Verification of Lexicalized Tree Adjoining Grammars

Valerie Barr, Ellen Siefring
Department of Computer Science
Hofstra University
Hempstead, NY 11549-1030 USA
vbarr@hofstra.edu

Abstract

One approach to verification and validation of language processing systems includes the verification of system resources. In general, the grammar is a key resource in such systems. In this paper we discuss verification of lexicalized tree adjoining grammars (LTAGs) (Joshi and Schabes, 1997) as one instance of a system resource, and as one phase of a larger verification effort.

1 Introduction

The work presented here is part of a larger project that has the goal of developing a suitable automated approach to verification and validation of natural language processing (NLP) systems, including structural (white-box) (Beizer, 1990) testing techniques that are suitable for language applications. In previous work (Barr and Klavans, 2001) we established that it is worthwhile to adapt for NLP systems the standard verification, validation, and testing practices that have been developed in the software engineering and intelligent systems communities. These new techniques will supplement the evaluation practices currently carried out and, in many cases, will not require significantly larger test sets.

For our working definitions we combine definitions from the intelligent systems community (e.g. (Gonzalez and Barr, 2000)) and from the software engineering community (e.g. (Voas and Miller, 1995)), as follows:

- Verification – the process of ensuring 1)that the intelligent system conforms to specification, and 2) its knowledge base is consistent and complete within itself; the application of dynamic software testing techniques involving both functional (black-box) and structural approaches.
- Validation – the process of ensuring that the output of the intelligent system is equivalent to that of human experts when given the same input.

As we have noted elsewhere (Barr and Klavans, 2001) there are a number of diagnostic evaluation methods that do a validation check on a system by carrying out a functional test and comparing actual results to expected results (provided by and compared by humans). There are also evaluation methods that allow us to determine whether a system conforms to its specification.

There are a number of methods that are still needed, however. First, we need to determine whether the knowledge represented within an NLP system is consistent and complete. The research presented in this paper begins to address this topic. Specifically, we detail work we have done on the verification of Lexicalized Tree Adjoining Grammars (LTAGs), specifically as implemented in the XTAG formalism (Joshi and Schabes, 1997). As described in the body of the paper, we have constructed a set of structural and relational tests for a LTAG that identify certain lexical and syntactic errors. We applied these tests to subsets of XTAG for English (as examples of a sublanguage in the XTAG formalism), using off-the-shelf database software.

In addition to the above, we need to determine ways by which we can obtain the benefits of structural testing for NLP systems and their components. This will be the subject of future research we plan to carry out. An additional open question, which we do not address here, is whether a more complete verification process will facilitate greater automation of the validation process.

NLP systems are built for a large number of application areas, such as speech recognition, language understanding, language generation, speech synthesis, information retrieval, information extraction, and inference (Jurafsky and Martin, 2000). Systems built for these application areas will differ in terms of the resources they include, the kind of input they expect, and the kind of output they

TAG+7: Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms.
May 20-22, 2004, Vancouver, BC, CA.
Pages 126-131.

generate. In order to narrow the scope of our work at this stage, we focus initially on natural language generation (NLG) systems.

2 Overview of Verification of NLG Systems

Dale and Mellish (Dale and Mellish, 1998) have suggested a direction for improving evaluation of NLG systems. Their proposal is that, rather than attempt to evaluate a complete system, the evaluation effort address the component tasks of the NLG process. They suggest a breakdown of the NLG process (Reiter and Dale, 2000; Dale and Mellish, 1998) into the six tasks of content determination, document structuring, lexical selection, referring expression generation, aggregation, and surface realization. This approach is consistent with our proposal (Barr and Klavans, 2001) that we carry out a component performance evaluation, in order to determine the impact on overall system performance of each subpart or subtask. In other work (Barr, 2003) we began to address the verification and validation questions relevant for each of these generation tasks. (The components of interest will differ across different types of language systems. See (Webber et al., 2002) for an example in the Question-Answering domain).

Another important area to consider is the issue of utilization of linguistic resources by a language processing system. This is an area that we believe cannot be adequately addressed by traditional testing approaches. Typically a language processing system has numerous resources within it, such as the lexicon, the grammar, morphological rules, a pragmatics component, and semantic knowledge (both formal and lexical).

There are a number of aspects of system behavior that are affected by the various resources. For example, it would be useful to clarify exactly how an incomplete lexicon affects system behavior. Or there may be sub-processes within a language generation system that should be verified separately because they utilize only a subset of the available resources. We are also interested in how the various resources participate in the input-output relationship. For example, can we determine which of a system's linguistic resources contributes to the transformation of an input to an output? Can we pinpoint exactly how each element of an output is affected by each linguistic resource? If the grammar in a generation system is capable of parsing a sentence, is there some context in which the system will generate that sentence?

Developing mechanisms for addressing these issues will enable us to more accurately assess the overarching verification issue, which is whether the system does the task, and only the task, for which it was intended. As part of our larger project we intend to define what it means to evaluate all the linguistic resources for completeness and consistency. As a first step in this aspect of verification, we focus on an assessment of the completeness and consistency of the grammar alone.

Previous testing approaches have attempted to identify grammar errors through evaluation of parse system coverage using test-suite or corpus-based methods (Doran et al., 1994; Doran et al., 1997; Bangalore et al., 1998; Prasad and Sarkar, 2000). While these testing approaches are vital to a complete test plan, the source of errors identified through these methods must be manually researched and categorized as a grammar or application defect. If a grammar error is suspected, the underlying grammar must be examined to determine if the error is a coverage issue or grammar fault. Our structural approach to grammar verification insures that grammar defects are identified and corrected early in the testing cycle, before the grammar is embedded in a component application, such as a parser. Our expectation is that this will improve grammar reliability, and subsequent test efforts may then focus on coverage and application defect issues.

3 Grammar Verification

The first step in verifying a grammar is to assess consistency and completeness. We cannot necessarily do this by applying existing methods from other domains. How we do it depends on the kind of grammar used. We have, from the expert systems' realm, methods and tools that are suitable for rule-based systems (for example, the TRUBAC tool (Barr, 1999)). However, the rule formalism, while used in some aspects of NLP, is frequently not used for grammar representation. Yet adapting to the grammar of an NLP system the underlying approach used for rule-based systems may give us the ability to determine consistency and completeness of a grammar.

The grammar formalism we focus on initially is the Lexicalized Tree Adjoining Grammar (LTAG), based on the original TAG formalism(Joshi et al., 1975; Joshi, 1987; Joshi and Schabes, 1997). Analysis of the consistency and completeness of an LTAG will serve as a first step toward the full verification and validation of the generation system in which the LTAG is used.

Our motivation to work with LTAGs, particularly with the XTAG formalism (XTAG Research Group, 2001), is threefold. First, we chose XTAG for English as a vehicle to demonstrate proof of concept of our verification approach. Certainly, given the extensive work that has been done on the XTAG for English, we did not anticipate that we would find any errors in the grammar. However, our expectation is that a verification methodology for XTAG grammars could also be adapted to other key grammar formalisms as well. Second, we assume that there are language systems for which a smaller, domain specific, grammar and sub-language would be desired. The grammar might be a subset of an existing XTAG, such as the XTAG for English, or it might be a newly constructed

grammar that employs the XTAG formalism (Kinyon and Prolo, 2002). The verification steps we propose would be able to detect errors or potential problems in such a grammar. Finally, any language system will be tested with a domain specific test suite. However, a set of static verification tests can serve as a useful and important step before a black-box test is carried out, and can potentially unearth grammar problems that might be masked in functional test results.

4 LTAG Verification

While it is possible that existing mechanisms for evaluating the consistency and completeness of the antecedent-consequent rules in an expert system could be used to do the same for the rewrite rules making up a phrase-structure grammar (PSG), these are not relevant for a grammar made up of trees, not rules. Given a grammar made up of trees, we cannot directly apply the characteristics that are used in evaluating rule-bases for consistency and completeness (conflict, redundancy, circularity, subsumption, unreachability, dead-ends, etc.), but rather must adapt the concepts of completeness and consistency for use with LTAGs.

The characteristics we currently check for in an LTAG can be divided into two categories, structural and relational. Structural tests include ensuring that each elementary tree is properly lexicalized, structurally correct and unique. This includes checking for proper tree hierarchy (e.g. unique root, one parent for each child node, proper tree level and node order) as well as TAG specific checks (e.g. each tree is properly anchored, leaf nodes marked with a phrasal label are substitution sites, no adjunction nodes exist in initial trees, label of adjunction node and root must be the same in an auxiliary tree). Trees with identical structures are flagged. Generally, structural errors will arise from incorrect coding or errors in the translation of the LTAG into a machine representation.

Relational tests look at the relationships between tree structures to identify that:

1. Each auxiliary tree can adjoin in at least one derived tree structure, i.e. every auxiliary tree can be used.

2. Each non-S rooted initial tree can substitute in at least one derived tree structure, i.e. every initial tree can be used.

3. At least one substitution operation can be performed at every substitution node in a derived tree, i.e. a tree exists for each substitution node.

4. All derived trees built using substitution operations are finitely bounded with no recursive end nodes (no recursive sentences or phrases). We cannot eliminate recursion, since adjunction allows unbounded sentences. However, if we consider only substitution, we can insure that a tree substituted at a node does not contain a node with the same phrasal label as an ancestor node.

5. Every sentence that can be built using substitution operations alone has a unique derivation tree structure. While the existence of multiple derivation tree structures does not necessarily represent a grammar error if part-of-speech ambiguity is considered, it could indicate conflicting semantic representations if tree anchors are not properly chosen with respect to linguistic relevance.

These checks on the grammar enable us to identify potential grammar errors such as

1. superfluous trees, which could be indicative of missing trees or errors in other trees. (A tree T may be superfluous, or unusable, because there is no other tree that presents a suitable adjunction or substitution use for T, or because there are errors that prevent a suitable adjunction or substitution site from being identified as such).

2. invalid tree structures, a grammar error which could cause an incorrect generation path to be chosen.

3. missing trees, which may indicate incomplete/inaccurate linguistic realization or communicative intent compromised.

4. duplicate trees, which will violate consistency.

5. redundant trees, which may indicate conflicting linguistic interpretations of anchor. This could happen if the linguistic assumptions on how elementary trees should be formed are not consistently followed in the grammar.

We are presently working on an extension of the work presented here that will identify relational problems in feature based LTAGs. (A static analysis approach that identifies structural problems with feature structures in XTAG (typographical errors, reference of undefined features, equating of incompatible features) is introduced in (Sarkar and Wintner, 1999)).

5 Implementation

We have constructed a system that carries out the above verification checks for an LTAG, employing a relational data representation of LTAG tree structures using the Oracle Database Management System. This relational database model provides the benefits of data independence (with the ability to separate the physical implementation from the logical view), multiple views of the same data (through structured queries across tables), data

consistency (enforcing completeness and consistency of schema), and management of data relationships (via table indexes, primary and foreign keys). In addition, the DBMS approach allows us to efficiently manage and access large quantities of structured data which insures future scalability for large grammars. Data verification is performed using SQL*PLUS, the PL/SQL language and reporting tool of the Oracle Database Client/Server product.

The system operates in four stages: tree conversion, structural testing, relational testing and reporting. Oracle tables are used to store type, classification, and node information about each tree. Tree structures in the grammar are automatically converted into the SQL Data Manipulation Language format to systematically build the associated Oracle tables. Structural tests are performed on each converted table to insure tree and lexical consistency. Relational tests perform comparisons on groups of tree structures to identify missing trees, unused trees and, using substitution operations alone, recursive and non-unique derivations. Control and error information is generated during the verification process.

Initially, tree structures are converted to a non-indexed database table set. This enables structural tree errors such as duplicate nodes to be identified and classified by our testing tool, not the DBMS product. A second conversion is then performed to assign primary and foreign keys to tables, encapsulating the data relationships into the structures. Tree nodes are stored as separate table rows, with identifying tree hierarchy represented as three-tuples of (level, order, parent order). Tree traversals may be accomplished in any order using either the indexed keys or identifying node characteristics (e.g. substitution nodes). Substitution and adjunction operations are performed using constrained table join operations.

6 Results

We have used the XTAG for English to test our grammar verification tool. Since XTAG system releases have been extensively utilized and broadly tested (Doran et al., 1994; Doran et al., 1997; Bangalore et al., 1998; Prasad and Sarkar, 2000; XTAG Research Group, 2001), we did not expect our verification tool to uncover any structural grammar defects in the current release of XTAG. We did, however, expect to identify non-unique derivation structures due to inherent sentence ambiguity in the English language. Additionally, we expected to identify as duplicates certain tree structures that are unique when node features are taken into account.

The results from our grammar verification on XTAG are encouraging. We ran our grammar verification tool on an XTAG set of 1,135 trees with a total of 11,514 nodes. We are able to make the following observations from our results:

- There are no errors in tree hierarchy. Every tree has one unique root node. Each non-root node has one parent node and consistent tree node level and ordering.
- Two tree structures have unidentified part of speech node values. Both trees have internal nodes of 'p'. Since we consider case in our validation of POS, these nodes were flagged as errors.
- There were 128 duplicate tree structures in the grammar. This was an anticipated result. Our expectation is that when we consider node features these trees will be identified as unique structures.
- Every tree was properly lexicalized. That is, there was at least one anchor node identified for each tree structure.
- There were three errors in tree classification. One tree was classified as an auxiliary tree but structurally looks like an initial tree. Two trees were classified as initial trees but structurally look like auxiliary trees. We used XTAG tree naming conventions as alpha or beta to drive our classification scheme. It must be determined if the conversion requirements must be modified or if this is a tree classification discrepancy in XTAG.
- Other than the three trees with classification discrepancies, every elementary tree was structurally correct. Every non-terminal node on the frontier marked with a phrasal label was identified as a substitution node. There were no internal nodes marked for substitution, and in the initial trees no internal nodes were marked as adjunction nodes . For auxiliary trees, there was one unique adjunction node per tree. This adjunction node was on the frontier and matched the POS node value of the tree root.
- Every non-S rooted initial tree was able to substitute in at least one derived tree structure. All initial trees could be used in the grammar.
- Every auxiliary tree was able to adjoin in at least one derived tree structure. All auxiliary trees could be used in the grammar.
- There exists at least one tree eligible for substitution at each substitution node in the grammar. Substitution operations may be performed until all frontier nodes are terminals.
- Application performance could be improved by database performance and tuning techniques. While proof of concept, not processing efficiency, was the

initial motivation for this work, subsequent development efforts should consider performance as an implementation requirement.

Identifying non-unique derivation structures using the full XTAG has proven more difficult. While the use of Oracle as our implementation paradigm allows us to efficiently retrieve, manipulate and store large amounts of data, our attempt to build all possible sentence derivations for a complete grammar proved too exhaustive. We modified our approach to maintain derived tree structures and linearize the nodes for comparison. This worked for simple sentence structures but did not scale up to more complex sentences. We continue to work on a viable solution for this problem. It may be that we are facing a limitation inherent in our choice of the database management system approach. A more recursive-based implementation strategy may be necessary.

One motivation of this work is to provide a tool for verification of smaller, domain specific grammars that may be subsets of larger grammars, such as XTAG. We simulated such a grammar by extracting a subset of XTAG trees and applying our verification tool to this grammar. Since the tree subset was randomly chosen without linguistic significance, we expected our verification tool to identify gaps in the grammar. Our verification tool reported several defects in the grammar including missing trees for substitution nodes and unused elementary trees. Working with a subset of 105 trees from XTAG, our system was able to identify 5 duplicate tree structures, 7 missing trees for substitution nodes and one superfluous tree. The complete verification process on this subset of 105 trees with an average of 12 nodes per tree took less than 20 seconds.

We expanded our test subset to simulate additional grammar errors. Duplicate tree structures were identified when node features were ignored. Extracted tree structures were manually changed to generate structural errors. Tree nodes were added to produce recursive phrases. Trees needed for sentences with multiple parses were selected for the grammar subset.

Our verification tool successfully identified all grammar defects with this handcrafted grammar subset. While some implementation issues remain for large grammars, we have shown that stand-alone grammar verification can be a useful initial test strategy in a complete NLP structural test plan. Grammar errors can be identified and corrected at their source, before the grammar is embedded in a component application. This improves grammar reliability so subsequent test efforts may focus on coverage and application defect issues.

7 Conclusions and Future Work

The set of structural and relational checks we have described can serve as the first stage of verification analysis for an LTAG. At present we have a stand-alone system, easily usable by an NLG researcher, that will convert a grammar into the DBMS format and perform the LTAG verification checks. More experimentation needs to be done to determine how the static identification of grammar errors affects the overall system development process and the quality of the final system. In addition, as these grammar checks do not guarantee any kind of semantic coherence, we are presently extending our approach to feature-based LTAGs, where elements of semantic coherence are enforced within the structure of the grammar components, so that a verified grammar is more likely to generate semantically coherent sentences. Much work remains to address the larger issues of resource verification, verification of generation tasks, and the application of structural testing to language processing systems. Finally, we plan to apply our verification approach to more complex grammars, including one that generates text combined with gestures for an embodied conversational agent.

Acknowledgments

The authors thank Bonnie Webber and Matthew Stone for their continued interest in and suggestions about this work.

References

Srinivas Bangalore, Anoop Sarkar, Christine Doran, and Beth Ann Hockey. 1998. Grammar and parser evaluation in the xtag project. In *Proceedings of the Workshop on Evaluation of Parsing Systems*, Granada,Spain. Language Resources and Evaluation Conference.

Valerie Barr and Judith Klavans. 2001. Verification and validation of language processing systems: Is it evaluation? In *Proceedings of the Workshop on Evaluation Methodologies for Language and Dialogue Systems, ACL2001*, Toulouse, France. Association of Computational Linguists.

Valerie Barr. 1999. Applications of rule-base coverage measures to expert system evaluation. *Journal of Knowledge Based Systems*, 12:27–35.

Valerie Barr. 2003. A proposed model for effective verification of natural language generation systems. In *Proceedings of Florida Artificial Intelligence Research Symposium 2003*, Saint Augustine, FL.

Boris Beizer. 1990. *Software Testing Techniques*. Van Nostrand Reinhold, New York.

Robert Dale and Chris Mellish. 1998. Towards evaluation in natural language generation. In *Proceedings*

of the 1st International Conference on Language Resources and Evaluation, Granada, Spain, May.

C. Doran, D. Egedi, B.A.Hockey, B. Srinivas, and M. Zaidel. 1994. Xtag system - a wide coverage grammar for english. In *Proceedings of the International Conference on Computational Linguistics (COLING '94)*, Kyoto, Japan.

Christine Doran, Beth Hockey, Philip Hopely, Joseph Rosenzweig, Anoop Sarkar, B. Srinivas, Fei Xia, Alexis Nasr, and Owen Rambow. 1997. Maintaining the forest and burning out the underbrush in xtag. In *Proceedings of the Workshop on Computational Environments for Grammar Development and Language Engineering (ENVGRAM)*, Madrid, Spain. Association of Computational Linguists.

Avelino Gonzalez and Valerie Barr. 2000. Validation and verification of intelligent systems - what are they and how are they different? *Journal of Experimental and Theoretical Artificial Intelligence*, 12(4), October.

A. Joshi and Y. Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–124. Springer, Berlin.

Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10:136–163.

Aravind Joshi. 1987. An introduction to tree adjoining grammars. In A. ManasterRamer, editor, *Mathematics of Language*, pages 87–113. John Benjamins, Amsterdam.

Daniel Jurafsky and James Martin. 2000. *Speech and Language Processing*. Prentice-Hall, New Jersey.

Alexandra Kinyon and Carlos A. Prolo. 2002. A classification of grammar development strategies. In *Proceedings of the Workshop on Grammar Engineering and Evaluation*, pages 43–49, Taipei, Taiwan.

Rashmi Prasad and Anoop Sarkar. 2000. Comparing test-suite based evaluation and corpus-based evaluation of a wide-coverage grammar for english. In *Using Evaluation within Human Language Technology Programs: Results and Trends*, Athens, Greece. LREC 2000 Satellite Workshop.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge, UK.

Anoop Sarkar and Shuly Wintner. 1999. Typing as a means for validating feature structures. In *Proceedings of Computational Linguistics in The Netherlands 1999 (CLIN99)*, Utrecht. CLIN.

Jeffrey M. Voas and Keith W. Miller. 1995. Software testability: The new verification. *IEEE Software*, 12(3):17–28.

Bonnie Webber, Claire Gardent, and Johan Bos. 2002. Position statement: Inference in question answering. In *LREC'02 Workshop on Question Answering: Strategy and Resources*. Las Palmas.

XTAG Research Group. 2001. A lexicalized tree adjoining grammar for English. Technical Report IRCS-01-03, IRCS, University of Pennsylvania.