From Metagrammars to Factorized TAG/TIG Parsers

Éric Villemonte de la Clergerie
INRIA - Rocquencourt - B.P. 105
78153 Le Chesnay Cedex, FRANCE
Eric.De_La_Clergerie@inria.fr

Abstract

This document shows how the factorized syntactic descriptions provided by Meta-Grammars coupled with factorization operators may be used to derive compact large coverage tree adjoining grammars.

1 Introduction

Large coverage Tree Adjoining Grammars (TAGs) tend to be very large, with several thousands of tree schemata, i.e., trees with at least one anchor node. Such large grammars are difficult to develop and maintain. Of course, their sizes have also a strong impact on parsing efficiency. The size of such TAGs mostly arises from redundancies, due to the extended domain of locality provided by trees.

Recently, Meta-Grammars (Candito, 1999) have been introduced to factorize linguistic information through a multiple inheritance hierarchy of small classes, each of them grouping elementary constraints on nodes. A MG compiler exploits these bits of information to generate a set of trees. While MGs help reducing redundancies at a descriptive level, the resulting grammars remain very large.

We propose to exploit the fact that MGs are already factorized to get compact grammars through the use of factorized trees, as provided by system DYALOG (Thomasset and Villemonte de la Clergerie, 2005).

This proposal has been validated by quickly developing and testing a large coverage French MG.

2 Generic factorization operators

The first factorization operators provided by DYALOG are the *disjunction*, *Kleene star*, and *optionality* operators. A finer control of optionality is provided through the notion of *guards*, used to state conditions on the presence or absence of a node (or of a node sequence). An expression $(G_+, x; G_-)$ means that the guard G_+ (resp. G_-) should be satisfied for x to be present (resp. absent). A guard G is a boolean expression on equations between FS paths and is equivalent to a finite set of substitutions Σ_G. Used to handle local free-word orderings, the *interleaving* (or shuffling) of two sequences $(a_i)_{i=1\cdots n}\#\#(b_j)_{j=1\cdots m}$ returns all sequences containing all a_i and b_j in any order that preserves the original orderings (i.e., $a_i < a_{i+1}$ and $b_j < b_{j+1}$).

These operators do not increase the expressive power or the worst-case complexity of TAGs. They are implemented without expansion, ensuring good performances and more natural parsing output (with no added non-terminals).

3 Meta-Grammars

MGs allow modular descriptions of syntactic phenomena, using elementary constraints grouped into classes. A class may inherit constraints from several parent classes and can also provide a resource or require a resource. Constraints on nodes include equality, precedence, immediate and indirect dominances. The constraints may also be on node and class decorations, expressed with Feature Structures.

The objective of our MG compiler, also developed with DYALOG, is to cross the terminal classes (i.e. any class without descendants) in order to obtain *neutral classes* where each provided resource has been consumed and conversely. Constraints are accumulated during crossing and are only kept the neutral classes whose accumulated constraints are satisfiable, taking into account their logical consequence. Minimal trees satisfying the constraints of the neutral classes are then produced.

Getting factorized trees results from several mechanisms. A node may group alternatives, and may be made optional or repeatable (for Kleene stars). When generating trees, underspecified precedences between sibling nodes are handled by the interleaving operator.

Positive and negative guards may be attached to nodes and are accumulated in a conjunctive way during the crossing phase, i.e. $N \Rightarrow G_1$ and $N \Rightarrow G_2$ is equivalent to $N \Rightarrow (G_1, G_2)$. The compiler checks the satisfiability of the guards, removing the alternatives leading to failures and equations in guards which become trivially true. The remaining guards are emitted as DYALOG guards in the trees.

4 Grammar anatomy

In just a few months, we have developed, for French, a MG with 191 classes, used to generate a very compact TAG of only 126 trees. Only 27 trees are anchored by verbs and they are sufficient to cover canonical, passive and extracted verbal constructions with at most 2 arguments (including objects, attributes, completives, infinitives, prepositional arguments, wh-completives). These trees would correspond to several thousand trees, if the factorization operators were expanded. This strong compaction rate stems from the presence of 820 guards, 92 disjunctions (to handle choices in realizations), 26 interleavings (to handle verb argument positions) and 13 Kleene stars (to handle coordinations). The grammar is mostly formed of simple trees (with less than 17 nodes), and a few complex trees (26 trees between 30 and 46 nodes), essentially anchored by verbs.

For instance, tree #111[1], used for canonical verb constructions, results from the crossing of 25 terminal classes, and has 43 nodes, plus 3 disjunction nodes (for the different realizations of the subject and other verb arguments) and 1 interleaving node (between the verb arguments and a possible post-verbal subject). The tree is controlled by 35 guards, governing, for instance, the presence and position of a subject and of clitics.

Such a tree covers much more verb subcategorization frames than the number of frames usually attached to a given verb. The anchoring of a tree α by a word w is done by unifying two feature structures $\mathcal{H}_\alpha$ and $\mathcal{H}_w$, called *hypertags* (Kinyon, 2000), that list the syntactic properties covered by α and allowed by w. The link between $\mathcal{H}_\tau$ and the allowed syntactic constructions is done through the variables occurring in $\mathcal{H}_\tau$ and in the guards and node decorations.

5 Evaluation

The resulting French grammar has been compiled, with DYALOG, into an hybrid TAG/TIG parser, by identifying the left and right auxiliary insertion trees. Following a left-to-right top-down tabular parsing strategy, the parser may be used to get either full or partial parses.[2] Coverage rate for full parsing is around 95% for two test suites (EUROTRA and TSNLP) and around 42% on various corpora (including more than 300K sentences of a raw journalistic corpus).

Our MG is still very young and needs to be improved to ensure a better coverage. However, we can already conclude that coupling MGs with factorized trees is a generic and powerful approach to control the size of grammars and to get efficient parsers.

The various tools and linguistic resources mentioned in this abstract are freely available at `http://atoll.inria.fr/`.

References

M.-H. Candito. 1999. *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées*. Ph.D. thesis, Université Paris 7.

A. Kinyon. 2000. Hypertags. In *Proc. of COLING*, pages 446–452.

F. Thomasset and É. Villemonte de la Clergerie. 2005. Comment obtenir plus des méta-grammaires. In *Proceedings of TALN'05*, volume 1, pages 3–12, Dourdan, France, June. ATALA.

[1]browsable online at `http://atoll.inria.fr/perl/frmg/tree.pl`.

[2]The parser may be tried online at `http://atoll.inria.fr/parserdemo`.