 2007

Abstract. Incrementality is a widely held assumption that constrains the language processor to parse the input words from left to right. In this paper we describe the basic features of a constituency-based dynamic grammar based on Tree Adjoining Grammar, which natively fulfills a strict version of incrementality. We focus on the linguistic appeal of the formalism, analyzing a number of linguistic phenomena and exploiting the relation between dynamic constituency analysis and lexical dependencies.

1. Introduction: motivation for incremental model of language

Incrementality is a basic feature of the human language processor. There is a considerable amount of objective data that humans build-up semantic interpretation before reaching the end of the sentence (Marslen-Wilson, 1973; Kamide et al., 2003), and there is also evidence to support the idea that such incremental interpretation is based on fast, left-to-right construction of syntactic relations⋆ (see, for example, (Aoshima et al., 2004; Sturt and Lombardo, 2005)). In order to grasp which components of the syntactic parser contribute to incrementality, we can refer to a general *anatomy* proposed in (Steedman, 2000). In this schema we can distinguish three main components: 1. a grammar, 2. a parsing algorithm and 3. an oracle. The grammar is a static object, usually defined in a declarative form, which contains competence-related information about the grammaticality of the sentences. The parsing algorithm can be divided into two parts: a parsing strategy specifying the order of application of the grammar rules (e.g. left-to-right, bottom-up, top-down), and a memory organization strategy, which specifies how the parser memorizes the partial results (e.g. depth-first, backtracking, dynamic programming). The oracle deals with the ambiguity of

⋆ Our discussion focuses on parsing. Similar arguments have been made for the generation process (Kempson et al., 2000).

syntactic analysis, and ranks the possible syntactic alternatives by applying a preference scheme, which may be based on a rule-based or probabilistic strategy.

In most incremental parsing models, the competence grammar is usually in the form of a a standard formalism such as a context-free grammar (Roark, 2001), a generative syntactic theory (Crocker, 1992; Joshi and Schabes, 1997), or a categorial approach, e.g. Combinatory Categorial Grammar (CCG, (Steedman, 2000)). As a consequence, the parsing strategy is the component that totally specifies the incremental nature of the syntactic process (i.e., incrementality is not part of the competence knowledge). However there exist a few approaches that incorporate the parsing strategy, i.e. the order of application of the rules of the grammar, into the grammar component. Phillips (1996,2003) proposed a model of grammar in which the parsing strategy is part of the competence grammar. His fundamental hypothesis (PiG, *parser is grammar*) is that a grammar is a formal generative device that specifies the well-formedness of structures in terms of their computation. In this model the notion of constituency is variable during the analysis of the sentence: it is possible for a fragment of the sentence that is a constituent at some point in the derivation not to be a constituent after the processing of a subsequent word. Phillips' fundamental hypothesis ties the notion of constituency to the parser's computation. Constituency depends on the parsing state. According to this model, the only difference between the parser and the grammar is that the former has to take into account the finiteness of the resources, while the latter does not.

The framework of Dynamic Syntax (Kempson et al., 2000) can also be interpreted as including an incremental parsing strategy as part of the competence grammar. Here, the grammar consists of a set of procedural rules for building the semantic structure in response to each word in the unfolding input string. The form of the rules constrains the analysis of the string to be strictly left-to-right and word-by-word, and Kempson et al (2000) show how this restriction can be used to explain a number of syntactic phenomena.

Milward (1994) proposes a dependency-based account of incrementality in terms of *dynamic grammars*, in which the syntactic analysis is viewed as a dynamic process. In this model the syntactic process is a sequence of transitions between adjacent syntactic states S_i and S_{i+1}, which are reached while the processor moves from left to right on the input string. The syntactic state contains all the syntactic information about the fragment already processed. Similarly to Phillips' model, the parsing strategy is a part of the dynamic competence grammar. Milward offers a competence account for non-constituency coordination: in Milward's case, two fragments x and y of the sentence are grammatical conjuncts if both allow the processor to reach the same end state from the same starting state.

In this paper we present a novel grammatical formalism that adheres to the hypothesis that *the parsing strategy is totally determined by the competence grammar*. The formalism is a dynamic version of Tree Adjoining Grammars (TAG), called DVTAG.

Parsing strategies can be defined in terms of the order with which the parser builds the nodes and branches of a tree during the left-to-right processing of a string. Within the framework of Generalized Left Corner Parsing (Demers, 1977), each context-free rule has an *announce point*, which determines which of the symbols on the rule's right hand side (RHS) need to have been enumerated before the mother node is enumerated. In a pure top-down (recursive descent) strategy, the announce point is before the first RHS symbol, so that the strategy enumerates the mother node before any of the daughter nodes. In contrast, in the pure bottom-up strategy (shift-reduce), the announce point is after the final RHS symbol, so that the mother node is enumerated only after all daughter nodes have been enumerated. In the left-corner strategy, the announce point is immediately after the first RHS symbol, so that the mother node is enumerated immediately after the first daughter node, but before any of the remaining daughter nodes.

Abney and Johnson (1991) considered the top-down, bottom-up and left-corner strategies in terms of the memory resources required to process different types of structures (see also Resnik, 1992) proposed to model the difficulty of the syntactic analysis by means of the memory resources required to store the disconnected partial structures built during syntactic analysis. It is well known that increasing degrees of center embedding lead to increasing processing difficulty (e.g. *The rat that the cat that the dog chased scared died*), while such difficulty is not experienced for left embeddings (e.g. *John's mother's sister's shoes*) or right embeddings (e.g. *I thought that John said that Bill arrived*). Abney and Johnson (1991) compared the memory profiles for top-down, bottom-up and left corner strategies, and concluded that the left-corner strategy showed the most suitability to model the difficulty of centre embeddings compared with left and right embeddings. The reason for this is that top-down strategies require unbounded memory resources for processing left branching structures, while bottom-up strategies require unbounded memory resources for processing right-branching structures. In contrast, left-corner strategies combine top-down and bottom-up processing—the leftmost daughter of each node in the tree is processed bottom-up, after which the remaining daughters are projected top-down. The result of this interplay of top-down and bottom-up processing is that neither left embeddings nor right embeddings require unbounded memory resources, but center embeddings do. Some of the parsing strategies (such as head-driven or shift-reduce strategies) involve systematic delays in structure building; for example, head-driven parsing strategies require a head daughter to be processed before the mother node is built. However, there

is evidence against many of these delay-based strategies, from experiments which show that syntactic or thematic relations are in fact available before the time that would be predicted according to head-driven or bottom-up strateiges. For example, Aoshima et al (2004) show that in certain Japanese sentences, the unbounded dependency between an extracted dative noun phrase and its trace position is formed before the appearance of the governing verb, *contra* the predictions of head-driven parsing strategies. Similarly, in Japanese, Kamide et al (2003) show that thematic roles are assigned, and partial interpretations made, for a sequence of case marked noun phrases, before the head verb appears. Sturt and Lombardo (2005) show that the second conjunct of a coordinated verb phrase can participate in c-command relations with the sentential subject before the VP is complete, *contra* the predictions of bottom-up strategies.

Lombardo and Sturt (2002a) argued that the most parsimonious account of data such as these is to assume that each new word is *connected* to the syntactic structure as soon as the word is perceived. This property of the syntactic processor, which we call *strong connectivity*, has been defined by Stabler: *people typically incorporate each (overt) word into a single, totally connected syntactic structure before any following words; incremental interpretation is achieved through the interpretation of this single syntactic structure* (Stabler, 1994). The formalism which we present here, DVTAG, relies on strong connectivity for the parsing strategy, and this simplifies the syntax-semantics interface.*

DVTAG belongs to the TAG family, i.e. is a tree-based formalism that employs the adjoining operation (Joshi et al., 1975). The formalism has origins in the model of Sturt (1997), where the adjoining was used in a monotonic architecture to model reanalysis, and in Sturt and Lombardo (2005) where the adjoining mechanism is proposed to model VP coordination. An operation similar to adjoining was also used for similar purposes in Thompson et al. (1991). The adjoining operation allows for a potentially complete constituent to be modified by the attachment of new material in the middle of the original structure. In practice, this feature of the TAG formalism allows one to model a high degree of incrementality in the processing of post-modifiers and coordination.

In Lombardo and Sturt (2002b); Lombardo et al. (2004); Mazzei et al. (2005), we have investigated about the theoretical foundation and the generative power of DVTAG, the focus of this paper is on the linguistic appealing of the formalism. The structure of the paper is the following one. In Section 2 we give a formal definition of DVTAG, and we describe the DVTAG mech-

* It should be pointed out that strong incrementality at the syntactic level is not necessarily required for word-by-word incremental interpretation, as pointed out by (Shieber and Johnson, 1993). However, the alternative requires the semantic interpreter to have access to the parser's internal state.

anism to derive lexical dependencies. In section 3 we discuss the linguistic adequacy of DVTAG by analyzing some well known linguistic examples from the TAG literature. Finally, in Section 4 we give some conclusions.

2. Incrementality and DVTAG

DVTAG is a constituency-based tree-rewriting dynamic grammar that fulfills the strong connectivity hypothesis. Similarly to LTAG, a DVTAG consists of a set of elementary trees and a number of attachment operations for combining them (Joshi and Schabes, 1997). DVTAG fulfills the strong connectivity hypothesis 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 called a *left-context*. Specifically, at 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 position 1 to position $i-1$. The result is a partial structure spanning the words from 1 to i.

Unlike LTAG, each node in the elementary tree is augmented with a feature indicating the lexical head that projects that node: we call this feature *head-feature*. If several unheaded nodes share the same lexical head, they are all co-indexed with a variable in the head-feature (e.g. $\_v_3$ in the elementary tree anchored by *often* in Fig. 2). During the derivation process not all the nodes in the left-context and in the elementary tree are accessible for applying operations. To fulfill strong connectivity we have to prevent the insertion of the lexical material to the left of the input words already scanned. Given the $i-1$-th word in the sentence, we compute a set of accessible nodes in the left-context (the *left-context fringe*); also, given the lexical anchor of the elementary tree, which in the derivation process matches the i-th word in the sentence, we compute a set of accessible nodes in the elementary tree (the *elementary tree fringe*). To take this issue into account, the elementary trees in DVTAG are dotted trees, i.e. a pair $\langle \gamma, i \rangle$ consisting of a tree γ and an integer i denoting the accessible fringe* of the tree. Now we first give some details on the formalism and later we discuss the mechanism used in DVTAG to compute lexical dependencies.

2.1. DVTAG DEFINITION

DVTAG is a triple, consisting of a set of *initial left-contexts* (the initial states), a set of *axiom schemata* (the mechanisms used to update the states based on some operations) and a set of *deduction rules* (defining how the process evolves). Now we formally define the eight attachment operations of

* In all the Figures we represent the integer in the dotted trees using a dot; in Fig. 1 the fringes are depicted as ovals. Cf. Mazzei (2005), Mazzei et al (2005) for formal definitions.

DVTAG, and give an informal description of the derivation process[**]. In this definition we neglect the notion of head-feature, discussed later in detail.

In DVTAG there are two substitution operations (similar to LTAG substitution), four adjoining operations (similar to LTAG adjoining), one shift operation, and one wrapping operation[*].

The **Shift** operation $Shi(\langle\gamma, i\rangle)$ is defined on a single dotted tree $\langle\gamma, i\rangle$ and returns the dotted tree $\langle\gamma, i+1\rangle$. It can be applied only if a terminal symbol belongs to the fringe of $\langle\gamma, i\rangle$ (Fig. 1-a).

The **Substitution** operation $Sub^{\rightarrow}(\langle\alpha, 0\rangle, \langle\gamma, i\rangle)$ is defined on two dotted trees, $\langle\gamma, i\rangle$ and $\langle\alpha, 0\rangle$. If there is a substitution node $\mathcal{N}$ equal to the root node of α in the fringe of $\langle\gamma, i\rangle$, the operation returns a new dotted tree $\langle\delta, i+1\rangle$ such that δ is obtained by grafting α into $\mathcal{N}$ (Fig. 1-b).

The **Inverse Substitution** operation $Sub^{\leftarrow}(\langle\zeta, 0\rangle, \langle\gamma, i\rangle)$ is defined on two dotted trees $\langle\gamma, i\rangle$ and $\langle\zeta, 0\rangle$. If the root of γ belongs to the fringe of $\langle\gamma, i\rangle$, and there is a substitution node $\mathcal{N}$ belonging to the fringe of $\langle\zeta, 0\rangle$ such that $\mathcal{N}$ is equal to the root of γ, the operation returns a new dotted tree $\langle\delta, i+1\rangle$ such that δ is obtained by grafting γ into $\mathcal{N}$ (Fig. 1-c).

The **Adjoining from the left** operation $\nabla_L^{\rightarrow}(\langle\beta, 0\rangle, \langle\gamma, i\rangle, add)$ is defined on two dotted trees $\langle\gamma, i\rangle$ and $\langle\beta, 0\rangle$, such that β is a *left auxiliary tree* (Schabes and Waters, 1995). If there is a non-terminal node $\mathcal{N}$ in the fringe of $\langle\gamma, i\rangle$ such that $\mathcal{N}$ is equal to the root of β, the operation returns a new dotted tree $\langle\delta, i+1\rangle$ such that δ is obtained by grafting β into $\mathcal{N}$ (Fig. 1-d).

The **Adjoining from the right** operation $\nabla_R^{\rightarrow}(\langle\beta, 0\rangle, \langle\gamma, i\rangle, add)$ is defined on two dotted trees $\langle\gamma, i\rangle$ and $\langle\beta, 0\rangle$, such that β is a *right auxiliary tree* (Schabes and Waters, 1995). If there is a non-terminal node $\mathcal{N}$ in the fringe of $\langle\gamma, i\rangle$ such that $\mathcal{N}$ is equal to the root of β, the operation returns a new dotted tree $\langle\delta, i+1\rangle$ such that δ is obtained by grafting β into $\mathcal{N}$ (Fig. 1-e).

The **Inverse adjoining from the left** operation $\nabla_L^{\leftarrow}(\langle\zeta, 0\rangle, \langle\gamma, i\rangle, add)$ is defined on two dotted trees $\langle\gamma, i\rangle$ and $\langle\zeta, 0\rangle$. If the foot of γ belongs to the fringe of $\langle\gamma, i\rangle$, and there is a node $\mathcal{N}$ belonging to fringe of $\langle\zeta, 0\rangle$ such that $\mathcal{N}$ is equal to the foot of γ, the operation returns a new dotted tree $\langle\delta, i+1\rangle$ such that δ is obtained by grafting γ into $\mathcal{N}$ (Fig. 1-f).

The **Inverse adjoining from the right** operation $\nabla_R^{\leftarrow}(\langle\zeta, 0\rangle, \langle\gamma, i\rangle, add)$ is defined on two dotted trees $\langle\gamma, i\rangle$ and $\langle\zeta, 0\rangle$. Moreover $\langle\zeta, 0\rangle$ has a null lexical item (ε node) as leftmost leaf. If the root of γ belongs to the fringe of $\langle\gamma, i\rangle$, and there is a node $\mathcal{N}$ belonging to fringe of $\langle\zeta, 0\rangle$ such that $\mathcal{N}$ is equal to the root of γ, the operation returns a new dotted tree $\langle\delta, i+1\rangle$ such that δ is obtained by grafting γ into $\mathcal{N}$ (Fig. 1-g).

[**] See Mazzei et al. (2005) for detailed formal definitions.

[*] To avoid unnecessarily verbose descriptions, we speak of node $\mathcal{N}$ both for cases of $\mathcal{N}$ itself and for its label, when the distinction is clear from the context. In these definitions the symbol *add* represents a Gorn address (a Gorn address is a way of referring to a particular node in a tree uniquely: the Gorn address of a root node is ε; if a node has Gorn address η, then its i-th child has Gorn address $\eta \cdot i$).

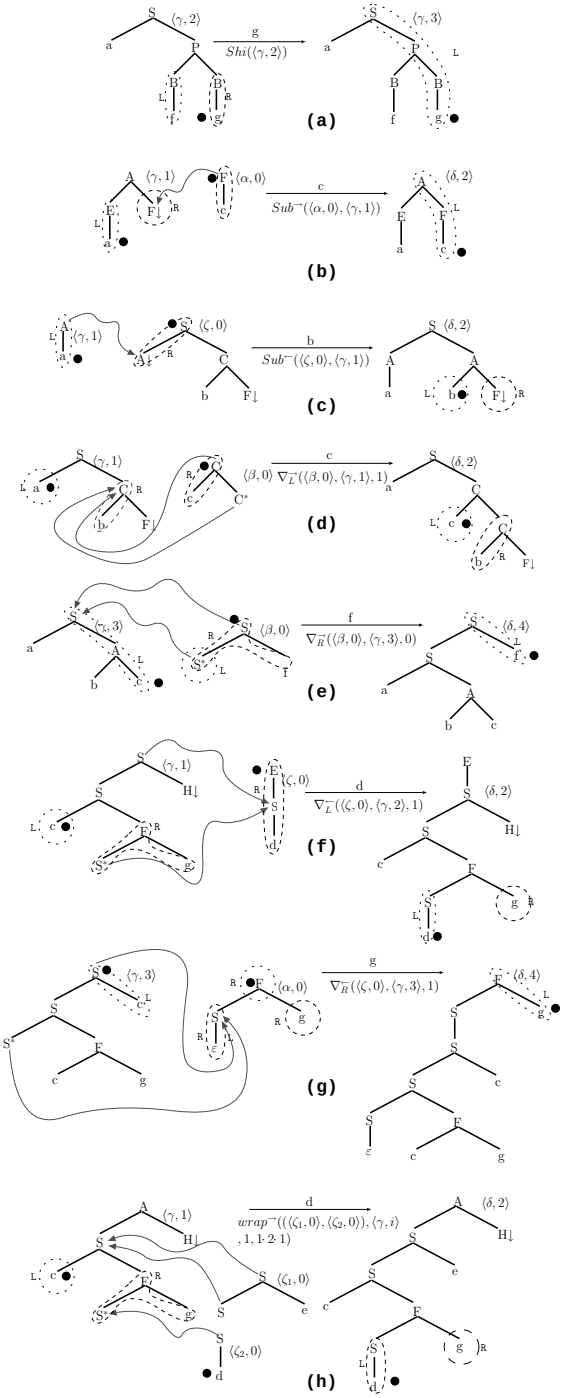

Figure 1. Operations in DVTAG.

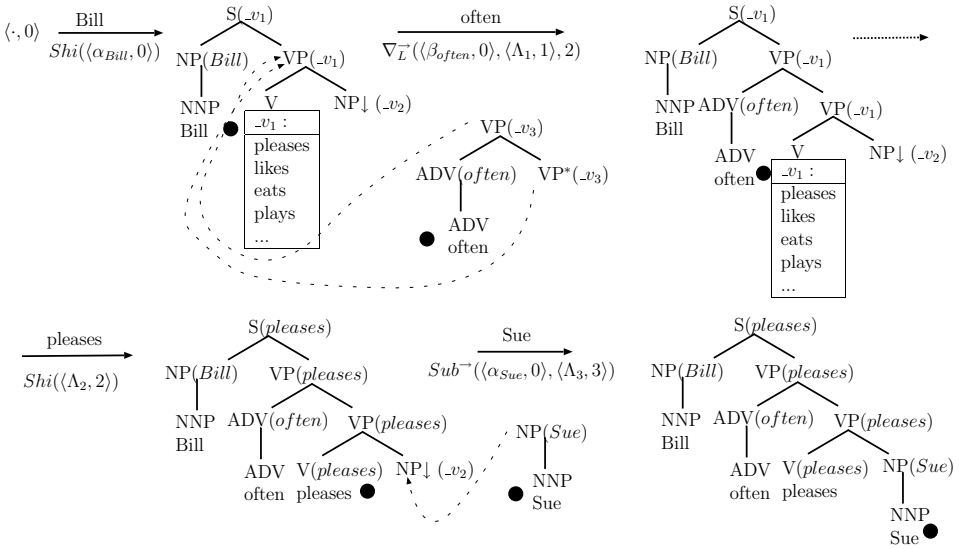

Figure 2. The DVTAG derivation of the sentence *Bill often pleases Sue*.

The **dynamic wrapping** operation** $wrap^{\rightarrow}((\langle\zeta_1, 0\rangle, \langle\zeta_2, 0\rangle), \langle\gamma, i\rangle, add_1, add_2)$ takes as input a dotted tree, $\langle\gamma, i\rangle$, and a pair of dotted trees $(\langle\zeta_1, 0\rangle, \langle\zeta_2, 0\rangle)$. If there is a non-terminal node $\mathcal{N}_2$ that belongs to the fringe of $\langle\gamma, i\rangle$ such that $\mathcal{N}_2$ is equal to the root of ζ_2, and there is a node $\mathcal{N}_1$ such that $\mathcal{N}_1$ is equal to of the root of ζ_1, the operation returns a new dotted tree $\langle\delta, i+1\rangle$ such that δ is obtained by simultaneously grafting ζ_1 into $\mathcal{N}_1$ and by grafting ζ_2 into $\mathcal{N}_2$ (Fig. 1-h).

DVTAG derives a *final** derived tree by starting from an initial left-context and by applying iteratively the eight operations described above. In Fig. 2 we have an initial *shift* of the word *Bill*, an adjoining from the left of the left auxiliary tree anchored by *often*, another *shift* grounding the variable $\_v_1$ to the constant *pleases*, and a *substitution* of the dotted elementary tree anchored by *Sue* to produce a final left-context. The elementary tree *Bill* is linguistically motivated up to the NP projection; the rest of the structure is required by connectivity. These extra nodes are called *predicted nodes*. A predicted preterminal node is annotated with (and refers to) a set of lexical items, that explicitly denotes the finite range of the *predicted head*. So, the

extended domain of locality available in LTAG has to be further extended (see Lombardo et al. (2004)).

Now we complete the definition of the DVTAG operations by describing the mechanism for updating the head feature. DVTAG uses the same feature structure as LTAG, i.e. the features of a non-terminal node are split in two sets, the *top* feature set and the *bottom* feature set. Moreover DVTAG uses the same unification mechanism as LTAG, the only difference being the special treatment of the head-feature. The value of the head-feature is made explicit only in the bottom feature set, i.e. we leave the head-feature in the top feature set uninstantiated. While this choice does not affect the substitution and shift operations, it plays a key role in the adjoining operations: the top feature set of the adjoining node unifies with the top feature set of the root of the auxiliary tree, and the bottom feature set of the adjoining node unifies with the bottom feature set of the foot of the auxiliary tree (Vijay-Shanker, 1987). This mechanism contributes to the computation of the lexical dependencies (see Sec. 3.2).

In Fig. 3-a, in the adjoining of $\langle \beta_{often}, 0 \rangle$ in $\langle \Lambda_2, 2 \rangle$, the head-variable $\_v_3$ unifies with the head-variable $\_v_1$. In Fig. 3-b, in the inverse adjoining of $\langle \Lambda_2, 2 \rangle$ in $\langle \alpha_{Mary}, 0 \rangle$, the head-variable $\_v_3$ unifies with the head-variable $\_v_1$. In the remainder of the paper we continue to represent the head-feature without an explicit representation of the feature structure.

2.2. THE SYNTAX-SEMANTICS INTERFACE OF DVTAG

The dependency tree format of the derivation tree has been used in TAG community as an interface to semantics[⋆]. There is a straightforward relation between lexicalized constituency grammars and dependency grammars. In a lexicalized grammar we associate (at least) one word to each elementary structure. Then we can view an operation on two elementary structures A and B, anchored respectively by the words w_a and w_b, as a dependency relation between the words w_a and w_b. Derivation trees in LTAG record the history of a derivation by attaching tree identifiers of substituted and adjoined elementary trees as children of the tree identifier of the host elementary tree (Vijay-Shanker, 1987). If we replace the tree identifiers with the words anchoring the trees in the derivation tree, we obtain a dependency tree.

[⋆⋆] The necessity of this operation will be clear in sec. 3.1. See Mazzei et al. (2005).

[⋆] Informally speaking, a *final* dotted tree does not have substitution or foot nodes on the right of the dot. See Mazzei (2005), Mazzei et al. (2005) for a formal definition of final dotted tree and language generated in DVTAG.

[⋆] There are several approaches for interfacing the LTAG derivation with semantics. We refer only to the *standard* approach (Rambow and Joshi, 1997). Other approaches have also been described, for instance in Frank and van Genabith (2001) and Dras et al. (2004).

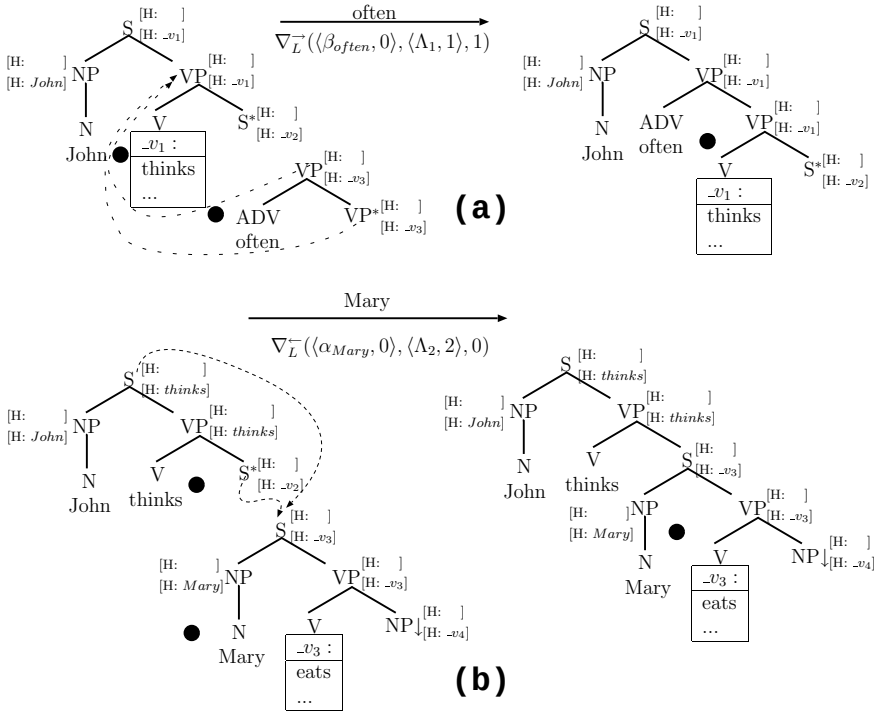

Figure 3. Update of the head feature in the adjoining operations for an *athematic* auxiliary tree (a), and for a *complement* auxiliary tree (b).

However, there are some known problems involved with the use of derivation trees for semantic interpretation, which are related to the fact that adjoining is used for the construction of the recursive structures, whether or not modification is involved. Thus, for example, *bridge* verbs such as *think* are introduced using auxiliary trees, because they introduce recursive structure. This means that the main clause verb will be analyzed counter-intuitively as being a modifier of the complement clause verb (see Rambow et al. (2001) for a discussion). Adjoining is also used in the case of *raising* verbs like *seem*, with similar results. When bridge and raising verbs are used together, as in "John thinks Mary seems to sleep" (Fig. 4), the derivation tree connects the bridge and the predicative verb, without any connection with the raising verb, neglecting the dependency between the bridge and the raising verbs.

In DVTAG the attachment operations combine an individual elementary tree with a left-context, rather than combining individual pairs of elementary

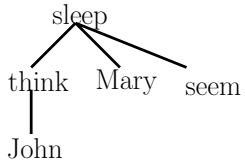

Figure 4. A LTAG derivation tree for the sentence *John thinks Mary seems to sleep.*

trees. Thus it is not possible to use the notion of derivation tree in the usual sense. Our solution is to adopt a *head-based* notion for deriving a dependency tree corresponding to the left-context. Now we describe the DVTAG mechanism that computes the lexical dependencies: in the next sections we show that in many cases, such as bridge and raising verbs, this solution leads to a more semantically adequate derivation tree than in standard LTAG.

Since the left-context is a constituency tree, we can view the extraction of a dependency tree from a DVTAG left-context as the translation of a constituency tree into a dependency tree. Lin (1995) proposed a very simple algorithm to transform a Penn Treebank tree into a dependency tree. This algorithm builds a dependency tree while visiting the nodes of the constituency tree top-down. For each non-terminal node of the constituency tree, the algorithm expands the dependency tree with new relations between the lexical head projecting the non-terminal node and the lexical heads projecting its children. Since the Penn treebank does not contain annotation about the lexical heads, Lin used a head-table (Magerman, 1995) to recover this information. Lin's algorithm is particularly appealing for DVTAG since the head feature provides the necessary information about lexical projections. Following Lin's algorithm, we transform a DVTAG left-context into a dependency tree by using a percolation procedure. In addition to Lin‘s approach, our dependency tree, similarly to LTAG derivation tree must allow for an appropriate treatment of non final derived trees, does not contain function words, must be able to describe the dependencies for non projective sentences (Rambow and Joshi, 1997). We can account for these facts by augmenting the value range of the head-feature with the special symbol "−". In fact, by treating the head variables as lexical items we can build a dependency tree that contains the relations between the fully specified lexical items and the underspecified lexical items too. Moreover the special value "−" can be used to distinguish head-feature values of the nodes projected by function words and those projected by words that are fillers of some traces (i.e. words that are moved from their semantically appropriate

positions in non projective sentences). For each node $\mathcal{N}$ of the left-context, the algorithm executes these steps:

- Construct the dependency tree on the head child, i.e. on the child that shares the head-value with $\mathcal{N}$.
- If the head-value of $\mathcal{N}$ is different from "$-$", then for each non-head-child $\mathcal{C}$
  - If the head-value of $\mathcal{C}$ is different from "$-$", add a dependency relation between the head-value of $\mathcal{N}$ and the head-value of the child. If the head-variable of $\mathcal{C}$ is equal to "$-$", add a dependency relation between the head-value of $\mathcal{N}$ and the head-value of the descendent of $\mathcal{C}$, which is not the filler of a trace, if there is one.
  - Construct the dependency tree for $\mathcal{C}$.

Note that the node dominating the trace will project a head-value equal to the extracted word: in this way, in the dependency tree, the co-indexed word will be in a dependency relation with the head dominating the trace*. Fig. 5 depicts the DVTAG derivation of the sentence *Beans I often eat*: next to each left-context can be found the dependency tree obtained applying the conversion algorithm.

3. Linguistic adequacy of DVTAG: what we do (not) miss in DVTAG by introducing dynamics

In this section we consider the linguistic adequacy of DVTAG. The linguistic constructions discussed in section 3.1 have widely accepted LTAG analyses (genitive construction in English and embedded clause construction in Dutch). These constructions are interesting from our point of view because of the restrictions that dynamics poses on DVTAG.

In contrast, the linguistic constructions discussed in the section 3.2 (athematic and complement adjoining, raising and bridge verbs) do not have widely accepted LTAG analyses. In these cases DVTAG can shed a new light on the representation of the lexical dependencies.

3.1. WIDELY ACCEPTED LTAG ANALYSES

The restrictions posed by the dynamic approach are a consequence of the proof about the generative power of DVTAG (Mazzei et al., 2005). In that proof we have defined a subclass of LTAG, called *dynamic LTAG* that is strongly equivalent to DVTAG, i.e. for each dynamic LTAG there is a DVTAG generating the same tree language and vice versa**. There are two

* We refer to a generic dependency formalism that does not use traces (e.g. (Mel'cuk, 1987)): we can straightforwardly modify the algorithm to generate dependency trees with explicit annotation of the traces.

** There is a trivial relation between the LTAG lexicon and the equivalent DVTAG lexicon

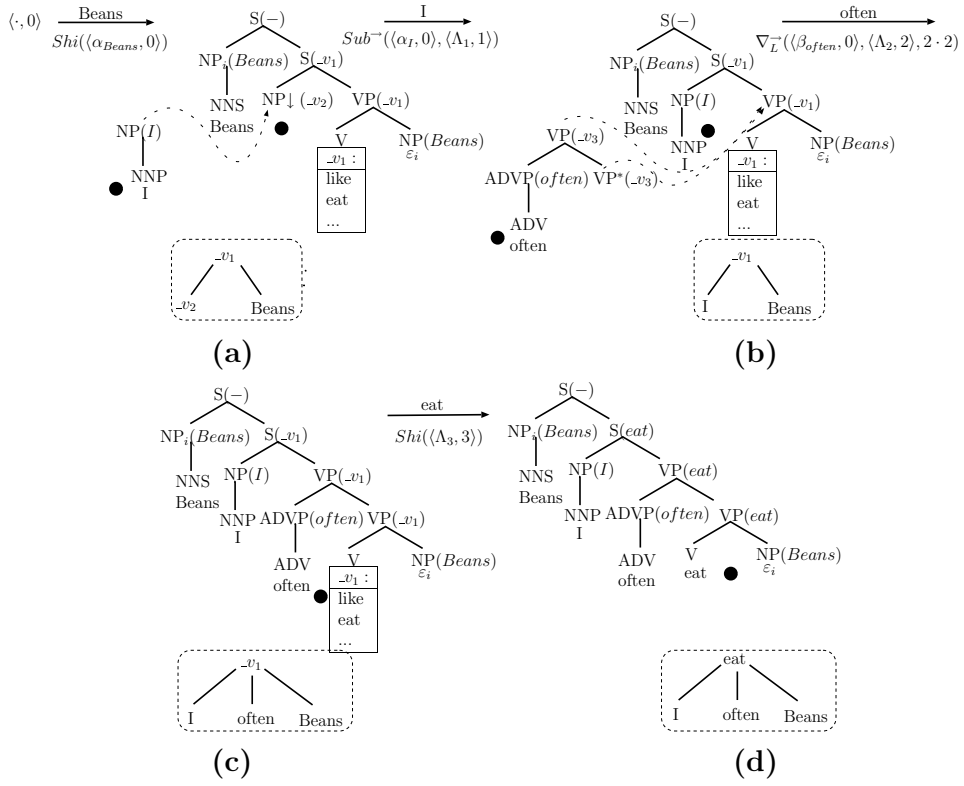

Figure 5. The DVTAG derivation and the corresponding sequence of dependency trees for the sentence *Beans I often eat*.

constraints on the LTAG derivation tree (*generative power constraints*):

1. Each node $\mathcal{N}$ in the derivation tree has at most one child such that the anchor of the child precedes the anchor of the node $\mathcal{N}$ in the linear order of the sentence (cf. Fig. 6-a)
2. If a node $\mathcal{N}$ in the derivation tree has a child such that the anchor of the child precedes the anchor of the node $\mathcal{N}$ in the linear order of the sentence, then the anchor of the parent of the node $\mathcal{N}$ does not precede the anchor of the node $\mathcal{N}$ in the linear order of the sentence (cf. Fig. 6-b).

At first sight, these two constraints appear to limit the linguistic constructions that are derivable in DVTAG. For instance: the first constraint appears to bar constructions in which two arguments precede the head in the linear order of the sentence, such as declarative transitive sentence constructions in head final languages (e.g., Japanese, see below); the second constraint

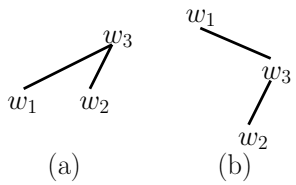

Figure 6. Generative power constraints (the names of the elementary tree are the names of their anchors). The first constraint forbids the case (a), the second constraint forbids the case (b).

appears to disallow the analysis of left modifiers and in general of *left recursion* constructions in the derivation tree (e.g., adjective and genitive constructions). However, most of these constructions are indeed derivable in DVTAG, although they require alternative adequate analyses in comparison with standard LTAG analyses. First we consider the DVTAG derivations of head-final languages, and later we consider the DVTAG derivations for a number of left recursive constructions in English (genitive construction) and Dutch (embedded clauses construction).

Head-final constructions

In pure head final languages the head is always preceded by its arguments. For instance in Japanese the verb is always the last word of the sentence. In the example (1) we report one of the sentences used in the Kamide et al. (2003)'s experiment. This is an active transitive sentence in which the verb *karakau* ("teases") follows the subject *weitoresu* ("waitress") and the direct object *kyaku* ("customer") in the linear order of the sentence. Moreover, the verb is pre-modified by the adverb *tanosigeni* ("merrily") (sentence (1)).

(1) *weitoresu-ga* *kyaku-o* *tanosigeni* *karakau*
waitress-NOM customer-ACC merrily tease
The waitress will merrily tease the customer

An LTAG analysis of this sentence would be similar to the standard LTAG analysis of many English transitive sentences. The main difference being that there would be a different elementary tree for the verb, in which both of the NP substitution nodes precede the verb. As a consequence of generative power constraints, is not possible to replicate the same analysis in DVTAG. In order to produce the correct dependency structure for this sentence, we need to define the DVTAG lexicon in which the elementary trees are designed to fulfill the strong connectivity hypothesis using predicted nodes. Fig. 7 shows a DVTAG lexicon for the derivation of the sentence (1). This lexicon

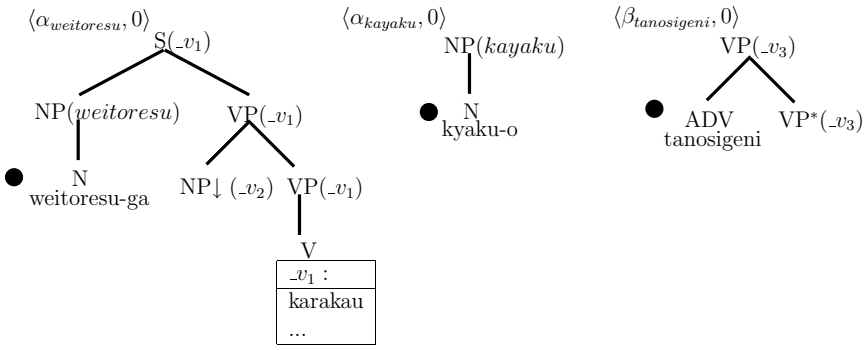

Figure 7. DVTAG deriving the sentence *weitoresu-ga kyaku-o tanosigeni karakau* (waitress-NOM customer-ACC merrily tease: *The waitress will merrily tease the customer*)

consists of three elementary dotted trees: the initial trees $\langle \alpha_{weitoresu}, 0 \rangle$ and $\langle \alpha_{kyaku}, 0 \rangle$, and the left auxiliary tree $\langle \beta_{tanosigeni}, 0 \rangle$. Some nodes in the initial tree $\langle \alpha_{weitoresu}, 0 \rangle$ are predicted: they are projected by the underspecified lexical item in the verb position. Using these predicted nodes we are able to project the argument structure of the underspecified verb at the first step of the derivation, when the word *weitoresu* is derived. In this way we can predict the relation that binds the word *weitoresu* to the word *kyaku* before the head of the sentence is derived[⋆]. In this example, with the aim to fulfill strong connectivity we have predicted the projected nodes of the verb in the elementary dotted tree $\langle \alpha_{weitoresu}, 0 \rangle$. There are some similarities between this elementary dotted tree and the category $S/(S\backslash NP)$ of CCG, produced by the operation of *type-raising* (Steedman, 2000). See Lombardo et al. (2004) for further discussion of this issue.

Consider that this solution is also employed in the canonical English construction subject-verb (cf. the elementary tree anchored by *Bill* in Fig 2). Note that the degree of prediction is fairly extensive for the DVTAG derivation in head-final sentences. In the case of (1), for example, the transitive argument structure of the verb is predicted at the first word of the sentence. In fact, recent experiments, including that of Kamide et al. (2003) do show evidence for a considerable degree of prediction of argument structure in processing head-final constructions, although the increase of prediction also implies an increase in local ambiguity. Further research is required to es-

[⋆] This fact is also consistent with the results of Kamide et al.'s experiments (Kamide et al., 2003), i.e. the empirical evidence that appropriate thematic roles are assigned to the subject and the direct object as soon as the two words are perceived.

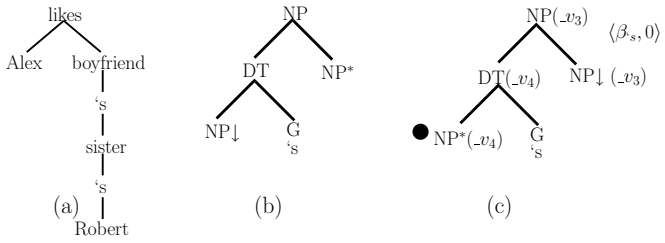

Figure 8. The dependency tree of *Alex likes Robert's sister's boyfriend* (a), the XTAG elementary tree (b) and DVTAG elementary dotted tree (c) for genitive construction.

tablish exactly how predictive a processing model needs to be in order to model human sentence processing appropriately, or whether it is desirable to introduce a degree of underspecification of the predicted structures.

Left embedding constructions: genitive and cross-serial constructions

A case of left embedding is the genitive construction in English, as in the sentences

(2) Alex likes Robert.

(3) Alex likes Robert's sister.

(4) Alex likes Robert's sister's boyfriend.

There is no limit to the number of words (or constituent boundaries) between the word *Robert* and the head of the noun phrase in the object position. Fig. 8-a, depicts the dependency tree of the sentence (4), which is equal to the LTAG analysis of genitive construction given in the XTAG grammar (Doran et al., 2000). The left recursion is obtained by means of the adjoining of a left auxiliary tree anchored by the suffix *'s* Fig. 8-b. But the second generative power constraint does not hold for this derivation tree. Thus we cannot reproduce the dependency tree (and the LTAG derivation tree) depicted in Fig. 8-a through adjoining of left auxiliary dotted trees. Moreover the number of predicted nodes necessary to guarantee connectivity is not limited a priori, since the embedding of the first noun is unbounded. However in this case we can derive the same LTAG derived tree by adjoining of right auxiliary trees, rather than left auxiliary trees. In Sturt (1997); Lombardo and Sturt (1997), a computational model was proposed in which the adjoining of right auxiliary trees was used in a monotonic architecture

to model syntactic reanalysis. Fig. 8-c shows an elementary dotted tree such that genitive constructions can be generated by using only right auxiliary trees. In LTAG, the genitive construction is obtained by adjoining left auxiliary trees, because in LTAG the lexical dependencies are expressed in the derivation tree. Note that by using adjoining of right auxiliary trees in LTAG, in the derivation of the sentence (4) the word *Robert* would be a child of the word *likes*, since the elementary tree anchored by *Robert* is substituted into the elementary tree anchored by *likes*. But in the sentence (4), the word *Robert* is a modifier of the word *sister*. Thus, if we use a right auxiliary tree for the genitive construction in LTAG the derivation tree would represent the wrong dependencies. In contrast, since in DVTAG we use a head-feature based mechanism to compute lexical dependencies, we can derive the correct dependency tree by using right adjoining.

A more challenging left embedded structure is represented by cross serial dependencies in Dutch embedded clauses. Consider the sentence fragment

(5) *... dat Jan Piet Marie zag laten zwemmen*
... that Jan Piet Marie saw help swim
... that Jan saw Piet help Marie to swim

Here the dots indicate that this fragment is the embedded clause of a larger matrix clause, e.g. *Ik denk dat Jan Piet Marie zag laten zwemmen* (*I think that Jan saw Piet help Marie to swim*). The correct verb order is obtained through extraposition of the verb (Joshi, 1990; Kroch and Santorini, 1991) (Fig. 9-a). We need to allow the insertion of lexical material on both sides of predicted material. For this purpose, we use the wrapping operation. In fact DVTAG can generate the correct derived tree and the correct dependency tree by using a different perspective on adjoining. Adjoining is as an operation that takes as input an auxiliary tree β and an elementary tree γ, and returns as output a new tree obtained from a transformation of γ by using the tree β. Recently Joshi (2004) has pointed out that we can see adjoining from another perspective. The basic idea of this view, called *flexible composition*, is to change the *direction* of adjoining, i.e. it is the tree β that is transformed by using the tree γ (Joshi, 2004). In 9-(b) the tree β is adjoined into γ by using the adjoining operation. In contrast, 9-(c) illustrates the *wrapping* perspective. In this view the tree γ is considered to be formed from a supertree γ_1 and a subtree γ_2. The derived tree is obtained by wrapping the trees γ_1 and γ_2 around β, i.e. by attaching the supertree γ_1 onto the root of β and attaching the subtree γ_2 onto the foot of β. The wrapping operation increases the derivational generative power of LTAG as well as the strong generative power of DVTAG (although remaining lower than LTAG) (Mazzei et al., 2005).

In the derivation depicted in Fig. 10 we use the wrapping operation to derive in DVTAG the LTAG derived tree for Dutch embedded clauses. The

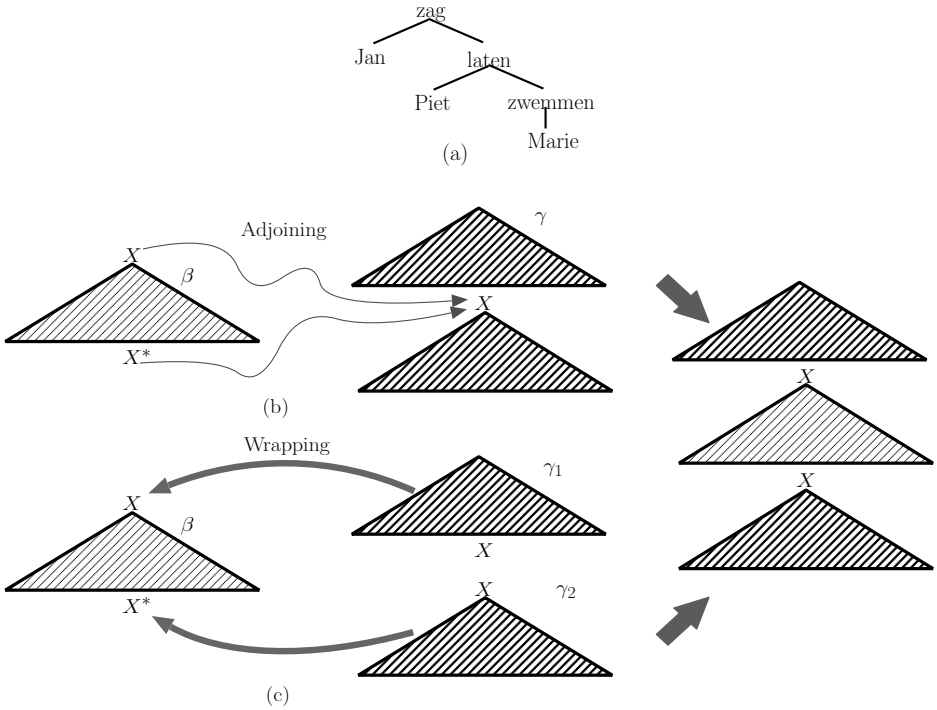

Figure 9. Adjoining operation (a) and wrapping operation (b): if γ is an elementary tree wrapping does not change the strong generative power of the grammar.

first operation (Fig. 10-a) is an inverse adjoining from the left of the left-context $\langle \Lambda_1, 1 \rangle$ in the elementary dotted tree $\langle \beta_{Jan}, 0 \rangle$. The second operation (Fig. 10-b) is a wrapping operation. The dotted tree $\langle \beta_{Piet}, 0 \rangle$ is split into the supertree $\langle \beta^1_{Piet}, 0 \rangle$ and the subtree $\langle \beta^2_{Piet}, 0 \rangle$: $\langle \beta^1_{Piet}, 0 \rangle$ is adjoined to the node that was the root of $\langle \beta_{Jan}, 0 \rangle$ and $\langle \beta^2_{Piet}, 0 \rangle$ is substituted in the node that was the foot of $\langle \beta_{Jan}, 0 \rangle$. The third operation (Fig. 10-c) is also a wrapping operation. The dotted tree $\langle \alpha_{Marie}, 0 \rangle$ is split into the supertree $\langle \alpha^1_{Marie}, 0 \rangle$ and the subtree $\langle \alpha^2_{Marie}, 0 \rangle$. The final dotted tree $\langle \Lambda_4, 4 \rangle$ is obtained by adjoining $\langle \alpha^1_{Marie}, 0 \rangle$ into the node that was the root of $\langle \beta_{Piet}, 0 \rangle$, and by substituting $\langle \alpha^2_{Marie}, 0 \rangle$ into the node that was the foot of $\langle \beta_{Piet}, 0 \rangle$.

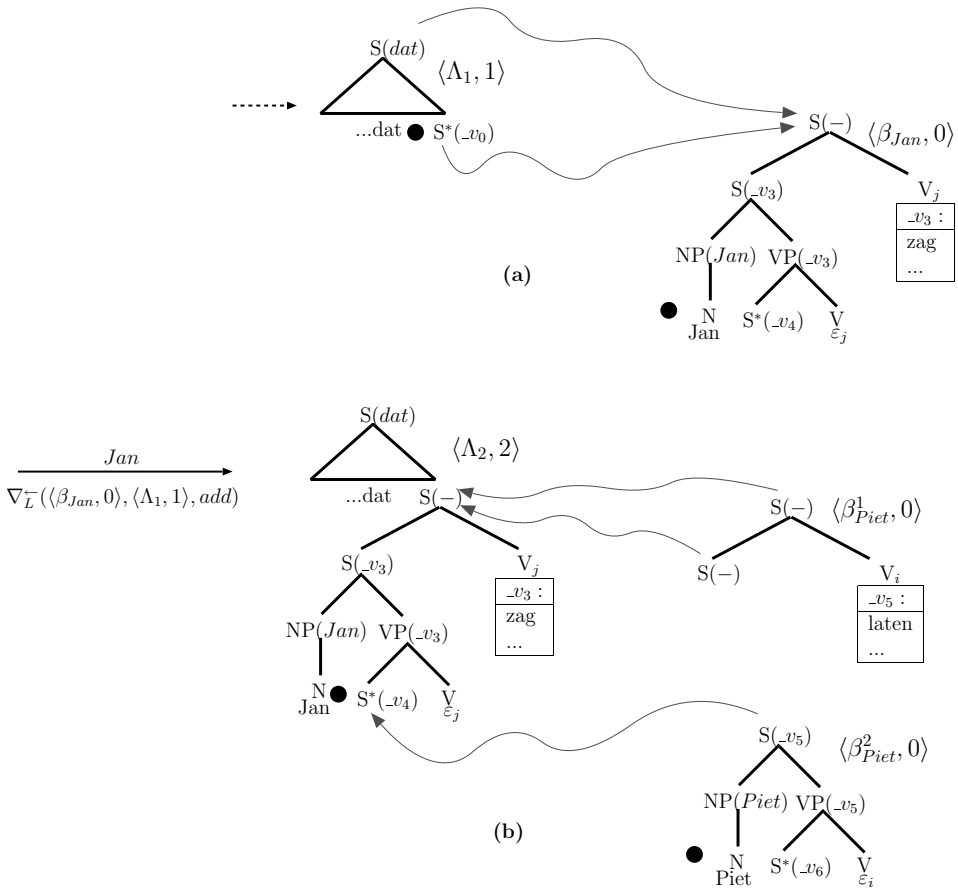

Figure 10. Scanning *Jan* and *Piet* DVTAG derivation of the Dutch embedded clauses (5).

3.2. CONTROVERSIAL LTAG ANALYSES

The case of bridge and raising verbs has been debated in TAG community. This issue becomes more complex when bridge and raising verbs are used with the presence object extraction, and subject-to-subject raising⋆.

⋆ Raising verbs are a subclass of verbs that take infinitive complements: they lack a specifier position and fail to assign case, i.e. there is no thematic relation between the raising verb and its subject. For instance the verb *seems* is a raising verb. In the sentence *he seems to understand her*, the verb *seems* has no thematic relation with its subject. The name *raising* derives from the transformational analysis of these constructions, in which the subject is raised from its original position in the embedded clause to the matrix

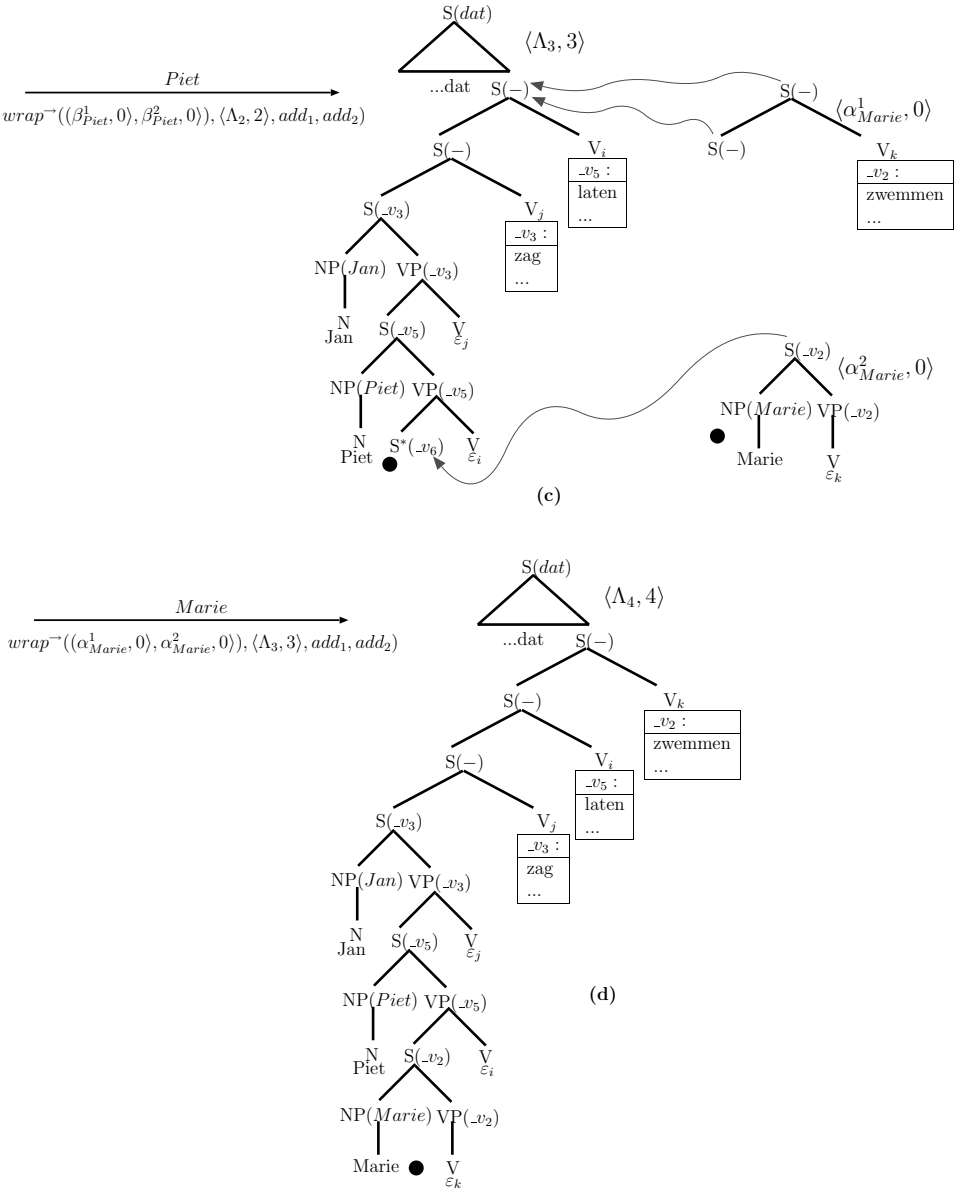

Figure 11. (*continued*) Scanning *Marie* and the three verbs in DVTAG derivation of the Dutch embedded clauses (5).

The sentence

(6) Hotdogs he claims Mary seems to adore.

has some lexical dependencies that cannot be described in a simple way by using LTAG derivation trees. As pointed out by Rambow et al. (2001), the elementary auxiliary trees anchored by both *claim* and *seem* are adjoined into the initial tree anchored by *adore*, while in dependency approaches *seem* typically depends on *claim*, and *adore* on *seem*, Fig. 12-a. The key point is that LTAG uses the adjoining operation to insert both modifiers and clausal complements, and there is no way in standard LTAG to tell apart these two relations in the derivation tree. Kroch (1989) pointed out that auxiliary trees in LTAG can be linguistically classified as *athematic* or *complement* trees:

Athematic trees introduce a modifying, complement or dislocated phrase. In this case the foot and root nodes belong to the same projection chain (e.g. Fig. 12-b).

Complement trees introduce a predicate, and the foot node is an argument of the predicate. Kroch specifies that in this case the foot node of the complement tree is "lexically governed" (e.g. Fig. 12-c).

Different types of auxiliary trees correspond to two different linguistic interpretations of the adjoining operation: in the first case the adjoining is used for modification and in the second case for complementation. The double function of adjoining does not allow for a uniform account of the lexical dependencies in the derivation tree. In (Rambow and Joshi, 1997) it was proposed to distinguish athematic and complement adjunction by noting that complement adjunction is usually performed at the root node, but this rule is not always true (e.g. in raising construction a complementary auxiliary tree is adjoined on a "VP" node Fig. 12-c (Frank, 2002)).

In DVTAG we do not use the derivation structure to build the dependency tree, and the head-feature produces a simple characterization of the distinction between athematic and complement auxiliary trees. In fact, following Kroch's characterization, in an athematic auxiliary tree, the head-value of the foot node will be the same as that of the root node (cf. Fig. 3-a and Fig. 3-b). Consider for instance the tree anchored by *often* in the derivation depicted in Fig. 5-b. The foot and the root have the same value $\_v_3$ in the head-variable. As a consequence of adjoining from the left, the head-variable $\_v_3$ will be unified with the head-variable $\_v_1$. Therefore *often* is correctly described as dependent of $\_v_1$ in the dependency tree built with the conversion algorithm. In contrast, in a complement auxiliary tree the

subject position: *It seems that he understands her* $\Rightarrow$ *he*$_i$ *seems* ε_i *to understand her* (Frank, 2002).

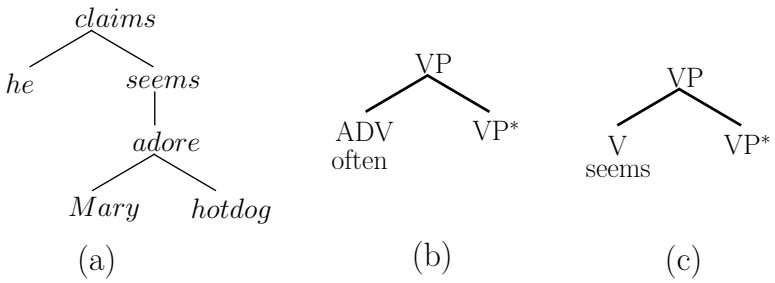

Figure 12. (a) The widely accepted dependency tree for the sentence *Hotdogs he claims Mary seems to adore*. (b) Athematic tree (c) Complement tree.

head variable of the foot node will be different from the head-variable of the root node. In Fig. 3 the elementary dotted tree $\langle \beta_{John}, 0 \rangle$ is a complement auxiliary dotted tree. In this case, the foot node has the head-variable set to the value $\_v_2$ and the head-variable of the root node has the value $\_v_1$. In this way, in the corresponding dependency structure, $\_v_2$ is dependent on $\_v_1$. After inverse adjoining from the left of the left-context into the dotted tree $\langle \alpha_{Mary}, 0 \rangle$, $\_v_2$ is unified with the head-variable $\_v_3$, and in the dependency structure corresponding to the new left-context the lexical item *think* dominates $\_v_3$.

Thus, in DVTAG the head-feature straightforwardly allows us to distinguish athematic from complement auxiliary trees, so that by using the conversion algorithm we are able to derive the correct dependency structure for the sentence (6), as reported in the Fig. 13 and Fig. 14. In the first step of the derivation, an initial elementary tree undergoes a *Shift* operation. This tree has the object *hotdogs* extracted to the first position of the yield, so this word is the leftmost lexical item of the tree. To satisfy the filler-trace relationship implemented by the algorithm, the head-variable of the root node, that connects the extracted *NP* to the trace has the special value "−". In the corresponding dependency tree the word *hotdogs* depends on the underspecified head $\_v_2$: in fact the co-indexed trace is dominated by the node "NP", and the value of the head-variable for this node is *hotdog*.
In the second step, a complement auxiliary tree is adjoined from the left. This tree is anchored by the lexical item *he*, and contains some nodes that are projected by the underspecified lexical item $\_v_3$. In the corresponding dependency tree, the underspecified lexical item $\_v_2$, that underspecifies the verb of the embedded clause, depends on $\_v_3$, which underspecifies the verb

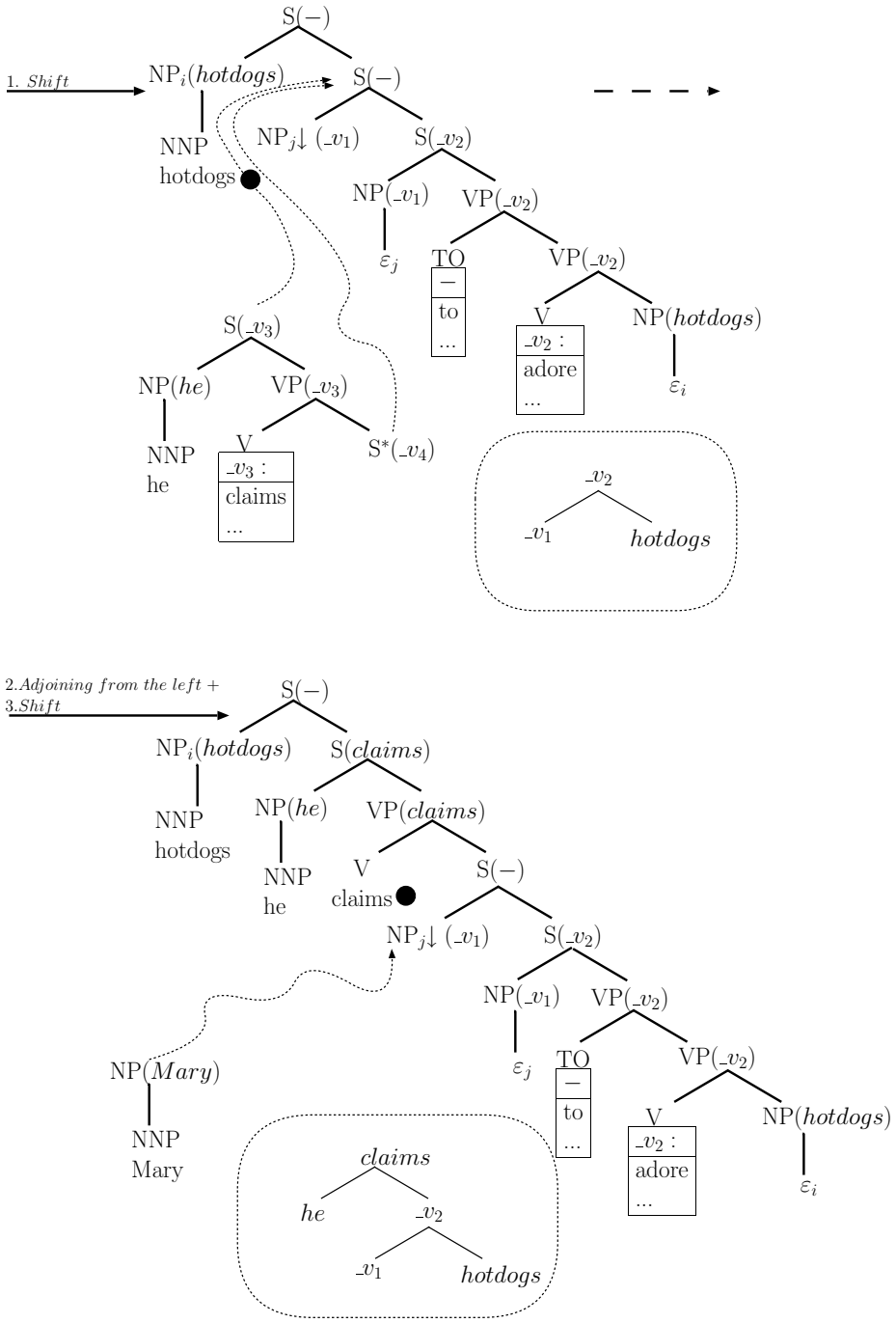

Figure 13. The derivation of the sentence *Hotdogs he claims Mary seems to adore*: operation 1, 2 and 3.

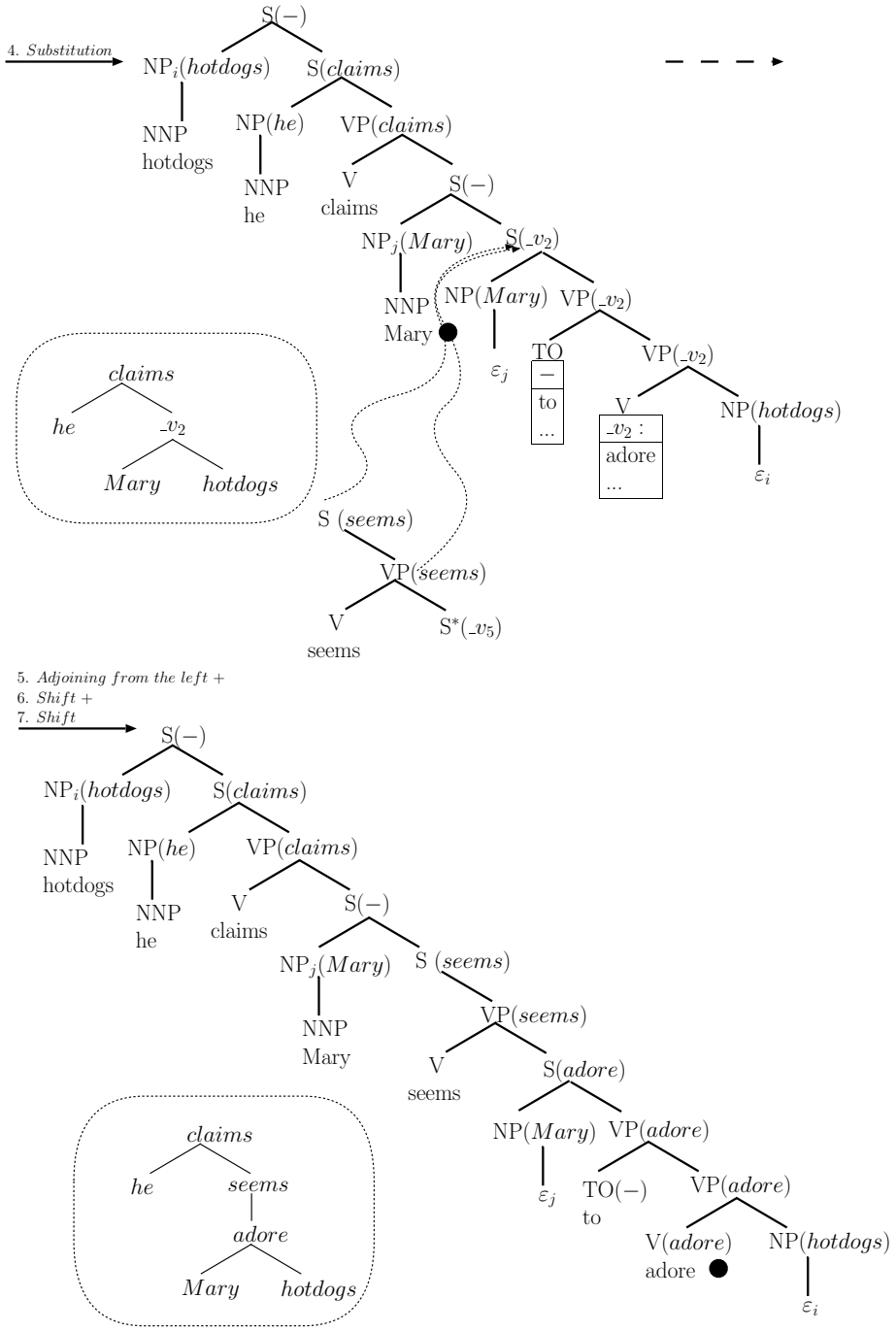

Figure 14. (*continued*) The derivation of the sentence *Hotdogs he claims Mary seems to adore*: operation 4, 5, 6 and 7.

of the matrix clause*.
In the third step, a *Shift* operation introduces the lexical item *claims* at the highest verb node.
In the fourth step the subject of the main sentence is inserted by the lexical item *Mary* with a *Substitution* operation.
In the fifth step a complement auxiliary tree anchored by the raising verb *seems* is adjoined from the left. In contrast to standard LTAG analysis proposed in Kroch and Santorini (1985), we propose that *seems* projects an S node too. As a consequence we suppose an extracted position for the subject of the embedded sentence, and a co-indexed trace is inserted in the original subject position. This approach follows the traditional transformational analysis of the raising verbs (see Frank (2002) for a discussion). In the dependency tree corresponding to the left-context, the word *seems* changes the relation between *claims* and the underspecified verb of the embedded clause $\_v_2$. In fact, after the fifth operation *seems* depends on *claims* and $\_v_2$ depends on *seems*.
The sixth and seventh steps of the derivation are two *Shift* operations, which introduce the lexical items *to* and *adore*. For each left-context of the derivation, the corresponding dependency tree obtained by the conversion algorithm described above is reported in Figg. 13 and 14.

4. Conclusions

This paper has presented a dynamic approach to syntax that implements full connectivity of partial syntactic structures at the competence level. The formalism is a version of Tree Adjoining Grammar, called DVTAG, which exploits the extended domain of locality and the adjoining mechanism to implement incrementality at a very eager pace. DVTAG further extends the domain of locality with the so-called predicted nodes, which are necessary to guarantee the full connectivity of partial structures, and introduces variants of substitution and adjoining to cope with the strict left–to–right processing of the sentence. Finally, DVTAG introduces a novel syntax–semantics interface consisting of a dependency tree that results from the relations stated over head projections in the derived tree.

The paper has focused on the linguistic adequacy of DVTAG. In particular, we have showed that the restrictions posed by the dynamic approach can be overcome through elementary tree conversions (from left to right branching), preservation of the lexical dependencies widely assumed in LTAG and use of the wrapping operation, recently introduced in the TAG formalism. The syntax–semantics interface allows an adequate description of dependencies in the TAG controversial case of bridge and raising verbs.

* In this step note that since the head-feature belongs only to the bottom feature set, $\_v_3$ is not unified with "−" (cf. the head-feature update mechanism in sec. 2.1).

The linguistic adequacy of DVTAG must be further investigated to provide a better understanding of the limits of the dynamic approach. Cases like the partitive constructions (XTAG Research Group, 2001), as well as coordination, need to receive serious consideration, though these issues are still controversial even in more established linguistic formalisms.

One question that can be raised naturally is the increase in the size of the lexicon, since the elementary trees with the predicted nodes add on the standard LTAG lexicon. This issue has received preliminary investigations in previous work (Lombardo et al., 2004; Mazzei and Lombardo, 2005), but the consequences on parsing ambiguity still need to be explored.

References

Abney, S. P. and M. Johnson: 1991, 'Memory Requirements and Local Ambiguities of Parsing Strategies'. *Journal of Psycholinguistic Research* **20**(3), 233–250.

Aoshima, S., C. Phillips, and A. Weinberg: 2004, 'Processing filler-gap dependencies in a head-final language'. *Journal of Memory and Language* **51**, 23–54.

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

Demers, A. J.: 1977, 'Generalized Left Corner Parsing'. In: *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. New York NY, pp. 170–182, ACM Press.

Doran, C., B. Hockey, A. Sarkar, B. Srinivas, and F. Xia: 2000, 'Evolution of the XTAG system'. In: A. Abeillé and O. Rambow (eds.): *Tree Adjoining Grammars*. Chicago Press, pp. 371–405.

Dras, M., D. Chiang, and W. Schuler: 2004, 'On Relations of Constituency and Dependency Grammars'. *Research on Language and Computation* **2**(2), 281–305.

Frank, A. and J. van Genabith: 2001, 'Glue TAG: Linear Logic based Semantics for LTAG'. In: *Proceedings of the LFG01 conference*.

Frank, R.: 2002, *Phrase Structure Composition and Syntactic dependencies*. The MIT Press.

Joshi, A.: 1990, 'Processing crossed and nested dependencies: An automaton perspective on the psycholinguistic results'. *Language and Cognitive Processes* **5**(1), 1–80.

Joshi, A.: 2004, 'Starting with Complex Primitives Pays Off: Complicate Locally, Simplify Globally'. *Cognitive Science* **28**(5), 637–668.

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

Joshi, A. and Y. Schabes: 1997, 'Tree-Adjoining Grammars'. In: G. Rozenberg and A. Salomaa (eds.): *Handbook of Formal Languages*. Springer, pp. 69–123.

Kamide, Y., G. T. M. Altmann, and S. L. Haywood: 2003, 'The time-course of prediction in incremental sentence processing: Evidence from anticipatory eye movements'. *Journal of Memory and Language Language* **49**, 133–156.

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

Kroch, A.: 1989, 'Asymmetries in long distance extraction in a Tree Adjoining Grammar'. In: M. Baltin and A. Kroch (eds.): *Alternative Conceptions of Phrase Structure*. Chicago: University of Chicago Press, pp. 66–98.

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

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

Lin, D.: 1995, 'A Dependency-based Method for Evaluating Broad-Coverage Parsers'. In: *IJCAI95*.

Lombardo, V., A. Mazzei, and P. Sturt: 2004, 'Competence and Performance Grammar in Incremental Parsing'. In: *Incremental Parsing: Bringing Engineering and Cognition Together, Workshop at ACL-2004*. Barcelona, pp. 1–8.

Lombardo, V. and P. Sturt: 1997, 'Incremental parsing and infinite local ambiguity'. In: *XIXth Cognitive Science Society*.

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

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

Magerman, D.: 1995, 'Statistical decision-tree models for parsing'. In: *ACL95*. pp. 276–283.

Marslen-Wilson, W.: 1973, 'Linguistic structure and speech shadowing at very short latencies'. *Nature* **244**, 522–523.

Mazzei, A.: 2005, 'Formal and empirical issues of applying dynamics to Tree Adjoining Grammars'. Ph.D. thesis, Dipartimento di Informatica, Università degli studi di Torino.

Mazzei, A. and V. Lombardo: 2005, 'Building a wide coverage dynamic grammar'. In: *Proc. of IX Congresso Nazionale Associazione Italiana per L'Intelligenza Artificiale (Lectures Notes in Artificial Intelligence 3673)*. Milano, pp. 303–314.

Mazzei, A., V. Lombardo, and P. Sturt: 2005, 'Strong connectivity hypothesis and generative power in TAG'. In: *Proc. of The 10th conference on Formal Grammar and the 9th Meeting on Mathematics of Language*. Edinburgh, pp. 169–184.

Mel'cuk, I.: 1987, *Dependency syntax : theory and practice*. State University Press of New York.

Milward, D.: 1994, 'Dynamic Dependency Grammar'. *Linguistics and Philosophy* **17**(6), 561–604.

Phillips, C.: 1996, 'Order and Structure'. Ph.D. thesis, MIT.

Phillips, C.: 2003, 'Linear Order and Constituency'. *Linguistic Inquiry* **34**(1), 37–90.

Rambow, O. and A. Joshi: 1997, 'A formal look at dependency grammars and phrase structure grammars, with special consideration of word-order phenomena'. In: *Recent Trends in Meaning-Text Theory*. Amsterdam and Philadelphia: John Benjamins, pp. 167–190.

Rambow, O., D. Weir, and K. Vijay-Shanker: 2001, 'D-Tree Substitution Grammars'. *Computational Linguistics* **27**(1), 87–121.

Resnik, P.: 1992, 'Left-Corner Parsing and Psychological Plausibility'. In: *COLING92*. Nantes, pp. 191–197.

Roark, B.: 2001, 'Probabilistic Top-Down Parsing and Language Modeling'. *Computational Linguistics* **27**(2), 249–276.

Schabes, Y. and R. Waters: 1995, 'Tree Insertion Grammar: A Cubic-Time, Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Trees Produced'. *Computational Linguistics* **21**(4), 479–513.

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

Stabler, E. P.: 1994, 'The finite connectivity of linguistic structure'. In: C. Clifton, L. Frazier, and K. Reyner (eds.): *Perspectives on Sentence Processing*. Lawrence Erlbaum Associates, pp. 303–336.

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

Sturt, P.: 1997, 'Syntactic Reanalysis in Human Language Processing'. Ph.D. thesis, University of Edinburgh, UK.

Sturt, P. and V. Lombardo: 2005, 'Processing coordinated structures: Incrementality and Connectedness'. *Cognitive Science* **29**(2), 291–305.

Thompson, H. S., M. Dixon, and J. Lamping: 1991, 'Compose-reduce parsing'. In: *ACL91*. pp. 87–97.

Vijay-Shanker, K.: 1987, 'A study of Tree Adjoining Grammars'. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA.

XTAG Research Group: 2001, 'A Lexicalized Tree Adjoining Grammar for English'. Technical Report IRCS-01-03, IRCS, University of Pennsylvania.