Lexical Disambiguation in LTAG using Left Context

Claire Gardent, yannick Parmentier, Guy Perrier, Sylvain Schmitz

Lexical Disambiguation in LTAG using Left Context

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

[1] CNRS, LORIA, Nancy, France
[2] Université d'Orléans, ENSI de Bourges, LIFO, Orléans, France
[3] Université de Lorraine, LORIA, Nancy, France
[4] ENS Cachan, LSV, Cachan, France

Abstract. In this paper, we present an optimization for parsing with Lexicalized Tree-Adjoining Grammar which takes inspiration from previous work on polarity based grammar abstraction (Bonfante et al., 2004). We illustrate the impact of this optimization on two benchmarks and we relate our approach to the more general optimization framework proposed for Interaction Grammars by (Bonfante et al. (2009) and Morey (2011).

Keywords: parsing, lexical disambiguation, lexicalized tree adjoining grammar, supertagging

1 Introduction

When parsing with a lexicalized grammatical formalism such as *lexicalized tree adjoining grammars* (LTAG), a first step (called *lexical selection*) consists in retrieving, for each word of the sentence to parse, the associated grammatical structures (here the elementary LTAG trees).

As shown by Bangalore and Joshi [1], in large lexicalized grammars, lexical selection usually yields an intractable search space. This is because each given word may be associated with hundreds of grammatical structures resulting in an exponential number of sequences of structures to be explored (namely, the cartesian product of the sets of lexical entries of each word of the sentence to parse). In practice however, very few of these sequences can yield a complete parse.

In this paper, we adapt an optimization technique proposed by Bonfante et al. [3] for interaction grammars [13] to prune the initial search space to LTAG and we illustrate the impact of this optimization on two benchmarks. We also relate our approach to a more general optimization technique proposed by Bonfante et al. [2] and Morey [11].

The paper is structured as follows. In Section 2, we summarize related work. Section 3 gives a brief introduction to LTAG. In Section 4 and 5, we present the optimization proposed and we illustrate its impact on two small benchmarks. In Section 6, we relate our approach to the more general approach of *companions* introduced in [2, 11]. Section 7 concludes.

2 Related Work

Drawing inspiration from part-of-speech tagging techniques, Bangalore and Joshi [1] use n-grams and probabilities to compute the set of most probable grammatical structures given an LTAG and an input string. 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 thus ignore valid structures (i.e. structures that are in fact needed to parse the input sentence), which in turn can degrade parsing accuracy.

To prevent lexical selection from ignoring valid structures, Boullier [5] proposed to compute 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 then selects the corresponding original structures and parse with the original grammar. This technique improves parsing efficiency only if parsing with the abstract grammar is significantly less complex than parsing with the input grammar. In his experiments, Boullier abstracted a context-free grammar (CFG) from an LTAG: the most common parsing algorithms for LTAGs have a polynomial time complexity in $O(n^6)$, n being the length of the input sentence, while those for CFGs have a complexity in $O(n^3)$. The main drawback of this approach is that one needs to first parse with the abstract grammar, which may still be quite time-consuming.

Following Boullier, Bonfante et al. [3] proposed a non-probabilistic lexical selection algorithm which uses a *polarity-based abstraction*, an abstraction inspired by interaction grammar [13]. 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. Bonfante et al. thus aimed at reducing the initial search space by applying a preliminary filter based on the polarity constraints. In their approach, each input grammatical structure is associated with a set of polarities. Valid lexical selection sequences are then those sequences whose total polarity set is neutral (we will elaborate on this in Section 4).

Perrier [14] proposes a finer abstraction in which the position of the polarities with respect to the anchor of the elementary structures of the grammar is taken into account. On this basis, he proposes a disambiguation algorithm which takes into account the linear order of the words in the sentence

As discussed in Section 6, our work takes inspiration from [3, 14] and proposes an algorithm that is similar to that proposed in [14] though less costly in terms of space.

3 Lexicalized Tree-Adjoining Grammars

A tree-adjoining grammar [8] is a tree rewriting system, where elementary trees can be rewritten using one of the two following operations: *substitution* and *adjunction*. Substitution consists in replacing a leaf node (marked with '↓') of an elementary or derived tree with an elementary tree whose root is labeled with

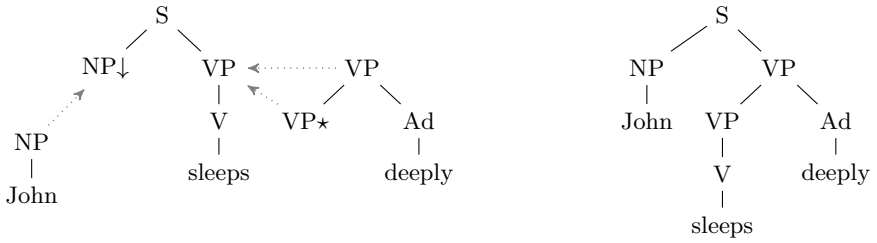

Fig. 1: Tree rewriting in an LTAG.

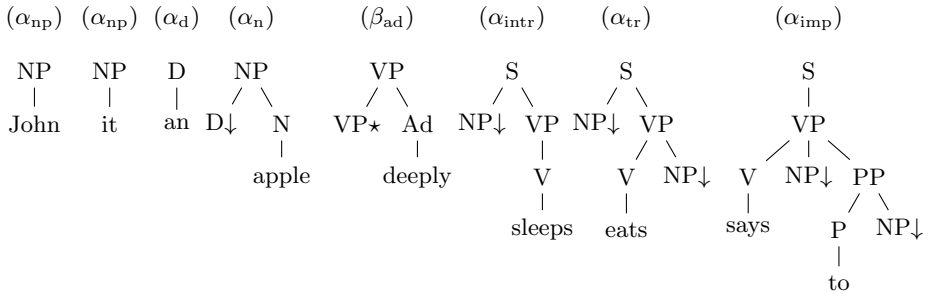

Fig. 2: A toy LTAG for English.

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 distinguished leaf node (marked with '⋆', and called the *foot* node) labeled with the same category as this internal node.

Furthermore, in a *lexicalized* TAG, every elementary tree is associated with at least one lexical item called its *anchor*. If a tree is associated with several lexical items, there is one anchor, and several *co-anchors*. Figure 1 shows on the left the substitution (resp. adjunction) of a tree anchored by 'John' (resp. 'deeply') into a tree anchored by 'sleeps', the two operations resulting in the *derived* tree displayed on the right. Figure 2 shows a toy LTAG that will be used throughout to illustrate the workings of the algorithms.

4 Lexical Disambiguation

Following Bonfante et al. [3, 4], our approach for lexical disambiguation uses a polarity-based abstraction over the input grammar. We first explain how this abstraction is built by Bonfante et al. [3]. We then show how it can be applied to LTAG. Finally we show how this first filtering algorithm can be extended to improve disambiguation performance by taking word order information into account.

4.1 Polarity-Based Lexical Disambiguation

Bonfante et al. [3] define a framework for lexical disambiguation which involves the following three-steps:

Polarization: First, the input grammar is polarized. That is, each grammatical structure g is associated with a set of polarities S_g where a *polarity* is a tuple (f, v, p) with f is a feature, v a value, and p a (possibly negative) integer. The sets of possible features and feature values are assumed to be finite. 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 possibly non-unique set of polarities).

Neutralization: Second, an abstract lexical selection is operated using the previously computed polarized grammar. This amounts to first computing the cartesian product P of the multisets associated with the words $w_1 \cdots w_n$ of the input sentence: $P = M_1 \times \cdots \times M_n$. Then, a binary rewriting rule, called the *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 $S + S'$ 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 $S + S'$.
- any other polarity (f, v, p) in $S \cup S'$ is copied into $S + S'$.

This rewriting goes on (in an arbitrary order) until all elements of E have been consumed, thus producing one polarity set per element E in P.

Filtering: In the end, an element E of P is called *well-formed* if, after neutralization, it consists of exactly one polarity $(cat, S, +1)$ (where S is the category of sentences), an arbitrary number of polarities of the form $(f, v, 0)$ and nothing else. Filtering only keeps the well-formed elements in P. For each such element, the associated grammatical structures in the input grammar are lexically selected.

A crucial point of this approach lies in the definition of polarization. It ought to include enough information to distinguish between useful grammatical structures in the context of the sentence to parse, and useless ones. Yet it should not lead to a complex, time-consuming abstraction: applying neutralization to polarized structures should remain fast.

4.2 Application to LTAG

We now show how this lexical disambiguation process can be applied to LTAG.

The polarization step consists in associating each tree t of the input LTAG with a set of polarities. For LTAG, we define our polarities to be of the form (cat, x, p), where x is a syntactic category labeling a tree node and $p = 1 - n$ with n the number of substitution or foot nodes labeled with x if x is the category of the root and $p = n$ otherwise.

For instance, the tree schemas used in Fig. 1 are polarized as follows:

$$\begin{aligned} \mathit{polarities}(\alpha_{\mathrm{np}}) &= \{(\mathit{cat}, \mathrm{NP}, +1)\} \ , \\ \mathit{polarities}(\alpha_{\mathrm{intr}}) &= \{(\mathit{cat}, \mathrm{S}, +1),\ (\mathit{cat}, \mathrm{NP}, -1)\} \ , \\ \mathit{polarities}(\beta_{\mathrm{ad}}) &= \{(\mathit{cat}, \mathrm{VP}, 0)\} \ . \end{aligned}$$

To take co-anchors into account, we change co-anchors in the grammar to substitution leaves and we add corresponding single node trees to match these new requirements.

The neutralization step first constructs the cartesian product of the set of polarities according to the sentence to parse. In the case of Fig. 1 ('John sleeps deeply'), this set contains a single element:

$$P = \big\{\{(\mathit{cat}, \mathrm{NP}, +1)\}, \{(\mathit{cat}, \mathrm{S}, +1),\ (\mathit{cat}, \mathrm{NP}, -1)\}, \{(\mathit{cat}, \mathrm{VP}, 0)\}\big\} \ .$$

Next, neutralization sums the polarities for compatible (f, v) pairs for each element E of P. In our toy example, neutralization yields the following set of polarities for the single element of P:

$$\mathit{neutralization}(E) = \{(\mathit{cat}, \mathrm{S}, +1), (\mathit{cat}, \mathrm{NP}, 0), (\mathit{cat}, \mathrm{VP}, 0)\} \ .$$

The filtering step keeps the well-formed elements of P. In our case, there is only one element in P and it is well-formed.

Note that lexical selection based on polarization as presented here does not rely on any particular word order. One can compute the well-formed elements of P following or not the word order defined by the input sentence.

4.3 Automata Representation

Using an automaton to represent the results of lexical selection is necessary in order to deal with the large number of resulting sequences of structures, and can be seamlessly integrated in the parsing process with the same $O(|G| \cdot n^6)$ time complexity, now with n being the number of states [9]. Hence we assume lexical selection to yield an automaton, which can later be processed by the parsing algorithm. The various lexical disambiguation algorithms strive to eliminate spurious paths from this automaton, i.e. combinations of grammatical structures that cannot yield a valid parse. In the case of polarity filtering, this *polarity automaton* tags each state with the sum of the polarities associated with the structures labeling the path from the initial state to that state. For instance, the sentence 'John eats an apple' yields the polarized automaton shown in Fig. 3 (where we only display the non-zero polarities).

Thus the final states of the polarity automaton indicate the (common) neutralization result for all the paths that lead into them, and the filtering step can be implemented by only keeping the paths that lead to a well-formed final state. For instance in Fig. 3, the top branch (in red) is discarded by the filtering step because state 4 is not a well-formed final state.

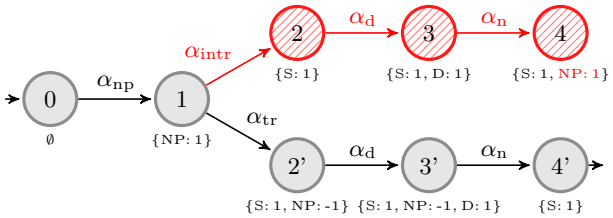

Fig. 3: A polarity automaton for the sentence 'John eats an apple.'

4.4 Lexical Disambiguation Using Word Order

As mentioned earlier, the polarity-based lexical selection introduced in Section 4.1 does not take word order into account. In order to enhance lexical disambiguation, we propose to use *left context* information by extending the algorithm presented in Section 4.2 as follows.

The polarization of the input grammar is extended to produce *pairs* of sets of polarities. The first element in this pair is the classical polarity set, and the second element is a polarization related to the left context, that only takes (a) the root node and (b) the substitution nodes situated to the left of the anchor into account. For instance, the two trees for the word 'eats' and the imperative tree for 'say' are associated with the following polarity sets:

$$\begin{aligned}
\mathit{polarities}(\alpha_{\mathrm{tr}}) &= \big(\{(cat, \mathrm{S}, +1), (cat, \mathrm{NP}, -2)\}, \{(cat, \mathrm{S}, +1), (cat, \mathrm{NP}, -1)\}\big)\ , \\
\mathit{polarities}(\alpha_{\mathrm{intr}}) &= \big(\{(cat, \mathrm{S}, +1), (cat, \mathrm{NP}, -1)\}, \{(cat, \mathrm{S}, +1), (cat, \mathrm{NP}, -1)\}\big)\ . \\
\mathit{polarities}(\alpha_{\mathrm{imp}}) &= \big(\{(cat, \mathrm{S}, +1), (cat, \mathrm{NP}, -2), (cat, \mathrm{to}, -1)\}, \{(cat, \mathrm{S}, +1)\}\big)
\end{aligned}$$

Neutralization is then applied to the two components of the pair. Let (a_1, b_1) and (a_2, b_2) be two such pairs:

$$(a_1, b_1) + (a_2, b_2) = (a_1 + a_2, b_1 + b_2)\ .$$

Filtering still enforces the result of neutralization to be well-formed on the first coordinate, but additionally constrains the successive neutralization results to only contain non-negative polarities in the second coordinate: during each neutralization step, $b_1 + b_2$ is constrained to only contain polarities of the form (f, v, i) where $i \geq 0$. Intuitively, this constraint excludes sequences where a requirement on the left of the current token has not been fulfilled.

Automata Representation. Let us now see how this extension impacts the automaton-based implementation of lexical selection. Because the automaton construction already relied on the input sentence order, the extension is easily implemented by decorating the states with pairs of polarity sets. The non-negativity constraint on the second component (which accounts for the left context polarities) is checked on-the-fly: as soon as a state has a negative polarity in the second component, the construction of its successors aborts.

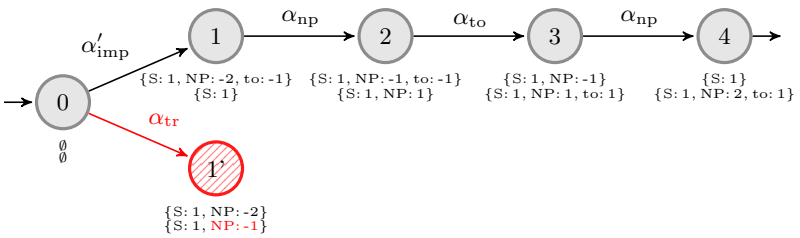

Fig. 4: Lexical selection using left context for 'Say it to John.'

Consider the sentence 'Say it to John' and the grammar of Fig. 2.[5] The resulting automaton is shown in Fig. 4. We notice that only the imperative tree schema α_{imp} for 'say' prevents the left context abstraction from having negative polarities; the construction for the transitive tree schema α_{tr} (e.g., 'John said it') aborts immediately. This is a vast improvement over the basic polarity automaton, which requires building a full automaton before filtering can take place and select well-formed final states.

5 Implementation and Evaluation

In order to evaluate the benefits brought by left context, we have implemented polarity-based lexical selection using left context within the TuLiPA parser [12].[6] 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. [3] presented in Section 4. This makes 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 total parsing times, even if the additional cost of automaton-based filtering is expected to be negligible compared with the cost of parsing with highly ambiguous grammars. Due to time and space restrictions, this is left for future work.

5.1 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

[5] Recall that the grammar is modified so that co-anchors are represented as substitution nodes and additional single node trees are added to match these nodes. This explains why the α_{imp} tree schema carries a *to* polarity feature.

[6] See `http://sourcesup.renater.fr/tulipa`.

nouns, clitics and auxiliaries), 9 lemmas, and 30 morphological entries. 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.

Table 1: Polarity-based lexical disambiguation.

	Sequences		States	
Sentence	Initial	Polarity	Polarity	Context
'Jean aime Marie' (*John loves Mary*)	12	2	26	4
'Marie dort' (*Mary sleeps*)	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

In Table 1, the first two figures provide the lexical ambiguity of the sentence before and after polarity-based filtering. In these toy examples, the remaining lexical ambiguity is the same whether we only use polarities or also consider left context information. The last two figures give the number of states in the automata representations after polarity filtering and polarity and left context filtering, respectively. One can notice that using left context significantly reduces the size of the automaton, even with a small grammar and short sentences.

5.2 Quantitative Study

In a second experiment, we used the French LTAG of Crabbé [6]. This grammar is compiled from a metagrammar developed in the XMG language [7].[7] It contains 6,080 tree schemas, and focuses 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 [10]. This subset contains 90 sentences whose length ranges from 2 to 12 words. Figure 5 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 'Polarity' automata of Bonfante et al. [3] compared to the 'Context' automata, when considering sentences with more than seven words.

As suggested by Fig. 5, for polarity filtering to stay computationally tractable when parsing real data, optimizations are needed. From these first experiments,

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

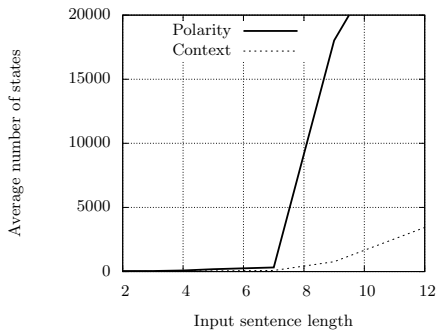

Fig. 5: Distribution of polarity automata sizes on the TSNLP.

using information about left context seems to be adequate in practice. This still needs to be confirmed with other evaluation resources and metrics.

6 Generalization to Companions

We now show that the optimization we proposed for LTAG is a special case of the more general optimization framework developed in [2, 11].

6.1 The Notion of a Companion

In a polarized grammar, each grammatical structure is associated with a polarity and saturation rules define valid polarity combinations. For instance in LTAG, two trees may only combine if their polarity sets contains a positive and a negative triple for the same category e.g., $\{cat, \mathrm{NP}, +1\}$ and $\{cat, \mathrm{NP}, -1\}$). Given a polarized grammar, Bonfante et al. [3] defines *companions* as follows.

Definition 6.1. *Let G be a polarized grammar. Given two grammatical structures G_1 and G_2 in G with polarities p_1 and p_2 respectively, p_2 is a* companion *of p_1 iff G_1 and G_2 can be combined by the grammar rules of G and p_1, p_2 is a valid polarity combination according to the saturation rules defined by G.*

Although the definition was initially designed for interaction grammars, it straightforwardly extends to the polarized version of LTAG described in subsection 4.2. Thus for instance, given the elementary trees of Fig. 1, the polarity $\{cat, \mathrm{NP}, -1\}$ triggered by the substitution node of α_{intr} is a companion of the positive polarity $\{cat, \mathrm{NP}, +1\}$ associated with the root node of α_{np}. More generally, the companions of a polarity in LTAG can be defined as follows. Given an initial tree whose root node has category C, the companions of the polarity of this root node are the polarities of all substitution nodes with category C. To account for word order and encode a filtering exploiting left context information as proposed in subsection 4.4, companions are furthermore divided into left- and

right-companions. A left companion is a polarity associated with a substitution node occurring to the right of the anchor of the tree containing that substitution node. Conversely, a right companion is associated with a substitution node occurring to the left of the anchor of the tree containing that substitution node.

6.2 Using Companions for LTAG Filtering

Morey [11] presents a general framework for filtering the initial search space when parsing with polarized grammars and in particular with interaction grammars. In particular, he defines the ICP algorithm which is based on the *Integer Companionship Principle* (ICP) and applies to systems with *linear polarities*. A polarity system is said to be linear if its non saturated polarities are grouped by pairs such that the only possible combination for a non saturated polarity is with its paired polarity. Thus the system of polarities for LTAG is linear: it has two non saturated polarities + and −. Furthermore, one positive polarity combines with exactly one negative polarity and conversely.

The ICP for interaction grammars is informally as follows:

> *For every path of the automaton, every polarity of a grammatical structure on this path must find a companion on the same path, and every possible companion on the path must be the companion of one polarity at most on the path, otherwise the path can be removed.*

The ICP algorithm starts from some input automaton of lexical selections and builds, from this initial automaton, an automaton which validates the constraints enforced by the ICP. At each step of the automaton construction process, polarity information is updated and a constraint (called RC or *right-companion constraint*) is applied to filter paths which violate this constraint. We now explain each of these points.

Each state q in the ICP automaton is associated with a record r that summarizes the state of saturation for the polarities of each polarized category or co-anchor C present in the initial automaton. More specifically, for each category C, r records a triple $(root, sat, sub)$:

- *root* is the number of root nodes with category C whose polarity is not saturated
- *sat* is the number of root nodes with category C whose polarity has been saturated with substitution nodes placed occurring to the right of their anchor (i.e. with left companions)
- *lc* is the number of left companions associated with nodes of category C which have not yet been saturated.

For instance, the record associated with a state occurring after a single transition from the initial state labeled with the tree schema α_{imp} would be $\{\text{S}: (1,0,0), \text{NP}: (0,0,2), \text{to}: (0,0,1)\}$. This says that the tree schema α_{imp} introduces an unsaturated root node with category 'S', two left companions of category 'NP' and one of category 'to' (i.e. three unsaturated substitution sites occurring to the right of their anchor).

Note that information about right companions (i.e. unsaturated substitution sites occurring to the left of their anchor) is not recorded. This is because right companions must be saturated immediately and thus need not be passed on through the automaton. As we shall see below the information is used by the RC-constraint however thereby filtering out paths violating this constraint.

Two constraints ensure the elimination of sequences of grammatical items which cannot possibly lead to a valid parse.

The first constraint, the *right-companion constraint* requires that all states must be such that the sum $root + sat$ of saturated and unsaturated root nodes with category C is bigger or equal to the number rc of right companions of category C. This effectively enforces the left context constraint discussed in section 4 in that it requires that for any substitution node occurring to the left of the anchor there is a corresponding root node of the same category. This root node may be unsaturated (it has not been substituted in) or saturated (it has already been combined with a left companion, i.e. a substitution site occurring to the right of its anchor). In each case, it can be used to satisfy the right companion requirement and the polarity information is updated accordingly, i.e. either *root* or *sat* is decremented, and *lc* is possibly incremented.

Thus for instance, given the sentence 'Say it to John', selecting the transitive tree schema α_{tr} for 'say' would result in a state tagged with $\{\text{S}\colon(1,0,0),\ \text{NP}\colon(0,0,1)\}$ where the number of saturated and unsaturated roots of category 'NP' (0) is smaller than the number of right companions (1). Since the RC-constraint is violated this state is not created and the path aborts. As a result, all combinations of α_{tr} with the tree schemas selected by 'it', 'to' and 'John' will be ignored during parsing.

The second constraint, the *valid path constraint*, states that in the final state of a valid path (i.e. a path that will be considered for parsing), the *lc* value is null for all categories (all substitution nodes occurring to the right of the anchor of their containing tree have been filled), the *root* value is 1 for the S category and null for all other categories (all root nodes have been substituted in a substitution site). Since the RC-constraint ensures that all right companions have been saturated, this additional constraint ensures that the ICP principle is respected, i.e. that each every polarity on a path of the automaton has a companion on the same path, and further that each companion on a path is the companion of exactly one polarity on this path.

In sum, we have informally shown that the ICP algorithm integrates the Left-Context constraint. A formal proof is outstanding. An other open issue concerns the relative efficiency of the two algorithms (ICP and Left-Context Polarity filtering). The state information carried by the states of the ICP is richer than that of the Left-Context algorithm, thereby inducing a larger number of states when building the automaton. To assess the impact of this difference on effective running times, an empirical comparison would be needed.

7 Conclusion

In this paper, we presented a polarity based lexical disambiguation technique for LTAG and showed that it significantly reduces the size of the automata from which parsing proceeds. We also related out proposal to the general framework for polarity based lexical selection disambiguation presented in Bonfante et al. [3], Perrier [14], Morey [11]. Future work includes a formal proof of the relation between these approaches and a large scale empirical evaluation of the impact of the proposed filtering technique on parsing efficiency. Another interesting question concerns the definition of polarization. Here we restricted polarities to syntactic categories. It would be interesting to investigate how the approach extends to the rich set of feature values usually present in large scale LTAG.

References

1. Bangalore, S., Joshi, A.: Supertagging: An approach to almost parsing. Computational Linguistics 25(2), 237–265 (1999)
2. Bonfante, G., Guillaume, B., Morey, M.: Dependency constraints for lexical disambiguation. IWPT 2009. pp. 242–253 (2009)
3. Bonfante, G., Guillaume, B., Perrier, G.: Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation. COLING 2004. pp. 303–309 (2004)
4. Bonfante, G., Le Roux, J., Perrier, G.: Lexical disambiguation with polarities and automata. CIAA 2006. pp. 281–282 (2006)
5. Boullier, P.: Supertagging: a non-statistical parsing-based approach. IWPT 2003. pp. 55–66 (2003)
6. Crabbé, B.: Représentation informatique de grammaires fortement lexicalisées : application à la grammaire d'arbres adjoints. Ph.D. thesis, Université Nancy 2 (2005)
7. Duchier, D., Le Roux, J., Parmentier, Y.: The metagrammar compiler: an NLP application with a multi-paradigm architecture. MOZ 2004. pp. 175–187 (2004)
8. Joshi, A., Schabes, Y.: Tree-adjoining grammar. Handbook of Formal Languages, pp. 69–123. Springer (1997)
9. Lang, B.: Recognition can be harder than parsing. Computational Intelligence 10, 486–494 (1994)
10. 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., Arnold, D.: TSNLP – Test suites for natural language processing. COLING 1996. pp. 711–716 (1996)
11. Morey, M.: Étiquetage grammatical symbolique et interface syntaxe-sémantique des formalismes grammaticaux lexicalisés polarisés. Ph.D. thesis, Université de Lorraine (2011)
12. Parmentier, Y., Kallmeyer, K., Lichte, T., Maier, W., Dellert, J.: A syntax-semantics parsing environment for mildly context-sensitive formalisms. TAG+9. pp. 121–128 (2008)
13. Perrier, G.: Interaction grammars. COLING 2000. pp. 600–606 (2000)
14. Perrier, G.: Désambiguisation lexicale à l'aide d'automates de polarités. Tech. rep. INRIA (2008), `http://hal.archives-ouvertes.fr/inria-00278443`