

EXTENDED PARTIAL PARSING FOR LEXICALIZED TREE GRAMMARS

Patrice Lopez
DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrcken

lopez@dfki.de

Abstract

Existing parsing algorithms for Lexicalized Tree Grammars (LTG) formalisms (LTAG, TIG, DTG, ...) are adaptations of algorithms initially dedicated to Context Free Grammars (CFG). They do not really take into account the fact that we do not use context free rules but partial parsing trees that we try to combine. Moreover the lexicalization raises up the important problem of multiplication of structures, a problem which does not exist in CFG. This paper presents parsing techniques for LTG taking into account these two fundamental features. Our approach focuses on robust and practical purposes. Our parsing algorithm results in more extended partial parsing when the global parsing fails and in an interesting average complexity compared with others bottom-up algorithms.

1 Introduction

Lexicalized Tree Adjoining Grammars (LTAG) [Joshi and Schabes, 1992] have given rise to a lot of interests for the modeling of syntax, in particular thanks to its three “key” properties: the principle of Extended Domain of Locality (EDL), the lexicalization and the representation of recursive phenomena using the operation of adjunction. Since these properties allow LTAG to capture semantic dependencies in elementary trees, the parsing result (the derivation trees) is closed to semantic dependency trees which is a relevant structure for post-parsing processes [Candito and Kahane, 1998].

This paper focuses on robust chart parsing with Lexicalized Tree Grammars. We do not consider here probabilistic approximation but only complete or partial structures obtained with valid rule applications. The classical parsing algorithms for LTAG, for instance CKY-like algorithm [Vijay-Shanker, 1987], Head-Corner [van Noord, 1994] or Earley-like algorithm [Schabes, 1994] focus on the parsing of complete grammatical utterances. We argue that an algorithm dedicated to the parsing of a Lexicalized Tree Grammar can take into account more efficiently EDL and lexicalization in order to (1) obtain beneficial extended partial results if the global parsing fails, (2) decrease the average complexity of the analysis obtained with bottom-up parsers.

The constraints expressed with lexicalized elementary trees are richer than the constraints captured with a set of rewriting context free rules. The *extended domain of locality* is the fact that an elementary tree encodes directly a syntactical substructure view as a partial parsed tree. It allows to define constraints in more than one level of the parsing tree as compared to context free rules and permits to use atomic features. One way to exploit this property during parsing is to consider for instance co-anchors which are often neglected in existing parsing algorithm. Co-anchors encode cooccurrence information which are significant for parsing. Co-anchors are frequently used in the French LTAG grammar for prepositional complements and idioms [Abeillé et al., 1999]. They could also be massively

used in elementary trees obtained with Explanation Based Learning (EBL) methods used to speed up parsing.

The *lexicalization* imposes that each elementary tree contains at least one lexical terminal symbol called the *anchor*. This constraint permits to represent in each elementary tree one of the syntactical contexts of a lexical entry. This property is usually exploited during a pre-parsing process which consists in the selection of the sub-set of elementary trees that can anchor at least one of the words of the input sentence. But such a property results also in a lot of duplication of the same sub-structures in the grammar which does not exist in a CFG. Existing parsers usually ignore this drawback and result in a lot of redundant computations.

Considering the syntactic level, the two main differences between parsing a sentence using a formal grammar and a grammar dedicated to natural language are the necessity to take into account the ambiguity of the language and the potential “out of grammar” words and structures. Addressing the problem of ambiguities, efficient results preserving all valid rule applications has been obtained using tabular parsing technics: The result stored in a chart is a shared parse forest [Lang, 1991]. From the point of view of robustness, it is important to be able to implement local analyses, i.e. to try to extract a maximum of information from the utterance (at least all potential constituents) even if the complete parsing failed. Prediction usually speeds up a parser. However as explained for example in [Magerman and Weir, 1992], Earley-style prediction can improve the average case performance but it has serious impacts on robust processing of incomplete and ungrammatical sentences which are very common in natural language systems. The same limit can be observed for LR-style parsing results as [Nederhof, 1998].

As a consequence, we present here a new tabular parsing algorithm for LTAG called *connection driven parsing*. This incremental bottom-up algorithm could be applied to other kind of LTG. Classical top-down predictions could be used to improve the parsing but they would decrease robustness.

2 Connection driven parsing technics

2.1 Lexicalized Tree Grammars

A Tree Adjoining Grammar is specified as a quintuplet $G = (\Sigma, NT, I, A, S)$, where Σ is a set of terminal symbols, NT the disjoint set of nonterminals including the start symbol S, and where I and A are two finite sets of trees called respectively *initial trees* and *auxiliary trees*. The set $I \cup A$ gathers the *elementary trees*. We consider the standard definition of completely Lexicalized TAG in which each tree contains at least a leaf node, called the *anchor*, which corresponds to a word. For the complexity results we note N the maximum number of nodes of an elementary tree, G the size of the set $I \cup A$ and n the length of the input string to parse.

We note a substitution node with its category and the mark \downarrow , internal nodes without mark, root nodes with Δ and anchors with the lexical mark \diamond . Finally, we note $*_l$ (resp. $*_r$) the foot node of a left (resp. right) auxiliary tree and $*$ the foot node of a wrapping auxiliary tree. In order to make explanations easier, we will not consider non-adjunction constraints on nodes. A Tree Inserted Grammar (TIG) is a LTG which does not include any wrapping auxiliary tree and supposes that this kind of tree never appears dynamically during the parsing. Using a TIG the worst case complexity of the parsing is decreased to (n^3) as all derivations are equivalent to Context Free derivations [Schabes and Waters, 1995].

2.2 Limits of CF algorithms invariant

One of the most important characteristic of a parser for natural language is its ability to deliver the richest possible partial parsing results when the complete parsing fails. Diagnostic, repairing and interpretation are much easier when we can exploit informative partial derivation trees. When we use a chart parsing algorithm, the partial results are given by the items which represent a chunk of well recognized words. The nature of a partial result is given by the invariant of the algorithm which characterizes the properties of each produced item.

In CFG algorithms, the invariants are based on subtree recognition: Each item represents one position in a elementary tree and a portion of the string to parse. The classical invariants impose that the sub-tree dominated by the dotted node associated to an item is completely parsed [Shieber et al., 1995]. Non predictive rules combine complete subtrees into larger ones preserving this invariant.

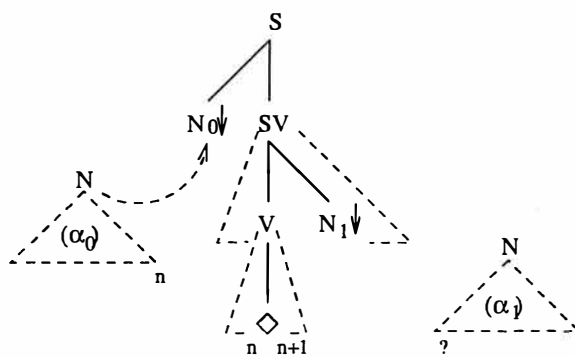


Figure 1: Bottom-up tree walk and subtree invariant.

In figure 1, we indicate different subtrees which have to be completely recognized to allow the processing of the parsing of the elementary tree. Now we suppose that the utterance to be parsed is agrammatical because of an unexpected component on the right (at position marked with a ?). The subtree α_1 on the right is not adjacent to the anchor of the elementary and so the attachment on substitution node N_1 is impossible. The subtree dominated by the node SV can not be complete and the parsing process of this elementary tree must stop, even if the subtree α_0 can be combined by adjacency to the substitution node N_0 . Considering a bidirectional parsing as a CKY type or Head Driven type [Lavelli and Satta, 1991] [van Noord, 1994] when an unexpected phenomena occurs the classical invariant stops the parsing process on both sides of the current recognized subtree even when it is possible to continue the parsing on one side, resulting in poorer partial results.

Our first proposition is to focus the parsing on recognized islands and not on recognized subtree and to allow real bidirectional extensions. Since tabular technics impose adjacency of items to combine, we will see that we can take into account the topology of the trees to decrease the average complexity with a new level of granularity for a linearized tree called *connected route*.

2.3 Finite States Automata representation of an elementary tree

The existing representations of the parsing process of a elementary tree are dotted tree [Vijay-Shanker, 1987] and dotted rules [Nederhof, 1997]. In both case this representation is constrained by their locality. We propose an alternative representation that allows to express constraints

of significant parts of the tree.

The linearization of a tree can be represented with a Finite State Automaton (FSA) as in figure 2. Every tree traversal (left-to-right, bidirectional from an anchor, ...) can be performed on this automaton. The dotted trees used for example in [Schabes, 1994] or [Shieber et al., 1995] are equivalent to the states of these automata.

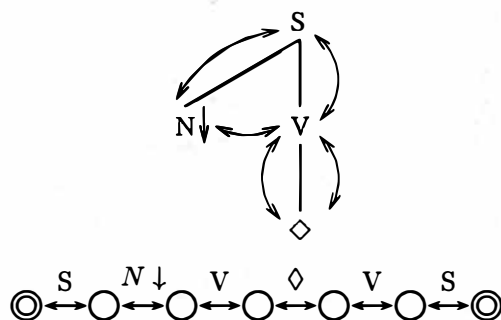


Figure 2: Simple FSA representing an elementary tree for the normal form of an intransitive verb.

We consider the following definitions and notations :

- Each automaton transition is annotated with a category of node, each non-leaf node appears twice in the list of transition framing the nodes which it dominates. In order to simplify our explanation the transition is shown by the annotated category.
- Transitions can be bidirectionnal in order to be able to start a bidirectionnal tree walk of a tree starting from any state.
- Considering a direction of transition (left-to-right, right-to-left) the FSA becomes acyclic.

2.4 Parsing invariant and island representation

A set of FSA corresponds to a global representation of the grammar, for the parsing we use a classical local representation called *item*. An item is defined as a 7-tuple of the following form :

```
state: ( left index, right index,
         left state, right state,
         foot left index,
         foot right index, star state)
```

The two first indices are the limits on the input string of the island (an anchor or consecutive anchors) corresponding to the item. During the initialization, we build an item for each anchor (or co-anchor) present in the input string. An item also stores two states of the same FSA corresponding to the maximal extension of the island on the left and on the right, and *only* if necessary two additional indices for the position of the foot node of a wrapping auxiliary tree and the state *star* corresponding to the node where the current wrapping adjunction have been predicted.

This representation maintains the following invariant: An item of the form $(p, q, \sigma_L, \sigma_R)$ specifies the fact that the linearized tree represented by a FSA Δ is completely parsed between the states σ_L and σ_R of Δ and between the indices p and q . No other attachment on the tree can happen on the nodes located between the anchors p and $q-1$.

2.5 Connected routes

Considering an automaton representing the linearization of an elementary tree, we can define a connected route as a part of this automaton corresponding to the list of nodes crossed successively until reaching a substitution, a foot node or a root node (included transition) or an anchor (excluded transition). Connected route is an intermediate level of granularity when representing a linearized tree: each elementary (or a derived tree) can be represented as a list of connected routes. Since they can represent frontiers of the constituents, considering connected routes during the parsing permits to take into account the topology of the elementary trees and to locate significative nodes for an attachment.

The main advantage of considering connected routes during the parsing is the following: The connected routes located between two consecutive anchors become useless, because there is no place for any operation, and they can be directly eliminated without considering the states they contain. Such elimination will be performed during the parsing each time anchors get close to each other, so each time that an attachment is performed because, by using classical tabular techniques and linearized trees, the parsing is driven by the connection of anchors. This elimination operation is called *co-anchor reduction* and permits to consider less states (so nodes) during the parsing and so less hypotheses to test.

We use the following additional simplified notations and primitives :

- The connected route passing through the state σ_d is noted Γ_d .
- $next(\Gamma)$ (resp. $previous(\Gamma)$) gives the first state of the connected route after (resp. before) Γ according to a left-to-right automaton walk.
- $next(N)$ (resp. $previous(N)$) gives the state after (resp. before) the transition N .
- $head(\Gamma)$ (resp. $tail(\Gamma)$) gives the first right (resp. left) transition of the leftmost (resp. rightmost) state of the connected route Γ .

2.6 Inference rules system

The derivation process can be viewed as inference rules which use and introduce items. The inference rules [Schabes, 1994] have the following meaning: If $item_1$ and $item_2$ are present in the chart and if the conditions are fulfilled then add $item_3$ in the chart *if necessary*:

$$\frac{item_1 \quad item_2}{add \quad item_3} \quad (\text{conditions})$$

We present in tables 1 to 5 the different rules of the parser. We illustrate each rule with an abstract tree combination diagram and the state positions on the automata involved in the rule. The first item can be considered as the functor of the rule, for each items we examine successively the nodes on the left and right connected routes. We just have to test then if another adjacent item fulfills the requirements to be the operand of one of the rules. For the bidirectional parsing we just implement the symmetric rules. Each rule extends the position on the linearized elementary tree instead of climbing up the *spine*¹ of the elementary tree as in [Lavelli and Satta, 1991] or [van Noord, 1994]. Even if the parsing is stopped on one side of the spine, our algorithm allows to reach the root node on the other side (of course if no agrammaticality occurs on this second side) when the classical algorithms would completely stop the parsing on the both sides.

Rules for substitution	
	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> $\circ \leftarrow \dots \leftarrow \overset{p}{\circ} \leftarrow \dots \leftarrow \overset{q}{\circ} \rightarrow \dots \rightarrow \overset{r}{\circ} \rightarrow \dots \rightarrow \circ$ <p style="text-align: center;">σ_L σ_R $next(\Gamma_R)$</p> </div> <div style="border: 1px solid black; padding: 5px;"> $\circ \xleftarrow{N} \dots \leftarrow \overset{q}{\circ} \leftarrow \dots \leftarrow \overset{r}{\circ} \rightarrow \dots \rightarrow \overset{N}{\circ}$ <p style="text-align: center;">$\sigma_{L'}$ $\sigma_{R'}$</p> </div> <p>Substitution case 1:</p> $\frac{(p, q, \sigma_L, \sigma_R) \quad (q, r, \sigma_{L'}, \sigma_{R'})}{(p, r, \sigma_L, next(\Gamma_R))} \quad (tail(\Gamma_R) = N \downarrow \quad (\sigma_{L'} \wedge \sigma_{R'}) \in \Delta \quad \Delta \in I \\ head(\Gamma_{L'}) = N^\Delta \quad tail(\Gamma_{R'}) = N^\Delta)$
	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> $\circ \leftarrow \dots \leftarrow \overset{p}{\circ} \xleftarrow{N} \overset{q}{\circ} \leftarrow \dots \leftarrow \overset{r}{\circ} \rightarrow \dots \rightarrow \circ$ <p style="text-align: center;">$previous(\Gamma_{L'})$ $\sigma_{L'}$ $\sigma_{R'}$</p> </div> <div style="border: 1px solid black; padding: 5px;"> $\circ \xleftarrow{N} \dots \leftarrow \overset{p}{\circ} \leftarrow \dots \leftarrow \overset{q}{\circ} \rightarrow \dots \rightarrow \overset{N}{\circ}$ <p style="text-align: center;">σ_L σ_R</p> </div> <p>Substitution case 2:</p> $\frac{(p, q, \sigma_L, \sigma_R) \quad (q, r, \sigma_{L'}, \sigma_{R'})}{(p, r, previous(\Gamma_{L'}), \sigma_{R'})} \quad (head(\Gamma_{L'}) = N \downarrow \quad (\sigma_L \wedge \sigma_R) \in \Delta \quad \Delta \in I \\ head(\Gamma_L) = N^\Delta \quad tail(\Gamma_R) = N^\Delta)$

Table 1: Inference rules for substitutions.

Substitution, left and right adjunctions are Context Free derivations: These operations only have to consider adjacent substrings. *Co-anchor reduction* rule allows to combine two islands which belong to the same tree. *End of connected route* rule permits to reach positions at the end of current left and right connected route on the automata in order to improve factorization of tabulated items. If we only consider these operations we can parse a TIG with a worst case complexity in $O(n^3)$ because a wrapping auxiliary tree is never built during the parsing.

The parsing of LTAG must deal with non-continuous strings occurring in *wrapping adjunctions*: A wrapping auxiliary tree can match a pair of non-adjacent strings, one on the left of the foot and one on the right of the foot. The parsing of this operation proceeds in two steps: First the foot node initialization which corresponds to the prediction of an adjunction, then the complete recognition of the sub tree dominated by the internal node where the adjunction occurs. The second step deals with 2 additional indices by items to represent the position of the foot node. Two additional cases are necessary but not represented here because of space limitation: the first one initializes first the root node of the auxiliary tree and then extends the island until the foot node. The last one is triggered only by wrapping auxiliary trees which have no anchor at one side of the spine. It initializes at all possible indices the position of the foot node at the side where there is no anchor. The complete inference rule system can be found in [Lopez, 1999a]. Parsing LTAG increases the worst case time complexity from $O(n^3)$ to $O(n^7)$. This worst case complexity can be decreased to $O(n^6)$ by using additional techniques as suggested in [van Noord, 1994].

¹The spine of an elementary tree is the path between the main anchor and the root node of the tree.

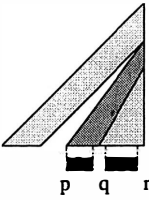
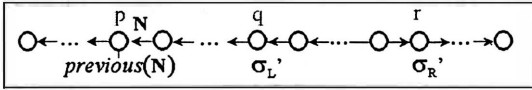
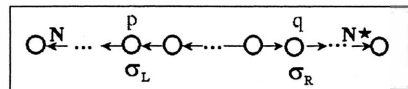
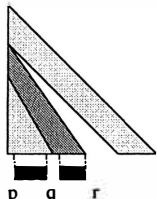
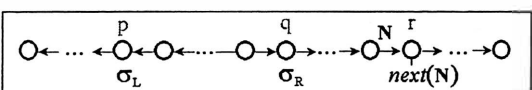
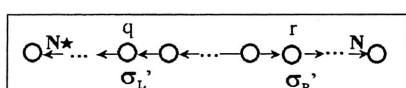
Rules for context free adjunction	
	<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">  </div> <div style="border: 1px solid black; padding: 5px;">  </div>
<p>Left adjunction:</p> $\frac{(p, q, \sigma_L, \sigma_R) \quad (q, r, \sigma_L', \sigma_R')}{(p, r, \text{previous}(N), \sigma_R')} \quad (\exists(N^\Delta \vee N) \in \Gamma_{L'} \quad (\sigma_L \wedge \sigma_R) \in \Delta \quad \Delta \in A)$ <p style="text-align: center;"><i>head</i>(Γ_L) = N^Δ <i>tail</i>(Γ_R) = $N*_l$)</p>	
	<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">  </div> <div style="border: 1px solid black; padding: 5px;">  </div>
<p>Right adjunction:</p> $\frac{(p, q, \sigma_L, \sigma_R) \quad (q, r, \sigma_L', \sigma_R')}{(p, r, \sigma_L, \text{next}(N))} \quad (\exists(N^\Delta \vee N) \in \Gamma_R \quad (\sigma_L' \wedge \sigma_R') \in \Delta \quad \Delta \in A)$ <p style="text-align: center;"><i>head</i>(Γ_L') = $N*_r$ <i>tail</i>(Γ_R') = N^Δ)</p>	

Table 2: Inference rules for context free adjunctions.

2.7 Chart parsing

Items are stored in a chart type data structure, indexed by their indices and combined according to their adjacency, which allows the factorization of items of the chart (tabular techniques). No item can be added that already exists in the chart. We use classical item history that prevents to add in the chart already existing items. The same technique is used to select possible operand items for a rule exploiting mutual exclusion between items corresponding to concurrent lexical hypothesis (same anchors or co-anchors). It allows to obtain a global coherence of an item considering its corresponding tree. We also have used a *subsumption test*, as described in [Satta and Stock, 1989] and suggested for LTAG in [Lavelli and Satta, 1991], in order to limit redundant items due to the bidirectional expansion. Parsing is complete when all possible rules have been executed.

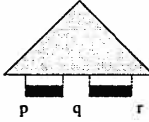
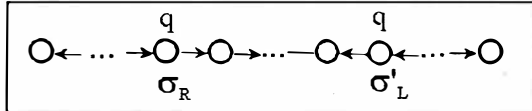
Coanchors reduction rule	
	<div style="border: 1px solid black; padding: 5px;">  </div>
$\frac{(p, q, \sigma_L, \sigma_R) \quad (q, r, \sigma_L', \sigma_R')}{(p, r, \sigma_L, \sigma_R')} \quad (\Gamma_R = \Gamma_L')$	

Table 3: Inference rule for co-anchors reduction.

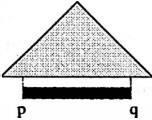
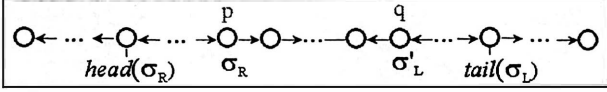
End of connected route rule	
	
$\frac{(p, q, \sigma_L, \sigma_R)}{(p, q, \text{head}(\sigma_L), \text{tail}(\sigma_R))}$	

Table 4: Inference rule for end of connected route.

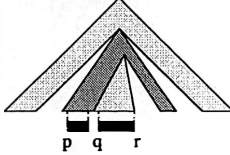
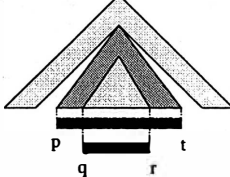
Wrapping adjunction case 1	
	
<p>Foot node initialization:</p> $\frac{(p, q, \sigma_L, \sigma_R) \quad (q, r, \sigma'_L, \sigma'_R)}{(p, r, \sigma_L, \text{next}(\Gamma_R), q, r, N_0)} \quad \left(\exists N_0 / (N_0 \in \Gamma'_L) \wedge (N_0 \in \Gamma'_R) \quad (\sigma_L \wedge \sigma_R) \in \Delta \right)$ $\Delta \in A \quad \text{tail}(\Gamma_R) = N^* \quad \text{head}(\Gamma_L) = N^\Delta$	
	
<p>Wrapping adjunction:</p> $\frac{(p, t, \sigma_L, \sigma_R, q, r, N_0) \quad (q, r, \sigma'_L, \sigma'_R, u, v)}{(p, t, \text{previous}(N_0), \text{next}(N_0), u, v)} \quad \left(\exists N_0 / (N_0 \in \Gamma'_L) \wedge (N_0 \in \Gamma'_R) \right)$	

Table 5: Inference rules for wrapping adjunction case 1.

3 Capturing CF factorizations: LTG sharing

Lexicalization raises the problem of multiplication of the same substructures which can be serious. In CFG the same rule can be used for all possible parsing trees which contain the corresponding substructure, but in LTG this substructure is duplicated. Considering classical linguistic choices, polystructures (for example *to speak to...*, *speak about...*, *to speak to ... about...*) are very common. The corresponding elementary trees must share common substructures and therefore do not cost as much as an independent elementary tree for each. As chart parsing can reduce the complexity in n exploiting factorisation of chart items by their positions on the string and on the elementary trees, we can reduce the average complexity in G using compaction of common substructures because such common structures of elementary trees imply common actions of the parser. For example [Nederhof, 1998] addresses this problem for LR parsing algorithm.

[Evans and Weir, 1998] shares different substructures of elementary trees using FSA and classical minimalization techniques. As presented in [Evans and Weir, 1997], the authors use automata corresponding to one particular traversal of the trees. We use similar techniques to share here linearized structures between different elementary trees. The main differences are that, since it represents el-

elementary trees for any kind of tree walk, this FSA does not impose a specific strategy during the parsing and makes it possible to exploit the granularity of connected route.

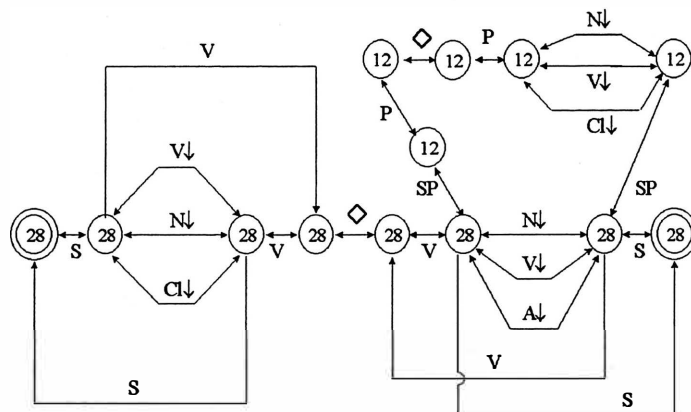


Figure 3: FSA representing 28 elementary trees corresponding to some intransive and transitive contexts.

Figure 3 gives an example of a minimized FSA which allows to share 28 elementary trees. The number in a state indicates how many trees pass through that state. Table 6 gives the resulting compaction statistics.

automaton	no. of trees	no. of states	no. of transitions	trees per state
divided	28	273	245	1
shared	28	13	23	21.84

Table 6: An example of Lexicalized Tree Grammar compaction

When FSA are shared in a single one, each state contains identifiers of the elementary trees which pass through it and each item the list of elementary tree identifiers valid for the item's positions. To test conditions of a rule we must consider every possible transitions paths according to connected routes and the shared FSA. The resulting item of a rule is valid for a subset of identifiers of the elementary trees passing through the both position states. The "uncompaction" can be done when we enumerate the derivations.

4 Implementation and results

The algorithm has been implemented in Java and is integrated in a graphical environment system dedicated to grammar design and parsing tests. The parsing workbench allows to test a grammar on corpus and to compare different parsing algorithms for LTAG. The connection driven parsing algorithm and the graphical workbench will be freely available with an open-source licence by the end of 1999. The connection driven parser includes derivation enumeration from the shared parse forest, features unification but not yet sharing of automata.

We present in table 7 statistics for the chart parsing of a corpora of 861 utterances in French of transcribed spontaneous spoken language with a Sun Ultra 1. We compared two sets of rules: One for the bottom-up connected driven parsing and one for a generalized CKY algorithm for LTAG with predictions as suggested by [Vijay-Shanker and Weir, 1993], both on the same data, with the same

chart implementation and without agenda. The LTAG grammar for the sublanguage corresponds to a syntactical lexicon of 819 entries and a set of 85 non-instanced elementary trees.

corpus	% complete parses	average no of parses/utter.
Gocad	78.31	2.06

Table 7: Global results of the complete parsing

A partial result corresponds here to the maximal extension of an island, so to an item which is not the origin of any other item. Table 8 shows that the average length of partial recognized substrings is higher with our techniques than with this other bottom-up strategy, preserving a running time better than the predictive CKY. This result means that our algorithm has checked more possible attachments and has delivered more extended partial results.

algorithm	average time /utter. (ms)	average island extension
predictive CKY	87	2.55
connection driven	58	2.79

Table 8: Comparison between bottom-up parsers for partial parses

The difference between these two running times can be explained by the consideration of connected routes which results in less items to tabulate and by the initialization of foot node. In a predictive CKY-like algorithm for LTAG, the foot nodes are initialized each time that an internal node (with the same category label) is recognized independently to other positions in the auxiliary trees they belong to. This initialization is more deterministic in our algorithm since a foot node is initialized when a left or right extension has reached this node.

Conclusion

We have presented a new tabular algorithm dedicated to LTG which really takes into account the fact that we manipulate partial parsing trees and not CF rules. Lexicalized partial parsing trees are richer structures than CF rules and allow to obtain more extended partial results which are relevant for natural language robust parsing. A complementary compaction of the LTG allows the factorization of sub-structures parsing.

This algorithm tries to compromise a chunk parsing and the complete parsing of an utterance, it satisfies the following properties at any moment:

- The maximal extension of islands according to adjacency and local constraints located on the left and right connected route of this island.
- The global correction of each partial results.

The theoretical worst case complexity is in $O(N^2G^2n^6)$, but we argue that in practice, when dealing with natural language, decreasing the average complexity is more important than considering a worst case complexity which conditions almost never happen.

When no connected parse can span the whole sentence, our algorithm results not only in more extended partial derivation trees but also consists in representations of islands and both their right and left connected routes. We can then exploit these results for parsing repairs: The adjacency of two islands which can not be combined according to a given LTAG often localizes an agrammaticality. The corresponding chart items can trigger additional repairing rules and flexible controls in a two-step strategy as shown in [Lopez, 1999b].

References

- [Abeillé et al., 1999] Abeillé, A., Candito, M.-H., and Kinyon, A. (1999). FTAG: current status and parsing scheme. In *VEXTAL, Venice, Italy*.
- [Candito and Kahane, 1998] Candito, M.-H. and Kahane, S. (1998). Defining DTG derivations to get semantic graphs. In *Fourth International Workshop on TAG+ (TAG+4), Philadelphia*.
- [Evans and Weir, 1997] Evans, R. and Weir, D. (1997). Automaton-based Parsing for Lexicalized Grammars. In *Fifth International Workshop on Parsing Technologies*, Cambridge, Mass.
- [Evans and Weir, 1998] Evans, R. and Weir, D. (1998). A structure-sharing parser for lexicalized grammars. In *COLING-ALC*, Montréal, Canada.
- [Joshi and Schabes, 1992] Joshi, A. K. and Schabes, Y. (1992). Tree Adjoining Grammars and lexicalized grammars. In Nivat, M. and Podelski, A., editors, *Tree automata and languages*. Elsevier Science.
- [Lang, 1991] Lang, B. (1991). Towards a Uniform Formal Framework for Parsing. In Tomita, M., editor, *Current Issues in Parsing Technology*. Kluwer Academic Publishers.
- [Lavelli and Satta, 1991] Lavelli, A. and Satta, G. (1991). Bidirectional parsing of lexicalized tree adjoining grammars. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Berlin, Germany.
- [Lopez, 1999a] Lopez, P. (1999a). *Parsing of spoken language for man-machine dialogue with Lexicalized Tree Grammars (in French)*. PhD thesis, Université Henry Poincaré Nancy 1.
- [Lopez, 1999b] Lopez, P. (1999b). Repairing strategies for Lexicalized Tree Grammars. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Bergen, Norway.
- [Magerman and Weir, 1992] Magerman, D. M. and Weir, C. (1992). Efficiency, Robustness, and Accuracy in Picky Chart Parsing. In *ACL'92*.
- [Nederhof, 1997] Nederhof, M.-J. (1997). Solving the correct-prefix property for ltags. In *Fifth Meeting on Mathematics of Language (MOL5), Schloss Dagstuhl, Germany*.
- [Nederhof, 1998] Nederhof, M.-J. (1998). An alternative LR algorithm for TAGs. In *COLING'98*, Montréal, Quebec.

- [Satta and Stock, 1989] Satta, G. and Stock, O. (1989). Formal properties and implementation of bidirectional charts. In *Eleventh International Joint Conference on Artificial Intelligence (IJCAI)*, Detroit, MI, pages 1480–1485.
- [Schabes, 1994] Schabes, Y. (1994). Left to Right Parsing of Lexicalized Tree Adjoining Grammars. *Computational Intelligence*, 10:506–524.
- [Schabes and Waters, 1995] Schabes, Y. and Waters, R. C. (1995). Tree insertion Grammar: A Cubic-Time, Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Trees Produced. *Computational Intelligence*, 21:479–514.
- [Shieber et al., 1995] Shieber, S., Schabes, Y., and Pereira, F. (1995). Principles and Implementation of Deductive Parsing. *Journal of Logic Programming*, 24:3–36.
- [van Noord, 1994] van Noord, G. (1994). Head Corner Parsing for TAG. *Computational Intelligence*, 10:525–534.
- [Vijay-Shanker, 1987] Vijay-Shanker, K. (1987). *A Study of Tree Adjoining Grammar*. PhD thesis, University of Pennsylvania, Philadelphia.
- [Vijay-Shanker and Weir, 1993] Vijay-Shanker, K. and Weir, D. J. (1993). Parsing some constrained grammar formalisms. *Computational Linguistics*, 19:591–636.