Towards a dynamic constituency model of syntax

Alessandro Mazzei, Vincenzo Lombardo

Dipartimento di Informatica, Università di Torino

C.so Svizzera, 185 10149, Torino, Italy

[mazzei|vincenzo]@di.unito.it

1 introduction

Incrementality is a basic feature of the human language processor. There is a considerable amount of objective data that humans build-up interpretations of the sentences before perceiving 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, incremental construction of syntactic relations (see, for example Kamide et al. (2003); Sturt and Lombardo (2005)) and which components of a syntactic parser account for incrementality. Steedman proposed a general anatomy to classify a syntactic parser (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, back-tracking, dynamic programming). The oracle takes into account the ambiguity of syntactic analysis and ranks the possible alternatives for the parsing algorithm through a rule-based or probabilistic strategy.

In most incremental parsing models, the competence grammar is usually a standard generative formalism, e.g. context-free grammar (Roark, 2001) or a categorial formalism, e.g. Combinatory Categorial Grammar (Steedman, 2000). As a consequence, the parsing strategy is the component that totally specifies the incremental nature of the syntactic process (i.e. it 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 proposed a model of grammar in which the parsing strategy is part of the competence grammar (Phillips, 1996). His fundamental hypothesis (PiG, parser is grammar) is that a grammar is a formal generative device, that specifies what are the grammatical structures (i.e. the legal constructions) in terms of their computation. He has pointed out that in this model the notion of constituency is variable during the analysis of the sentence: a fragment of the sentence that is a constituent at some point in the derivation, may not be a constituent after the processing of a subsequent word (Phillips, 2003). Phillips’ fundamental hypothesis ties the notion of constituency to the parser’s computation. Constituents have a variable status during the analysis of the sentence and depend on the parsing state. The only difference between a parser and grammar is that the former has to take into account the finiteness of the resources, while the latter does not.

An example of a categorial grammar oriented approach that follows PiG hypothesis is proposed by Milward in the context of dynamic grammars, in which the
syntactic analysis is viewed as a dynamic process (Milward, 1994). In this model the syntactic process is a sequence of transitions between adjacent syntactic states $S_i$ and $S_{i+1}$, while moving from left to right in 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. Both Phillips and Milward can give a competence account for non-constituency coordination: two fragments $x$ and $y$ of the sentence are grammatical conjuncts if they continue in the same way from the same starting state.

Assuming that the competence grammar specifies the parsing strategy, there are several strategies that model the incremental nature of the syntactic process. Abney and Johnson proposed to model the difficulty of the syntactic analysis by means of the memory resources required to store the disconnected partial structures built during the analysis (Abney and Johnson, 1991). It has been argued that the most parsimonious account of psycholinguistic experimental data is to assume that each new word is connected to the syntactic structure as soon as the word is perceived (Kamide et al., 2003; Sturt and Lombardo, 2005). A formalized version of this property of the syntactic processor, that we call strong connectivity, has been proposed 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).

In this paper we present a grammatical formalism such that the parsing strategy is totally determined by the competence grammar and that fulfills the strong connectivity hypothesis for simplifying the syntax-semantic interface. In particular it
is a dynamic version of Tree Adjoining Grammars (TAG, Joshi and Schabes (1997)), called DVTAG. TAG is a family of generative formalism that uses tree elementary structures rather than string elementary structures, as in context-free grammars. Moreover, TAG uses two distinct rewriting operations. A substitution operation, that is the attachment of an elementary trees on a node of the fringe of another elementary tree. An adjoining operation, that grasp an elementary tree into an internal node of another elementary tree. This latter operation increases the generative power of TAG with respect to context-free grammars. Indeed DVTAG generates the class of the mildly context-sensitive languages, that is supposed to be the class where the human languages fall in the Chomsky hierarchy (Vijay-Shanker et al., 1990).

In the section 2 we present some previous dynamic model of syntax. In section 3 we describe the essential feature of TAG and of our dynamic version of TAG. Finally in section 4 we point out some open issues.

2 Generative and dynamic models of syntax

Generative grammars perspective on syntax follows the distinction between competence and performance given by Chomsky:

Linguistic theory is concerned primarily with an ideal speaker-listener, in a completely homogeneous speech-communication, who knows its (the speech community’s) language perfectly and is unaffected by such grammatically irrelevant conditions as memory limitations, distractions, shifts of attention and interest, and errors (random or characteristic) in applying his knowledge of this language in actual performance (Chomsky, 1965).
We can identify in the competence what the grammar computes and in the performance how the grammar does the computation. A generative grammar is a tuple $G=(\Sigma, V, S, P)$, where $\Sigma$ is a set of terminal symbols (i.e. the lexicon), $V$ is a set on non-terminal symbols (i.e. the constituents), $S \in V$ is the start symbols (i.e. the sentence constituent), $P$ is a set of production rules of the form $A \rightarrow \gamma$ with $A \in V$ and $\gamma \in (V \cup \Sigma)^+$. We define a sentential-form of the grammar $G$ to be any sequence of grammar symbols (terminals or nonterminals) derived in 0 or more steps from $S$. A sentence is a sentential-form which contains only terminal symbols. The language defined by grammar $G$ is the set of all sentences which can be derived from the start symbol of $G$. Formally, we define $\Rightarrow$ a relation between two sentential-forms $\alpha$ and $\beta$, $\alpha \Rightarrow \beta$, such that $\beta$ can be obtained from $\alpha$ by applying a production rule of $G$. Moreover we define $\Rightarrow^*$ the transitive-reflexive closure of $\Rightarrow$. Finally we say that a sentence $w_1 w_2 ... w_n$ is generated by $G$ if and only if $S \Rightarrow^* w_1 w_2 ... w_n$, i.e. there is a sequence of sentential forms such that $S$ can be rewritten into $w_1 w_2 ... w_n$.

In the standard generative grammar approach the competence information is only declarative. There are no constraints on the production rules order in the derivation of the sentence, then there are no constraints on sentential-forms transition $\Rightarrow$. The key point is that the competence specifies the production rules $\rightarrow$ but does not specify any constraint on derivation $\Rightarrow$. A limit of the standard generative approach is that some syntactic phenomena, as the non-constituency coordination, can be straightforwardly accounted for in terms of sentential-forms. By assuming the competence-performance dichotomy, we arrive to the contradiction that we consider a
performance issue, i.e. the sequence of sentential forms, to state on the grammaticality of the sentence, i.e. a competence issue.

Milward proposed a different paradigm. He investigated about the relation between incrementality and non-constituency coordination (Milward and Cooper, 1994), by proposing a dynamic model of the syntax: in this model the syntactic process is a sequence of transitions between adjacent syntactic states $S_i$ and $S_{i+1}$, while moving from left to right in the input string. The syntactic state contains all the syntactic information about the fragment already processed. Using this paradigm, Milward assumes that the coordination of two fragments $x$ and $y$ of the sentence is grammatical whether the two fragments update in the same way some start state. Consider the sentence

\[ \text{Sue lent Joe a book and Ben a paper} \]  \hspace{1cm} (1)

In the Milward’s paradigm, after the derivation of the fragment \textit{Sue lent} the system is in the syntactic state $S_1$. The coordination of the fragments \textit{Joe a book} and \textit{Ben a paper} depends on the fact that both fragments update $S_1$ in the same way, i.e. that after the fragments \textit{Sue lent Joe a book} and \textit{Sue lent Ben a paper} the system is in the same syntactic state $S_2$. In contrast to the standard definition of generative grammars, the Milward’s definition describes the derivation process as part of the grammar. In other words, the grammar defines the production rules ($\rightarrow$) as well as the derivation process ($\Rightarrow$). Note that the notion of left-context in the Milward’s paradigm corresponds to the notion of sentential-form in the standard generative grammars. A dynamic grammar explicitly includes the derivability of a left-context, whereas this
notion is outside of the grammar in the standard generative formalization. Moreover the dynamic approach allows us to define several rules for derivation: Milward used this feature to take into account the non-constituency coordination in the dynamic dependency grammars formalism (Milward, 1994).

In the next section we define a new formalism that uses Milward’s paradigm: in contrast to (Milward, 1994) we define constituency dynamic grammar rather than a dependency dynamic grammar.

3 From TAG to a dynamic version of TAG

Tree Adjoining Grammar (TAG) is a tree rewriting system (Joshi et al., 1975; Joshi, 1985; Abeillé and Rambow, 2000). The elementary structures of the formalism are trees, and the operations of the formalism are substitution and adjoining. The first operation merges a tree on the frontier of another tree (figure 1-(a)), the second operation merges a tree in the middle of another tree (figure 1-(b)). In a TAG derivation some elementary trees are combined, using substitution and adjoining, to generate a final derived tree.

TAG is an appealing formalism for several reasons. From a linguistic perspective, a tree rewriting system allows us to work with trees rather than strings. Since many linguistic phenomena are modeled by using trees, we can use the elementary trees of the formalism as the elementary component of a linguistic theory. From a formal perspective, a tree rewriting system is more powerful in comparison with a string rewriting system. Shieber showed that the natural languages are mildly context-sensitive (Shieber, 1985), and a tree rewriting system equipped with adjoining can generate exactly the mildly context-sensitive languages.
We can talk about the TAG family, since there are many variants of the TAG formalism, based on different properties of the elementary trees. Here we consider the Lexicalized Tree adjoining Grammar (LTAG), that associates each elementary tree with a lexical item, called the anchor of the tree. A LTAG grammar consists in a set of elementary trees, divided into initial trees (usually denoted by the $\alpha$ symbol) and auxiliary trees (usually denoted by the $\beta$ symbol). An initial tree represents the domain of the argument structure of its anchor. For example, in figure 2-(a), there are three initial trees, anchored in John, likes and beans respectively. Notice that the elementary tree for likes includes substitution nodes corresponding to the subject and the object of likes. Auxiliary trees are usually used for modification. Each auxiliary tree includes a foot node on its frontier, which is of the same syntactic category as the root of the tree. For example, in Figure 2-(a), the auxiliary tree anchored by from includes root and foot nodes labeled with NP. There are two operations for combining trees: with substitution, the root of an initial tree is substituted at a substitution node in another elementary tree; with adjoining, an auxiliary tree is inserted at some adjoining node in the structure of another elementary tree, such that the subtree dominated by the adjoining node is substituted at the foot node of the auxiliary tree, and the root of the auxiliary tree is substituted at the adjoining node. For example, the auxiliary tree anchored in from in Figure 2-(a) can be adjoined at the NP node indicated in the figure. LTAG is attractive because it allows for an elegant characterization of argument structures and syntactic dependencies. In particular, for any LTAG derivation, it is possible to build a derivation tree (Figure 2-(c)), which consists of a history of the combination operations that the derivation has used. Given that substitution and adjoining are used to attach arguments and modifiers respectively, and that the lexicalized character of LTAG assigns a lexical anchor to
each elementary tree, the derivation tree is therefore commonly interpreted as a
dependency structure (Rambow and Joshi, 1997; Joshi and Rambow, 2003), which
can then be the basis for semantic interpretation. The method used to build the
derivation tree is to attach the identifiers of the substituted and adjoined elementary
trees as daughters of the tree identifiers where substitutions and adjoining have
operated upon.

Now we describe three fundamental properties of LTAG: all the mathematical
and linguistic properties of LTAG are consequences of these three features (Joshi and
Schabes, 1997).

**Extended domain of locality (EDL)** LTAG has a greater domain of locality in
comparison with context-free grammars. The domain of locality of a grammatical
formalism is the extension of the elementary structure. Context-free grammars uses
rewriting rules, that can be considered as single level trees, i.e. tree with one parent
and several children. In this case, the syntactic relations that can be specified in a
single rule, are those dependencies that hold between the parent and the children, or
those that hold between two children. For instance, with respect to the tree \( \alpha \) of
figure 2-(a), it is impossible with a context-free rule to specify the syntactic relation
stating between the verb and the first and second noun phrases, to specify a sub-
categorization frame of the verb, and at the same time to account for the verbal phrase
constituent. In other words, if we want to embed the verb and the two noun phrases in
a single context-free rule (e.g. \( S \rightarrow NP \ V \ NP \)), we lose the internal structure of the
verbal phrase, that is considered as essential in a number of linguistic theories. On
contrary, in LTAG this is possible because the elementary structures are multilayered
trees, and then the dependency can be defined on the different levels of the trees. In
this way the formalism is able to state in the elementary structures several syntactic relations as sub-categorization, filler-gap, agreement.

Factoring recursion from domain by Adjoining (FRD) The operation of adjoining allows to reduce the long-distance syntactic dependencies to the local dependencies defined on the elementary structures. In other words, “the long-distance behavior of same dependencies follows from the operation of adjoining, thus factoring recursion from the domain from which dependencies are initially stated” (Joshi and Schabes, 1997).

Lexicalization (LEX) A grammatical formalism is lexicalized if each elementary structure of the formalism is associated with a lexical item, that will be called anchor of the corresponding structure. Several mathematical properties of LTAG are a direct consequence of the lexicalization, e.g. the finite ambiguity of the number of derivations, and then the decidability of the recognition problem (Joshi and Schabes, 1997). The lexicalization has also important linguistic consequence: the whole tree can be considered as an extended projection of the anchor (Frank, 2002).

LTAG does not take into account the incrementality of human languages as a competence feature. LTAG adheres to the generative paradigm and fulfills the competence-performance dichotomy. As a consequence the incrementality is explained at the performance level in most of the work about LTAG and incrementality. In other words, many incremental TAG parsing algorithms have been proposed in the last twenty years (Schabes, 1990; Shieber and Johnson, 1993; Nederhof, 1999), but to our knowledge DVTAG is the first attempt to consider the issue of incrementality straightforwardly in the grammar definition. In DVTAG the dynamic process encodes the strong connectivity hypothesis in a partial structure called “left-context”, which spans a left fragment of the sentence, and is expanded by
inserting elementary structures anchored by the words that occur at the immediately right of the left context. In Fig. 3 we can see the DVTAG derivation of the sentence *John loves Mary madly*. Like LTAG (Joshi and Schabes, 1997), a DVTAG 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.

In standard LTAG the lexical dependencies are directly represented in the derivation tree, and are established when the two heads involved are both introduced into the syntactic structure. In DVTAG, in contrast, the introduction of predicted projections carries constraints on head dependencies that are yet to come. So, in order to represent such constraints, we have to augment each node with a feature indicating the lexical head that projects that particular node. The head variable is a variable in logic terms: \( \_v_3 \) is unified with the constant *loves* in the derivation of Fig. 3.

The derivation process in DVTAG builds a constituency tree by combining the elementary trees via some operations that are illustrated below. DVTAG 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 each step is an updated partial structure. 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 \( i \). In DVTAG 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, and it
will subsequently have to be combined with some elementary tree on the right. At the end of the derivation process the left-context spans the whole sentence, and is called the *derived tree*: the last tree of Fig. 3 is the derived tree for the sentence *John loves Mary madly*.

In DVTAG we always combine a left-context with an elementary tree. Thus there are seven attachment operations. Three operations (1. substitution, 2. adjoining from the left, 3. adjoining from the right) are called *forward operations* because they insert the current elementary tree into the left-context; three other operations (4. inverse substitution, 5. inverse adjoining from the left, 6. inverse adjoining from the right) are called *inverse operation* because they insert the left-context into the current elementary tree; the seventh operation (7. shift) does not involve any insertion of new structural material.

The *substitution* operation (1.) is similar to the LTAG substitution, where an initial tree is inserted into a substitution node in the left-context. In Fig. 3 we have a substitution of the elementary tree anchored by *Mary*.

Standard LTAG adjoining is split into two operations: *adjoining from the left* (2.) and *adjoining from the right* (3.). The type of adjoining depends on the position of the lexical material introduced by the auxiliary tree in relation to the material currently dominated by the adjoined node (which is in the left-context). In Fig. 3 we have an adjoining from the right in the case of the right auxiliary tree anchored by *madly*, and in Fig. 4 we have an adjoining 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* (4.) the left-context replaces a
substitution node in the elementary tree. In Fig. 3 we have an inverse substitution in the case of the initial tree anchored by John.

In the case of inverse adjoining from the left (5.) and inverse adjoining from the right (6.), 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.

Finally, the shift operation (7.) either scans a lexical item which has been already introduced in the structure or derives a lexical item from some predicted heads. The grounding of the variable \( _v_1 \) in Fig. 4 is an example of shift.

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 operations: given the \( i-1 \)-th word in the sentence we can 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 can compute a set of accessible nodes in the elementary tree (the elementary tree fringe). To take into account this feature, the elementary tree in DVTAG is a dotted tree, i.e. a pair \( \gamma, i \) formed by a tree \( \gamma \) and an integer \( i \) denoting the accessible fringe\(^4 \) of the tree (Mazzei, 2005).

The DVTAG derivation process requires the full connectivity of the left-context at all times and hence satisfies SCH. 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 DVTAG derivation of John said that tasty apples 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 predicted 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 Fig.4). The
elementary tree *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 non-terminal node is referred by a set of lexical items, that represent a
predicted head. 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.  

4 Conclusions and open issues

In this paper we analyzed some issues of the standard generative paradigm for
computational models of syntax. We pointed out that the standard approach relies on
Chomsky dichotomy competence-performance, and that some well known open
problems are hard to resolve in this approach. Moreover we described the basic
features of a different approach, i.e. dynamic grammars. By using this feature we
devised a new grammatical formalism called DVTAG. This dynamic formalism
belongs to the TAG family, i.e. is a tree based rewriting system that uses adjoining
operation. To our knowledge DVTAG is the first attempt to define a constituency
based dynamic grammar. Following the working road depicted by Milward for
dependency dynamic grammars, in future work we intend to investigate about the
non-constituency coordination in DVTAG.
Notes

We stress our discussion about parsing. Similar arguments have been made for the generation process.

2 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.

3 With the aim to simplify the exposition, we omit the dynamic wrapping operation defined in (Mazzei, 2005; Mazzei et al., 2005).

4 In the picture we represent the integer using a dot. Note that fringe and dotted tree are two concepts borrowed from parsing (cf. Sikkel, 1997)) as a consequence of the dynamic nature of DVTAG.

5 In contrast with the elementary tree anchored by Bill the elementary tree anchored by John does not introduce any predicted head. In these derivations we have decided to use the expanded tree only when it is necessary in order to guarantee the connectivity, i.e. we have decided to delay until necessary the prediction of the heads in the left-to-right derivation of the sentence. One of the consequences of this choice is a difference in between elementary trees involved in main and embedded clauses.
Figures

Figure 1: Substitution (a) and adjoining (b) TAG operations. In the substitution a tree is merged on the node $X_1$ on the frontier of another tree. In the adjoining a tree is merged on the node $X$ in the middle of another tree.

Figure 2: Elementary trees (a), derived tree (b) and derivation tree (c), for the sentence *John likes beans from France.*
Figure 3: The DVTAG derivation of the sentence *John loves Mary madly*. The operations used in this derivation are shift (Shi), inverse substitution (Sub$^-$), substitution (Sub$^+$) and adjoining from the right ($\nabla^R$). The abbreviation under the arrows denotes the elementary dotted tree, the left-context, and eventually the adjoining node (Gorn address) involved the operation.
Figure 4: The DVTAG derivation of the sentence *Bill often pleases Sue*. The operations used in this derivation are shift (Shi), adjoining from the left ($\nabla_L^-$), shift (Shi) and substitution ($Sub^-$).
References


K. Vijay-Shanker, A. Joshi, and D. Weir. The convergence of mildly context-sensitive grammatical formalisms. In Peter Sells, Stuart Shieber, and Tom