Lexical Disambiguation in LTAG using Left Context

Claire Gardent[1], Yannick Parmentier[2], Guy Perrier[3], Sylvain Schmitz[4]

(1) CNRS - LORIA – 615, Rue du Jardin Botanique – F-54500 Vandoeuvre-Lès-Nancy, France
(2) LIFO - Université d'Orléans – 6, Rue Leonard De Vinci – F-45067 Orléans Cedex 2, France
(3) LORIA - Université de Nancy, 615, Rue du Jardin Botanique – F54500 Vandoeuvre-Lès-Nancy, France
(4) LSV – ENS - 61, avenue du Président Wilson - F-94235 Cachan Cedex, France
claire.gardent@loria.fr – yannick.parmentier@univ-orleans.fr – guy.perrier@loria.fr – sylvain.schmitz@lsv.ens-cachan.fr

Abstract

In this paper, we present an automaton-based lexical disambiguation process for Lexicalized Tree-Adjoining Grammar (LTAG). This process builds on previous work of Bonfante *et al.* (2004), and extends it by computing a polarity-based abstraction, which contains information about left context. This extension allows for a faster lexical disambiguation by reducing the filtering automaton.

Keywords: parsing, lexical disambiguation, lexicalized tree adjoining grammar

1. Introduction

When deep-parsing natural language with LTAG, one needs to first retrieve, for each word of the sentence to parse, the grammatical structures (i.e., LTAG trees) this word is associated with in the lexicon. This operation is called lexical selection.[1] Unfortunately, in large lexicalized grammars, a given word may be associated with hundreds of grammatical structures. This high lexical ambiguity causes a combinatorial explosion, which highly impacts parsing times. Indeed, lexical selection may yield an exponential number of sequences of structures (in the worst case, one needs to retrieve the cartesian product of the lexical entries of each word of the sentence to parse). In order to reduce lexical ambiguity, one wants to compute, for a given input sentence, a subset of the grammatical structures, where useless structures with respect to the input sentence are discarded. Bangalore and Joshi (1999) proposed to use n-grams and probabilities to compute the set of most probable grammatical structures. This probability-based lexical selection is called *supertagging*. A major drawback of this approach is that it heavily relies on the training corpus used for assigning lexical probabilities. Supertagging may ignore valid structures (i.e., structures that are actually needed to parse the input sentence), which in turn can degrade parsing accuracy.

To prevent lexical selection from ignoring valid structures, Boullier (2003) proposed to rather define a grammatical abstraction (i.e., a morphism), which produces an abstract grammar from the input one, and to use this abstract grammar to parse the input sentence. For each set of abstract structures that succeed in parsing the input sentence, one selects the original structures associated with these by the morphism. This technique improves parsing efficiency only if parsing with the abstract grammar is less complex than with the input grammar. In his experiments, Boullier (2003) abstracted a Context-Free Grammar from an LTAG. Parsing with LTAG has a polynomial complexity of $O(n^6)$, n being the length of the sentence to parse, while parsing with CFG has a complexity of $O(n^3)$. The main drawback of this approach is that one needs to first parse with the abstract grammar, which may still be time consuming.

Following Boullier (2003), Bonfante *et al.* (2004) proposed a non-probabilistic lexical selection, where one uses a polarity-based abstraction. This abstraction is inspired by Interaction Grammar (Perrier, 2000).[2] Interaction Grammar can be seen as an extension of Categorial Unification Grammar where attribute-value pairs are replaced with attribute-polarity-value triples that also encode valency constraints. Bonfante *et al.* (2004) thus aimed at reducing the combinatorial explosions of a $O(n^6)$ Unification Grammar by preliminary filtering with the valency constraints of Categorial Grammars with $O(n^3)$ complexity. Concretely, they extracted sets of polarities from the input grammatical structures. These sets were then combined according to the sentence to parse, in order to compute neutral sets referring to valid selections (we will elaborate on this in Section 3). In this paper, we build on the work of Bonfante *et al*. (2004), and propose to compute a polarity-based abstraction, which also takes into account the linear order of the words of the sentence to parse to speed up filtering.

This paper is organized as follows. In Section 2, we briefly introduce Lexicalized Tree-Adjoining Grammar. Then, in Section 3, we give a detailed presentation of the polarity-based lexical selection of Bonfante *et al.* (2004) and show how we extend it to reduce the cost of lexical selection. In Section 4, we present an implementation of this extended lexical selection within an LTAG parser, and an evaluation of this approach. Finally, in Section 5, we give some perspectives about future work.

2. Lexicalized Tree-Adjoining Grammar

2.1 Definition

A Lexicalized Tree-Adjoining Grammar (Joshi and Schabes, 1997) is a tree rewriting system, where

[1] Lexical selection is related to part-of-speech tagging. While the latter assigns to a word its possible syntactic categories in a given context, lexical selection assigns to a word its possible grammatical structures.

[2] In this formalism, grammatical structures are tree descriptions, where nodes are labeled with polarized feature-structures and parsing corresponds to computing syntactic tree models where polarities are neutralized.

elementary trees can be rewritten using one of the two following operations: *substitution* and *adjunction*.

Substitution consists in replacing a leaf node (marked with a ↓) of an elementary or derived tree with an elementary tree whose root is labeled with the same syntactic category as this leaf node. Adjunction consists in replacing an internal node of an elementary or derived tree with an elementary tree having both a root node and a leaf node (marked with *, called foot node) labeled with the same category as this internal node. As an illustration, see Fig. 1 below. It shows the substitution (resp. adjunction) of the tree associated with *John* (resp. *deeply*) into the tree associated with *sleeps*.

Furthermore, in a Lexicalized TAG, every elementary tree is associated with at least one lexical item (called *anchor*). If a tree is associated with several lexical items, there is one anchor, and several co-anchors.

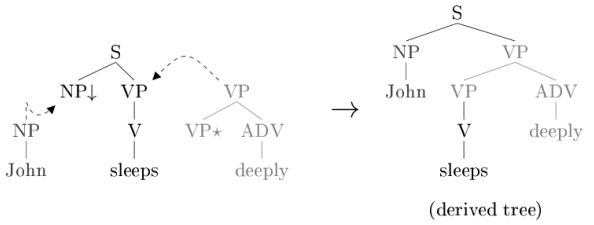

Fig. 1: Tree rewriting in LTAG

2.2. Parsing with LTAG

A large LTAG is made of thousands of elementary trees. To reduce redundancy within this huge set of trees, a three-layer lexicon architecture has been proposed by the XTAG Group (2001). It consists in dividing the LTAG lexicon in three sub-lexicons:

- a morphological lexicon associating inflected words with lemmas,
- a syntactic lexicon associating lemmas with tree schemas (i.e., unanchored LTAG trees),
- and a grammatical lexicon containing tree schemas to be anchored. These tree schemas are gathered into *families* according to the sub-categorization frame they describe.

When parsing with LTAG, (i) the input sentence is first tokenized, then (ii) each token is associated with the corresponding entries from the morphological lexicon. For each selected morphological entry, the syntactic lexicon is used to retrieve corresponding lemmas. For each pair (word, lemma), corresponding tree schemas are selected and anchored. (iii) In a final step, parsing is done using the selected sub-grammar.

As mentioned above, step (ii) may yield an exponential number of sequences of anchored grammatical structures, while step (iii), i.e., LTAG parsing, has a polynomial complexity in the length of the input sentence. The global complexity of (ii)+(iii) can be evaluated as $O(|G|.A^6)$ where A is the number of transitions of the automaton that describes the sequences of useful anchored grammatical structures (Lang, 1994).

In our work, we aim at reducing the practical cost of parsing, by discarding as many useless grammatical structures as possible during lexical selection (step (ii)). The next section shows how this can be done using a polarity-based grammar abstraction.

3. Lexical disambiguation

Following Bonfante *et al.* (2004 ; 2006), our approach for lexical disambiguation uses a polarity-based abstraction over the input grammar. Let us first see how this abstraction is built by Bonfante *et al.* (2004) and applied to LTAG. We shall then show how it can be extended to improve disambiguation times.

3.1 Bonfante *et al.* (2004)'s lexical disambiguation

Bonfante *et al.* (2004) define a general layout for lexical disambiguation, that fits a large number of lexicalized grammar formalisms. They apply this layout to Lambek Grammar (Lambek, 1958) and LTAG. Their lexical disambiguation is a three-step process:

1. (grammar polarization) First, the input grammar is polarized. Every grammatical structure g is associated with a set of polarities S_g. This set is made of polarities of the form (f,v,p) where f is a feature (constant), v a value (constant), and p a (possibly negative) integer. Since the grammar is lexicalized, we can associate each word w of the lexicon with its corresponding multiset of polarities M (recall that a word can anchor several grammatical structures, each associated with a set of polarities, hence the multiset).

2. (abstraction) Second, an abstract lexical selection is operated using the previously computed polarized grammar. Concretely, this abstract lexical selection amounts to first computing the cartesian product P of the multisets associated with the words $w_1 \ldots w_n$ of the input sentence : $P = M_1 \times \ldots \times M_n$. Then, a binary rewriting rule (called neutralization rule) is applied on the elements $E = (S_1, \ldots, S_n)$ of this product P as follows.
   Let (S, S') be a couple of polarity sets in E, these are replaced with a set S'' such that:
   ▪ if a feature f with value v is present in both S and S' as (f,v,p) and (f,v,p') respectively, then the polarity $(f,v,p + p')$ is added to S''.
   ▪ any polarity (f,v,p) in $(S \cup S') \setminus (S \cap S')$ is copied into S''.
   This rewriting goes on (in an arbitrary order) until all elements of E have been consumed, thus producing one polarity set per element E.

3. (filtering) In the end, the elements E of P are filtered to keep only those whose abstraction after neutralization is made of exactly one polarity $(cat, axiom, +1)$ and an arbitrary number of polarities of the form $(f,v,0)$, and nothing else. For each such element, the associated grammatical structures in the input grammar are lexically selected.

Note that this general layout relies on the computation of a huge number of sequences of structures (i.e., the elements of the product P). As pointed out by Perrier (2008), among possible implementations of this lexical selection, using automata helps in sharing information (i.e., sequences with identical sets of polarities).

Also, a crucial point of this approach lies in the definition of polarization. Polarization should include enough information (i.e., enough features) to distinguish between useful grammatical structures in the context of the sentence to parse, and useless ones, and yet it should not lead to a complex, time-consuming abstraction (i.e., applying neutralization to polarized structures should stay computable in reasonable times).

3.2 Application to LTAG

Let us now see how this lexical selection is applied to LTAG, where grammatical structures are trees.

1. The first step (polarization) consists in associating each tree *t* of the input LTAG with a set of polarities of the form *(cat, x, n),* where :
   - *x* is either the category of the root, then *n* is 1 minus the number of substitution or foot nodes labeled with the category *x*, or
   - *x* is not the category of the root, and then *n* is -1 times the number of substitution or foot nodes labeled with the category *x*, or
   - x is a lexical item (i.e., a word) labeling a co-anchor (e.g., a preposition), and then *n* is -1 times the number of such co-anchors.[3]

That is, we consider only one type of feature,namely syntactic category. With this definition, the LTAG trees from Fig. 1 are polarized as:

$polarities(t_{John}) = \{(cat, NP, +1)\}$
$polarities(t_{sleeps}) = \{(cat, S, +1) ; (cat, NP, -1)\}$
$polarities(t_{deeply}) = \{(cat, VP, 0)\}$

2. The second step (abstraction) consists in computing the cartesian product of the set of polarities according to the sentence to parse:[4]

$$P = polarities(t_{John}) \times polarities(t_{sleeps}) \times polarities(t_{deeply})$$

Neutralization consists in summing, for each element of *P*, the polarities for compatible *(f,v)* pairs. In our toy example, we obtain:

neutralization(P)={(cat, S, +1);(cat, NP, 0);(cat, VP, 0)}

3. The third step (filtering) keeps among neutralized elements of *P*, those which are well-formed. In our case, there is only one element in *P* and its neutralization is well-formed for an axiom S (i.e., if we are parsing a sentence).

Note that lexical selection based on polarization and abstraction as presented here does not rely on any particular word order. One can compute well-formed elements of *P* (i.e., apply neutralization) following or not the word order defined by the input sentence.

Also, as mentioned above, using deterministic automata to implement this lexical selection allows to factorize information, which is very important to deal with the huge number of sequences of structures. The idea of this automaton-based implementation is the following. States are associated with sets of polarities. We create an empty initial state. We create as many edges from the initial state as candidate polarized grammatical structures for the first word of the input sentence. These edges lead to new states whose set of polarities is the sum of the polarities of the original state and those of the structure labeling the edge. We merge states with similar sets of polarities. We go on with the second word, that is, we create as many edges from these new states as candidate polarized structures for this word, leading to new states, and so on. As an illustration, consider Fig. 2 below. In this figure, one can see the lexical selection for the sentence "John eats a cake". For the word *John*, there is only one associated grammatical structure in the lexicon, namely that for proper nouns. It is polarized as *(cat, NP, +1)*. For *eats*, there is an ambiguity between intransitive and transitive verbs. There are thus two new states, depending on which tree is used. When the sentence is fully processed, we obtain two final states, among which only one is well-formed (state number 7).

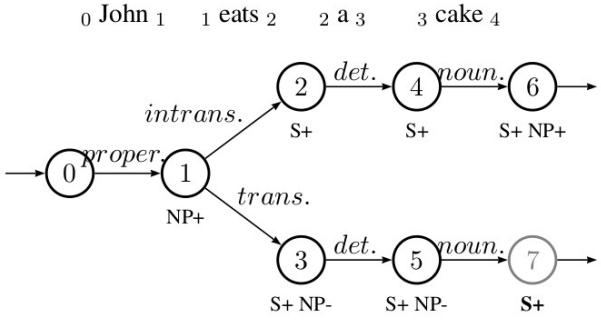

Fig. 2: Automaton-based lexical selection

3.3 Lexical disambiguation using word order

As mentioned above, the polarity-based lexical selection introduced in Section 3.1 does not rely on a specific order. In particular, polarization and abstraction do not use any information about the word order within the input sentence. In order to enhance lexical selection, we propose to use such information by extending Bonfante *et al.* (2004)'s lexical selection as follows.

The polarization of input grammar (step 1) is extended to produce pairs of sets of polarities. That is, polarization generates for each input grammatical structure, a pair of sets of polarities, where the first element is the classical polarization of Bonfante *et al.* (2004), and the second element a polarization related to left context. This second polarization is similar to that presented in Section 3.2 except that it only considers the root node and substitution nodes that are on the left of the anchor. For instance, the tree *sleeps* in Fig. 1 is now polarized as:

$polarities(t_{sleeps})$=
({(cat,S,+1);(cat, NP, -1)},{(cat, S, +1);(cat,NP,-1)})

[3] In step 2, input words which can be coanchors in the lexicon, are associated with trees made of a single root node. These trees bring the +1 polarity needed to neutralize the coanchor in a valid lexical selection.

[4] Note that in our toy example, there is only one tree per word of the input sentence (i.e., there is no lexical ambiguity), and thus *P* has a single element. The purpose here is merely to illustate the lexical selection process.

Abstraction (step 2) is then applied to these pairs of sets of polarities. The definition of the neutralization rule extends that of Bonfante *et al.* (2004), by applying on pairs of sets, while following the word order of the input sentence. Let (a_1, b_1) and (a_2,b_2) be two such pairs:

$$\text{neutralization}((a_1,b_1), (a_2,b_2)) = (\text{neutralization}(a_1,a_2), \text{neutralization}(b_1,b_2))$$

Furthermore, b_1 and *neutralization*(b_1,b_2) are constrained to only contain polarities of the form (f,v,i) where $i \geq 0$.

Finally, filtering (step 3) remains identical to that of Bonfante *et al.* (2004), and applies on the first element of the neutralized pairs.

Let us now see how this extension impacts the automaton-based implementation of lexical selection. Consider the sentence "say it to her", and a lexicon that associates the verb *say* with an imperative, a transitive and a ditransitive grammatical structure. We can build an automaton for lexical selection as previously (see Fig. 3).

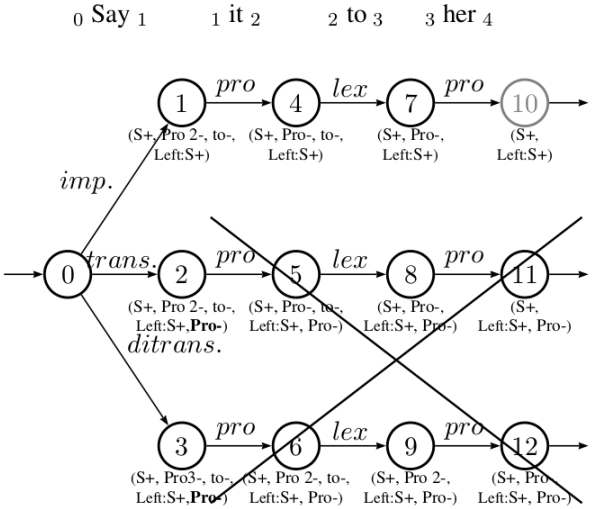

Fig. 3: Lexical selection using left context

We notice that only the imperative lexical entry for *say* prevents the left context abstraction from having negative polarities (i.e., transitive and ditransitive grammatical structures have "uncombinable" substitution nodes on the left of the anchor). Once an abstraction has a non-empty set of left polarities, there cannot be any other abstraction that could combine to produce a well-formed polarity set. In Fig. 3, we can thus discard states 2 and 3, and we do not need to build states 5, 6, 8, 9, 11 and 12.

4. Implementation and evaluation

In order to evaluate the benefits brought by left context information, we implemented polarity-based lexical selection using left context within the TuLiPA parser (Parmentier *et al.*, 2008).[5] TuLiPA is an open source parser for mildly context-sensitive formalisms which supports LTAG and already includes the automaton-based lexical selection of Bonfante *et al.* (2004). This will make it easier to compare the two approaches. To exhibit the benefits of left-context disambiguation, we used two types of resources, a toy LTAG and a large one. These are described below. The question of which metrics to use to evaluate the gain of polarity filtering is not trivial. We chose to compare the sizes of automata when using left context or not. Another interesting metrics would be to measure parsing times using the lexical disambiguation or not, even if the additional cost of automaton-based filtering is expected to be small compared with the cost of parsing with highly ambiguous grammars. Due to time and space restrictions, this is left for future work.

4.1 Qualitative study

In a first experiment, we performed polarity-based lexical selection using a toy LTAG made of 12 tree schemas (intransitive verb with clitic / canonical/ extracted subject, active transitive verb with clitic / canonical / extracted subject, passive transitive verb with clitic / canonical / extracted subject, proper nouns, clitics and auxiliaries), 9 lemmas, and 30 morphological entries (note that there is no part-of-speech tagging ambiguity in our toy lexicon, i.e., lexical ambiguity here is purely grammatical). The results are given in Table 1.

Sentence	# lex	#pol	#st.1	#st.2
Jean aime Marie *(John loves Mary)*	12	2	26	4
Marie dort *(Mary sleeps)*	3	1	5	3
Ils dorment *(They sleep)*	3	1	5	3
Jean qui dort aime Marie *(John who sleeps loves Mary)*	36	8	78	7
Marie est appelée par Jean *(Mary is called by John)*	6	1	21	9
Jean qui dort est aimé par Marie *(John who sleeps is loved by Mary)*	18	4	63	13

Table 1. Polarity-based lexical disambiguation

In Table 1, the column labelled #lex gives the general lexical ambiguity of the sentence (i.e., cardinality of the cartesian product of the lexical entries selected by each word). The column #pol gives the remaining lexical ambiguity once polarity filtering has been performed (on this toy example, the remaining ambiguity is the same using left-context optimization or not). The column #st.1 gives the size (i.e., number of states) of the polarity-based automaton of Bonfante *et al.* (2004), and #st.2 the size of the automaton that uses left context. One can notice that using left context significantly reduce the size of the polarity-based automaton, even with a small grammar and short sentences.

4.2 Quantitative study

In a second experiment, we used the French LTAG of Crabbé (2005). This grammar is compiled from a metagrammar developed in the XMG language (Duchier *et al.*, 2004).[6] It is made of 6,080 tree schemas, and

[5] See http://sourcesup.cru.fr/tulipa.

[6] Crabbé's metagrammar is available at https://sourcesup.cru.fr/scm/viewvc.php/trunk/METAGRAMMARS/FrenchTAG/?root=xmg and its French documentation at http://www.linguist.univ-paris-diderot.fr/~bcrabbe/frenchgrammar/index.html

focusses on verbal, adjectival and nominal predicatives. We used this grammar to evaluate polarity-based lexical disambiguation on a subset of the *Test Suite for Natural Language Processing* (Lehmann *et al.*, 1996). This subset contains 90 sentences whose length ranges from 2 to 12 words. Fig. 4 displays the average number of states of the automaton (y-axis) depending on the length of the input sentence (x-axis). One can notice the combinatorial explosion of the size of the automaton of Bonfante *et al.* (2004) with sentences having more than 7 words.

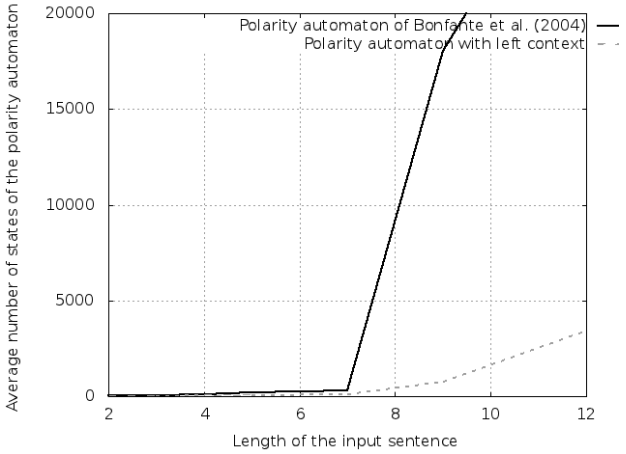

Fig. 4: Distribution of the size of polarity automata

As suggested by Fig. 4, for polarity filtering to stay computationally tractable when parsing real data, optimizations are needed. From these first experiments, using information about left context seems to be adequate in practice. This still needs to be confirmed with other evaluation resources and metrics.

5. Conclusion

In this paper, we presented an extension of the technique of Bonfante *et al.* (2004) for lexical disambiguation. This extension consists in defining a grammar abstraction containing information about left context. In LTAG, this information corresponds to substitution sites located on the left of the anchor. This abstraction makes it possible to reduce *on-line* the size of the automaton used for lexical selection, thus improving parsing time efficiency.

As advocated by Bonfante *et al.* (2009), using information about context (e.g., dependency constraints) for improving lexical disambiguation may be applied to several types of lexicalized grammar through the concept of companionship. Similarly, it would be interesting to apply left context abstraction to other formalisms.

Another interesting question concerns the definition of polarization. How to polarize a grammar so that invalid grammatical structures can be filtered out as soon as possible while keeping neutralization computationally tractable ? For LTAG, we chose to use only polarities related to syntactic categories. For Interaction Grammar, other types of features are used (e.g., morpho-syntactic features), leading to a complex abstraction because of neutralization cost. There is a trade-off to be found.

Acknowledgement

The implementation of the left-context automaton has been inspired by Johannes Dellert's code within TuLiPA.

References

Bangalore, S., and Joshi, A. (1999). Supertagging: An Approach to Almost Parsing. In: *Computational Linguistics*, 25(2). Pp. 237-265. MIT Press.

Bonfante, G., Guillaume, B., and Morey, M. (2009) *Dependency Constraints for Lexical Disambiguation. 11Th Int. Conference on Parsing Technologies, IWPT 2009, Paris, France*. Pp. 242-253.

Bonfante, G., Guillaume, B., and Perrier, G. (2004). *Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation. 20Th Int. Conference on Computational Linguistics, COLING 2004, Geneva, Switzerland.* Pp. 303-309.

Bonfante, G., Le Roux, J., and Perrier, G. (2006). *Lexical disambiguation with polarities and automata. 11Th Int. Conference on Implementation and Application of Automata, CIAA 2006, Taïpei, Taïwan.* Pp. 281-282.

Boullier, P. (2003). *Supertagging: a Non-Statistical Parsing-Based Approach. 8Th Int. Workshop on Parsing Technologies, IWPT 2003, Nancy, France.* Pp. 55-66.

Crabbé, B. (2005). Représentation informatique de grammaires fortement lexicalisées : application à la grammaire d'arbres adjoints. *PhD Thesis. Université Nancy 2, France.*

Duchier, D., Le Roux, J., and Parmentier, Y. (2004). *The Metagrammar Compiler: an NLP Application with a Multi-Paradigm Architecture. 2Nd Oz-Mozart Int. Conference, MOZ 2004, Charleroi, Belgium.* Pp 175-187. Lecture Notes in Computer Science, Vol. 3389, Springer Verlag.

Joshi, A., and Schabes, Y. (1997). Tree-Adjoining Grammar. In: Handbook of Formal Languages. In: Rozenberg G. and Saloma A. (Eds.) Springer Verlag. Vol. 3. Pp. 69-123.

Lambek, J. (1958). The Mathematics of Sentence Structure. *American Mathematical Monthly*, 65(3). Pp. 154-170.

Lang, B. (1994). Recognition can be harder than parsing. In: *Computational Intelligence, vol. 10.* Pp. 486-494.

Lehmann, S., Oepen, S., Regnier-Prost, S., Netter, K., Lux, V., Klein, J., Falkeda, K., Fouvry, F., Estival, D., Dauphin, E., Compagnion, H., Baur, J., Balkan, L., and Arnold, D. (1996). TSNLP — Test Suites for Natural Language Processing. *16Th Int. Conference on Computational Linguistics, COLING 1996, Copenhagen, Denmark.* Pp. 711-716.

Parmentier, Y., Kallmeyer, K., Lichte, T., Maier, W., and Dellert, J. (2008). A Syntax-Semantics Parsing Environment for Mildly Context-Sensitive Formalisms. *9Th Int. Workshop on Tree-Adjoining Grammar and Related Formalisms, TAG+9. Tübingen, Germany.* Pp. 121-128.

Perrier, G. (2000). *Interaction Grammars. 18Th Int. Conference on Computational Linguistics, COLING 2000, Saarbrücken, Germany.* Pp. 600-606.

Perrier, G. (2008). Désambiguisation lexicale à l'aide d'automates de polarités. Rapport de recherche INRIA. http://hal.inria.fr/docs/00/27/84/43/PDF/automates2008.pdf Access date: August, 26, 2011. 11 pages.

XTAG Research Group (2001). A Lexicalized Tree-Adjoining Grammar for English. Technical report - IRCS, University of Pennsylvania. 341 pages.