An Integrated Parser for TFG with Explicit Tree Typing

Marc Cavazza
EIMC Department, University of Bradford
Bradford, BD7 1DP, United Kingdom
M.Cavazza@Bradford.ac.uk
http://www.eimc.brad.ac.uk/~mcavazza

1. Introduction

One of the main conditions for the development of successful NLP applications is the usability of the syntactic formalisms adopted and the degree to which they facilitate syntax-semantics integration. TAG+ formalisms show a real potential for NLP applications, due to their linguistic descriptive capabilities. However, both the standard formalisms and parsing strategies are often quite complex and cannot easily be used as such for the development of NLP systems. We have thus investigated a simplified TAG+ formalism, which sacrifices some of TAG's descriptive and formal properties for the sake of usability. This is especially relevant considering the recent success of empirical approaches to NLP which tend to be based on very simple techniques and/or discard linguistically-motivated formalisms [Basili et al., 1996] [Appelt et al., 1993]. We report the implementation of a parser for a simplified TAG+ formalism, Tree Furcating Grammars (TFG), which integrates semantic processing, performing both syntactic disambiguation and the construction of a semantic representation for the sentence parsed. The parser has been developed for the purpose of real-time speech understanding of sublanguages (i.e., application-dependent vocabularies of 500-1000 words with specific, sometimes quite simplified, syntactic constructs). TAG+ formalisms were initially investigated because of their potential for syntax-semantics integration (see e.g., Abeille [1994]). We will successively describe the rationale for the TFG formalism, the principles underlying the algorithm used and a first assessment of its performance.

2. The Tree Furcating Grammars (TFG) Formalism

Tree Furcating Grammars are a lexicalised TAG+ formalism, in which adjunction is replaced by the *furcation* operation that essentially adds an additional branch to the target node in the initial tree, instead of copying the auxiliary tree under it. The furcation operation was originally introduced in segment grammars [De Smedt & Kempen, 1990]. A detailed comparison of furcation and adjunction has been given by Abeille [1991]. Though some syntactic phenomena are not properly handled by furcation, the fact that it introduces modifiers without embedding them into the tree structure is a definite advantage for syntax-semantics integration, and was the rationale for choosing it[1]. Successive furcations do not increase tree depth and complexity, producing derived trees that retain some properties of dependency trees. These can support the integrated construction of a semantic structure, based on the appropriate association of semantic functions to the tree structures (see below).

We have adapted our tree representations accordingly, by distinguishing between left auxiliary trees (which have a *X root node)[2] and right auxiliary trees (X* root node). The auxiliary symbol is on the root node, as these trees do not have a foot node. Also, in our implementation trees are explicitly typed as left or right auxiliary (l-aux, r-aux), initial and left or right substituable (l-subst, r-subst). Trees can have multiple types, for instance being both right and left substituable or, in the case of some PP trees, both left auxiliary and right substituable (e.g., fig 2, *V-with-N0).

Left (resp. right) auxiliary trees are combined through right (resp. left) furcation. Left and right furcations, as described by De Smedt & Kempen [1990] produce "flat" structures and in that sense differ from left and right adjunction in Tree Insertion Grammars [Schabes & Waters, 1994]. They tend to be closer to the operations described by Nasr [1995] for his dependency-based TAG variant. Also, furcations are not allowed to take place at substituable nodes prior to their substitution, but are allowed on auxiliary nodes (as compared with Schabes & Waters [1994]).

Another goal, which was the result of early experimentation, was to minimise tree traversal operations that can prove computationally expensive. These are minimised due to the representation itself and to the explicit recording of substituable leaves within tree representations. Only the determination of target nodes for furcation still requires tree

[1] We do not make a direct use of the properties of the derived tree, like dominance relations.
[2] With X in {P, N, V, A}.

traversal, but is made easier by the relatively flat structure of the derived trees.

Finally, a set of atomic semantic features, corresponding to the semantic description of the anchor is associated to the root node as well. These semantic features are used for semantic representations as well as selectional restrictions, in the spirit of preference semantics [Wilks, 1975]. Substituable nodes in initial and some auxiliary trees are associated semantic relations, which also constitute an explicit typing. The definition of these semantic relations can be quite specific, as it derives from the specific distributions of the lexicalised trees themselves [Cavazza, 1997].

3. The Integrated Parsing Algorithm

Several parsing algorithms have been described for TAG+, including CKY [Vijay-Shanker & Joshi, 1985] and Earley-type parsers [Schabes & Joshi, 1988] [Schabes et al., 1988] and a deterministic parser [Schabes & Vijay-Shanker, 1990], which was developed for reasons of efficiency (Schabes & Joshi, 1990). The latter has been recently revisited by Kinyon [1997], who proposed an improved LR(0) algorithm. Recently, Nederhof [1998] has described a new LR parsing method and a new recogniser based on Linear Indexed Automata. Specific approaches have also been developed for partial parsing of potentially ungrammatical sentences [Issac, 1994]. Another major source of innovation in parsing has been the many TAG+ variants developed in recent years, such as the "supertagging" approach [Joshi & Srinivas, 1994], dependency formalisms inspired by TAG [Nasr, 1995] and Tree Insertion Grammars [Schabes & Waters, 1994].

Due to the interleaving of syntactic and semantic processing in our system, we have opted for an *ad hoc* strategy, which eventually resulted quite similar to the one described by Nasr [1995]. The main idea is to make the syntactic part of the algorithm as simple as possible and to avoid "hidden" integration of syntax and semantics through contextual constraints on syntactic operations. Rather, keeping the parsing algorithm elementary would offer more space for experimentation and the integration of semantic processing.

The first step, which corresponds to a lexical filtering of the grammar, consists in generating all the possible set of trees (often termed *forests*) compatible with the input string. This step is very similar to the construction of a pushdown stack for trees as described in [Nasr, 1995]. The parsing algorithm consists in scanning the forest left-to-right and determining possible tree fusions from the explicit typing of the trees considered. The process is iterated until the forest is reduced to a single tree or no further operations are possible [Cavazza & Constant, 1996]. All the forests not reduced to a single tree are discarded as unsuccessful parses. Adjacent trees in a forest are considered for a possible fusion on a pairwise basis. From their explicit categories, the corresponding operation is given by a compatibility table. This table specifies the nature of the operation (substitution, furcation, or nil) as a function of the types of the adjacent trees. However, successful operations also depend on the existence of an appropriate target node as well as semantic compatibility (when applicable). In that sense, tree operations are not fully determined by the compatibility table. The target node for substitution is directly recorded in the representation for substituable trees, while target node for furcation is dynamically computed as being the rightmost/leftmost compatible node, including nodes internal to the tree. The forest is scanned left to right without look-ahead and the "cursor" backtracks one position after a successful fusion has been completed. The forest may have to be scanned several times, due to the conjunction of a strict left-to-right scanning with the restrictions imposed on tree operations. For instance, in the parsing of the forest on fig. 2, the first pass essentially assembles the nominal descriptions through furcation, and substitution at the N1 node takes place at the second pass only.

Additional heuristics are used as a declarative control strategy. For instance, whenever a PP tree is both of type l-aux and r-subst (like e.g., *V-with-N0), substitution has to be performed first, thus enabling correct semantic feature propagation, which will be subsequently needed for selectional restriction at (right) furcation time. This can be achieved by attributing precedence to some types; as a result some operations are postponed until proper conditions are met. It should be noted that PP attachments are a major requirement for the processing of definite descriptions, spatial expressions and instrumental actions, which constitute a significant fraction of the requirements for speech-based multimedia applications.

Throughout parsing, there is a full integration of semantic processing[3], which consists both in semantic features propagation and establishment of semantic/functional links for actants and various modifiers. Semantic features for a lexical entry are associated to the tree root and are transferred through furcation operations to the root node of the target tree (fig. 1 and 2). This ensures proper propagation of semantic features to constitute complete semantic frames.

[3] In that sense, our implementation would fall under the "Parallel" + "Generate-and-Test" paradigm for Syntax-Semantics integration [Dahl et al., 1992].

Furcation is responsible for semantic aggregation, while substitution establishes semantic relations between meaning units, essentially through the structure of initial trees of root S. However, furcation can also result in the establishment of semantic relations, for instance instrumental cases, as with *V-with-N0 trees. Figures 1 and 2 illustrate two different cases of selectional restriction, implementing the PP-attachment rules described above[4].

4. Results

The system is implemented in Common LISP and runs on a SGI O2 workstation with an R10000 processor at 150 MHz. Processing of a single forest corresponding to a 10-15 word sentence is regularly carried in 10-20 ms CPU time. The important point is that, even when parsing several forests for a sentence, the user time remains below 200 ms. Though this was measured with small vocabularies (typically less than 300 words), it is expected to remain roughly unchanged with the target application vocabulary being approx. 500 words in size. The reason is that global response times depend on the number of forests to parse, which is a function of the trees/word ratio. This ratio tends to remain stable within small sublanguages and is certainly much smaller than the generic ratio of 7 mentioned in [Schabes & Waters, 1994]. It is interesting to compare these results to the requirements proposed by Goerz and Kessler [1994] for anytime algorithms to be used in speech understanding. They give Result Production Granularity (RPG) values in the range of 10-100 ms, which means that in most cases our parser, developed for similar applications, could fit into that range.

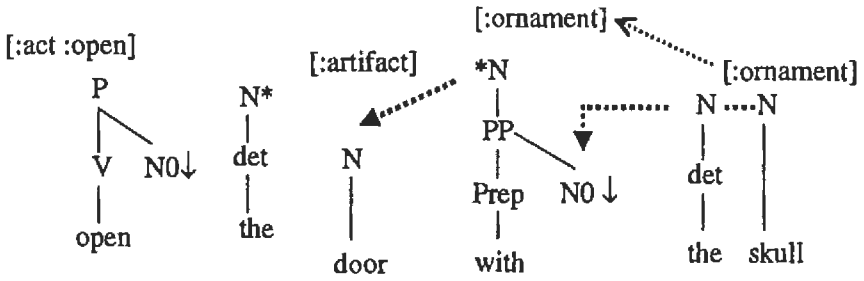

Fig. 1. Semantic propagation through substitution ("N0" node of the PP group) enables selection of right furcation on "N", because of compatibility between :artifact and :ornament features.

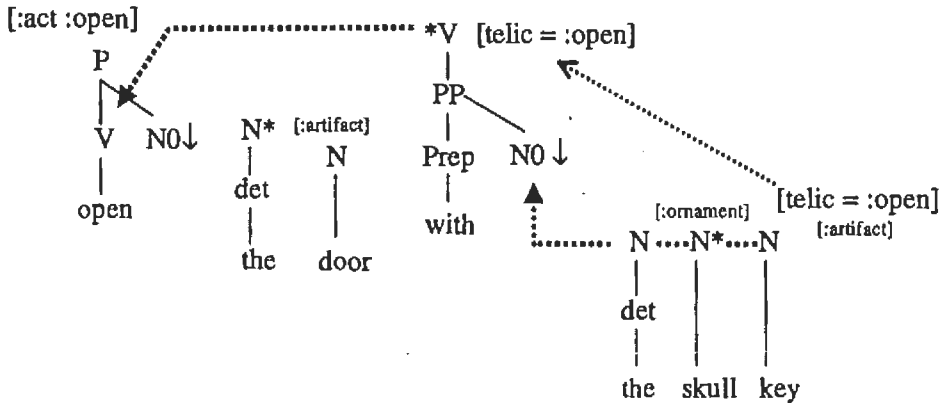

Fig. 2. Semantic propagation through substitution ("N0" node of the PP group) enables selection of right furcation on "V", because of compatibility between "telic" features *and* feature precedence rules.

[4] These refer to situations encountered in the popular "DOOM" video game (trademark of ID Software).

References

Abeillé, A., 1991. Une grammaire lexicalisée d'arbres adjoints pour le francais: application a l'analyse automatique. These de Doctorat de l'Université Paris 7.

Abeillé, A., 1994. Syntaxe et Sémantique: Interactions dans une grammaire d'unification. In: F. Rastier, M. Cavazza, A. Abeille, Sémantique pour l'Analyse. Paris, Masson. (English translation to appear, CSLI, University of Stanford Press).

Appelt, D. E., Hobbs, J. R., Bear, J., Israel, D. & Tyson M. (1993). FASTUS: A Finite-state Processor for Information Extraction from Real-World Text. In *Proceedings of the IJCAI'93 Conference.*

Basili, R., M.T. Pazienza, & Velardi, P., 1996. An Empirical Symbolic Approach to Natural Language Processing, *Artificial Intelligence*, vol. 85.

Cavazza, M. & Constant, P., 1996. Le Traitement Automatique du Langage Naturel et ses Applications. In: J. Caelen (Ed.), Nouvelles Interfaces Homme-Machine, Paris: Tec & Doc.

Cavazza, M., 1997. Sémiotique textuelle et contenu linguistique. *Intellectica*, vol. 23, pp. 53-78.

Dahl, D., Weir, C., Norton, L.M. & Linebarger, M., 1992. Integration of Syntax and Semantics in the Pundit System. *Proceedings of the ANLP'92 Workshop on Fully-Implemented NLP Systems*, IBM Technical Report, TR-80.92-033.

De Smedt, K. & Kempen, G., 1990. Segment Grammars: a Formalism for Incremental Sentence Generation. In: C. Paris (Ed.) *Natural Language Generation and Computational Linguistics*, Dordrecht, Kluwer.

Goerz, G. & Kesseler, M., 1994. Anytime Algorithms for Speech Parsing? *Proceedings of COLING'94*, Kyoto.

Issac, F., 1994. Un algorithme d'analyse pour les grammaires d'arbres adjoints. In: Proceedings of the Third International TAG Workshop, Paris. Technical report TALANA-RT-94-01, Universite Paris 7.

Joshi, A.K, & Srinivas, B., 1994. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. *Proceedings of COLING'94*, Kyoto.

Kinyon, A., 1997. Un algorithme d'analyse LR(0) pour les Grammaires d'Arbres Adjoints Lexicalisees. *Proceedings of TALN'97.*

Nasr, A., 1995, A Formalism and a Parser for Lexicalized Dependency Grammars, *Proceedings of the Fourth International Workshop on Parsing Technologies*, Prague.

Nederhof, M.-J., 1998. Linear Indexed Automata and Tabulation of TAG Parsing. *Proceedings of the First Workshop on Tabulation in Parsing and Deduction*, Paris.

Rambow, O., 1994. Multiset-Valued Linear Index Grammars: Imposing Dominance Constraints on Derivations. *Proceedings of the COLING'94 Conference*, Kyoto.

Schabes, Y., Abeillé, A. & Joshi, A.K., 1988. Parsing Strategies with "Lexicalized" Grammars: Applications to Tree-Adjoining Grammars. *Proceedings of COLING'88*, Budapest.

Schabes, Y., & Joshi, A.K., 1988. An Earley-Type Parsing Algorithm for Tree Adjoining Grammars. *Proceedings of ACL'88*, Buffalo.

Schabes, Y. & Joshi, A.K., 1990. Two Recent Developments in Tree Adjoining Grammars: Semantics and Efficient processing. *Proceedings of the Third DARPA Speech and Natural Language Workshop.*

Schabes, Y. & Vijay-Shanker, K, 1990. Deterministic Left to Right Parsing of Tree Adjoining Languages. *Proceedings of ACL'90*, Pittsburgh.

Schabes, Y. & Waters, R.C., 1994. *Tree Insertion Grammar: A Cubic-Time Parsable Formalism That Lexicalizes Context-Free Grammar Without Changing the Trees Produced*, Technical Report TR-94-13, Mitsubishi Electric Research Laboratories, Cambridge (MA).

Wilks, Y., 1975. A Preferential Pattern-seeking Semantics for Natural Language Inference. *Artificial Intelligence*, vol. 6, pp. 53-74.