Expanding Tree Adjoining Grammar to create Junction Grammar trees

Ronald Millett
BYU Linguistics
RonMillett@byu.edu

Deryle Lonsdale
BYU Linguistics
lonz@byu.edu

Abstract

Junction Grammar (JG) combines junction operators, multiple linked syntax/semantics trees, and flexible traversal algorithms. The multiple tree and flexible ordering characteristics of MC-TAG and other TAG extensions are somewhat analogous. This paper proposes that these similarities can be integrated to form a new approach, JG-TAG. Relevant aspects of both theories and the proposed new model are discussed in turn, and representative examples are sketched.

1 Introduction

This paper presents an enhanced version of Tree Adjoining Grammar (TAG) that can create trees based on Junction Grammar (JG), a linguistic theory that was proposed in the mid-1960's. The JG tree structures discussed in this paper represent syntactic/semantic structures with node joining operators, multiple linked trees and flexible tree traversal algorithms that recode the concepts of a sentence into a separate articulation tree. Adding junctions and multiple linked trees to Multiple Component TAG (MC-TAG) and other enhancements results in a formalism that can capture more sophisticated sentence relationships using fewer TAG trees than would otherwise be required. With the advantages that TAG's expanded domain of locality provides, an enhanced TAG-like system for JG could also provide an improved platform for computational linguistic applications of JG.

2 Theoretical background

2.1 Overview of Junction Grammar

Junction Grammar (JG) is a linguistic theory proposed in the 1960's (Lytle, 1971; Lytle, 1974; Lytle, 1985; Melby, 1985) that is still being pursued. The theory was developed as a reaction to early Transformational Grammar and challenged many of the basic assumptions that TG was based upon (Lytle, 1979). Early applications of JG have included machine translation (Gibb, 1970; Billings, 1972; Gessel, 1975; Lytle, 1975; Melby, 1978) including the development of a JG transfer language between source and target languages (Melby, 1974); speech synthesis (Melby, 1976; Millett, 1976); and second language instruction (Olsen and Tuttle, 1973). More recent efforts have involved PC-based grammar checking (Lytle and Mathews, 1986), automatic holistic scoring of essays (Breland and Lytle, 1990), and secondary school English grammar teaching (Millett and Lytle, 2004).

Junction Grammar challenged the notion that a basic grammar involves simple concatenation of strings via phrase structure rewrite rules. A fundamental premise of Junction Grammar is that JG operators (and their associated operands) constitute the basic building blocks of grammar. A well-defined process specifies operators and their appropriate application. The basic operator names and their symbols are: conjunction (&), adjunction (+), subjunction (∗), and interjunction (a combination of adjunction and subjunction). Two types of JG trees involving these operators are discussed in this paper: concept trees and articulation trees. Nodes in concept trees have a basic category label (N, V, A or P), predicate/phrase label (PN, PV, PA, or PP) or sentence/clause label (SN, SV, SA, or SP).

Figure 1 shows the JG concept tree for a simple sentence involving JG conjunction and adjunction. The former is used for coordinating conjunctions (e.g. *and*, *or*, *but*) and the arithmetic operators and their lexicalizations such as *plus* (hence &+). JG adjunction joins verbs or prepositions to their objects and phrases to their subjects.

The JG subjunction operator is perhaps the most flexible of the junction operators. Subjunction is used for determiners, quantifiers, complements, relative structures

TAG+7: Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms.
May 20-22, 2004, Vancouver, BC, CA.
Pages 163-170.

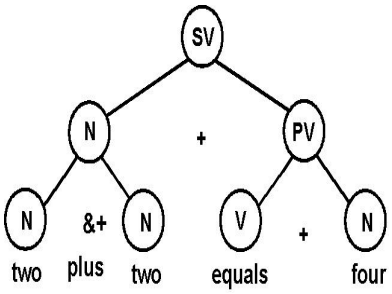


Figure 1: A binary application of the JG conjunction operator for the sentence *Two plus/and two equals four.* The SV node is a (verbal) sentence and the PV node is a (verbal) predicate.

and modifiers. It can also be subdivided into specialized operators that show the direction of information processing in the sentence. For an illustration, consider the following sentence:

(1) *It surprised me that the three children ate the vegetables that I cooked.*

and its associated JG concept tree in Figure 2. The word *three* selects three elements of the class *children*, and the junction has a single dash to the right in the direction of the noun class node. The determiner *the* retrieves a specific discourse-salient instance of three children. This retrieval function is shown by the equal sign to the left of the junction pointing to the determiner.

Figure 2 illustrates a second basic premise of Junction Grammar. JG allows for multiple tree structures unlike other theories that associate all trees structures under a single root. JG keeps complement structures such as *... that the children ate the vegetables ...* in the same tree structure with the subjunction operator, thus providing interoperability between sentence and noun. Using interjunction, JG allows for multiple intersecting trees for modifiers and relative structures via links, thus avoiding the need for empty categories and allowing a distinct contrast between complements and modifiers.

A third basic premise of JG is that trees can be ordered and lexicalized using flexible traversals guided by language-specific lexical ordering and hiatus (word deletion) rules. A JG concept tree represents syntactic/semantic information about the utterance that defines the syntactic and semantic relationships among the basic word concepts. The "goto" instructions and circled numbers in Figure 2 show lexical ordering rules that differ from the default left-to-right depth-first traversal algorithm. The complement clause *... that the three children ate the vegetables ...* is co-referent with the pronoun *it*, but the traversal of this clause is delayed until after the main clause is processed. Another ordering of the same sentence and same tree structure without lexicalizing the pronoun *it*, could be *That the three children ate the vegetables surprised me*. The JG tree would not change for that ordering, except that the pronoun *it* would be annotated as hiatused (another language-specific lexical rule in JG) by including parentheses.

The traversal processing for the trees in Figure 2 would thus be as follows:

1. Start processing at main SV node at top of figure.
2. Process left-to-right order to the subject N node and output *it* at the first N node of the rule $N * SV = N$.
3. Order discontinuously the complement $N * SV$ structure by going to ordering point #1, just above the PV node.
4. Process the PV left-to-right and output *surprised me*.
5. Now return to the unprocessed nodes in the subject by going to ordering point #2, by the embedded SV.
6. Order the SV left-to-right and output words *... that the three children ate the vegetables that ....* The relative pronoun *that* now triggers the processing of the subordinate SV structure. After going to the linked node, go to ordering point #3 at the top of the relative clause SV.
7. Process the relative clause in the default left-to-right order and output *I cooked*. The relative pronoun has already been marked as processed and at this point the entire tree has been processed.

In fact, the concept tree traversal specified above does not directly produce an output word string. Instead, following another foundational principle of JG, some concept tree information is recast into an articulation tree. The flexible traversal described above constructs such a tree, which encodes prosodic, phonological, and graphical information necessary for spoken or written language production. The basic JG operators in an articulation tree specify and relate breath groups and suprasegmental information; its nodes are of category H or W (for prosodic and lexical content respectively). Figure 3 shows concept and articulation trees for the following sentence:

(2) *Which job has Sally declined?*

Relevant traversal processing for the concept tree in Figure 3(a) is as follows, where lexicalization means mapping content appropriately into the articulation tree:

1. Start processing at the main SV node.
2. The ordering rule attached to the top node checks below the SV for a lexical entry with a $< +WH >$

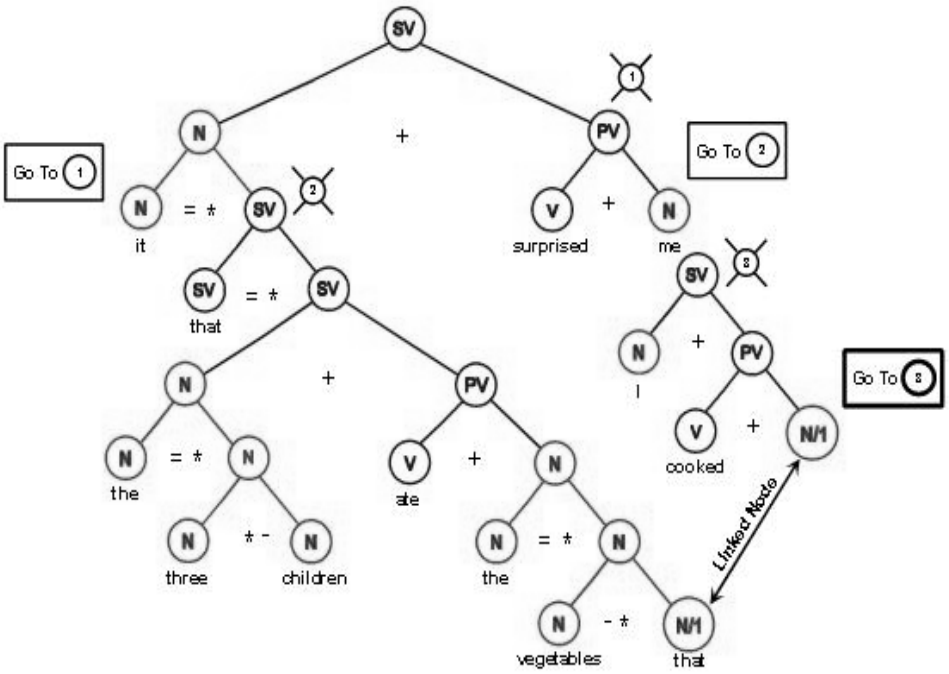


Figure 2: JG subjunction, adjunction and interjunction: a sentence with a relative clause shared across matrix and dependent clauses (via subjunction and adjunction respectively). Lexical ordering is also specified.

feature attached. Therefore, go to the N node dominating the N node with the *Which* lexical entry labeled with the ordering point #1.

3. Lexicalize *Which job* and then return back up the tree go ordering point #2.

4. In Junction Grammar trees the "modalizer" extending to the left of the PV level is the point where auxiliary verbs such as *has* are lexicalized.

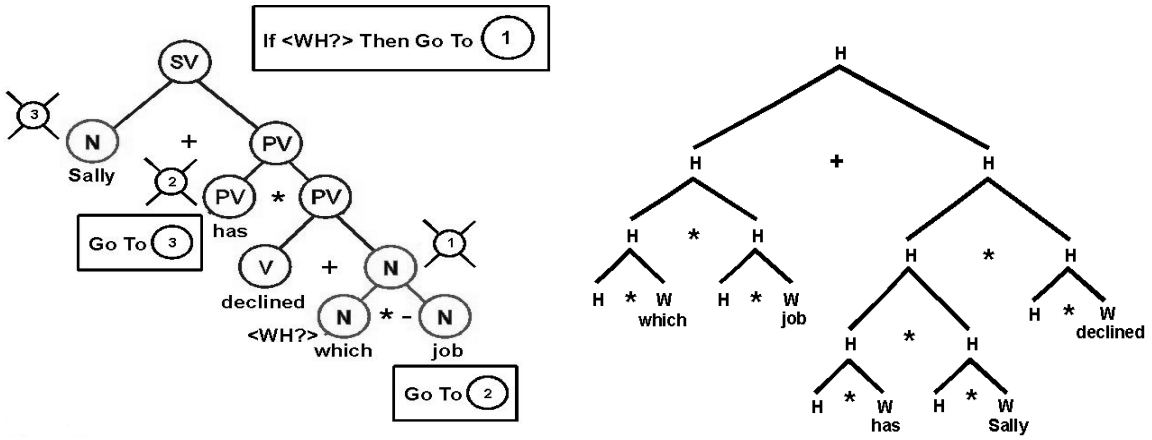
5. Go to ordering point #3 and continue using normal left-to-right order to finish processing the rest of the tree that has not yet been visited. *Sally declined* is lexicalized.

In combination with the basic junction operators discussed earlier, the basic JG rules and constraints are able to describe a wide variety of linguistic structures. Figure 4 shows a table of possible JG linguistic structures for a portion of the interjunction general rule template $X * X/1 = X$ and $Y + X/1 = PY$ or $PZ + X/1 = SZ$, where X varies over N, V, PV, and SV and Y varies over V and P and Z varies over V, A, P, and N.

This sketch of part of the JG theory focuses on only two of the four possible levels; consideration of the other levels is beyond the scope of this paper. It should be noted that JG parsers and/or generators have been implemented for a wide variety of languages including English, French, Spanish, German, Portuguese, Russian, Chinese, and Japanese.

2.2 Overview of TAG

Tree adjoining grammars (TAG) have provided a theoretical framework for linguistic description and natural language processing that has been shown to be superior to simply using rules of a context free grammar (CFG) due in large part to the extended context or "domain of locality" that TAG provides (Abeillé and Rambow, 2000; XTAG Research Group, 2001). With its lexicalized nature and its detailed syntax/semantics interface, LTAG (Lexical TAG) facilitates a straightforward representation of important data such as subcategorization frames for verb classes used in parsing. Appropriate usage of lexical entries is formalized by LTAG trees that not only derive a standard surface level parse tree but also a derivation tree that represents semantic and thematic role relationships for the sentence. Enhancements to TAG have included multicomponent TAG (MC-TAG) that allows for simultaneous linked operations on two or more trees into the derived tree to successfully derive parse trees for examples like "picture-NP extraction". To address the many com-



(a) (b)

Figure 3: Two JG trees for a sentence: the JG concept tree3(a), which is traversed to create the articulation tree 3(b).

binations of orders possible in languages such as German, a free order TAG (FO-TAG) variation was developed that did not force a strict left-to-right processing of all nodes of the derived tree (Becker et al., 1991; Rambow and Lee, 1994) Further enhancements of MC-TAG have included VMC-TAG that allows for non-simultaneous adjunction of multiple component trees to better describe free word order languages (Rambow and Lee, 1994). D-Tree Grammars (DTG) adjusted the basic TAG operations of substitution and adjunction to become subsertion and sister adjunction in order to separate complementation and modification operations and correct inaccuracies in the TAG derivation tree for topicization and to handle word order for wh- extraction sentences for Kashmiri (Rambow et al., 1995). In machine translation applications, synchronous TAG (S-TAG) is used to represent linked source and target language sets of trees that represent required transfer operations while translating between the two languages (Harbusch and Poller, 2000).

3 Comparative analysis

The most obvious similarity between JG and TAG is their use of multiple tree structures. TAG initial and auxiliary trees can define basic lexical options such as subcategorization frames for verbs along with their allowed auxiliary verbs and modifiers. JG represents modifier structures using subordinate tree structures. However, while the output of a TAG derivation is a standard tree diagram for a given sentence plus an associated derivation tree, the JG trees represent the final syntax/semantic representation of the sentence. JG applications have created JG trees by processing a grammar of allowed JG rules plus language-specific lexical ordering and other algorithms. Even though the JG interjunction rule pairs give some expanded domain of locality over single rules, JG has domain of locality limitations similar to context free grammars (CFG) and could greatly benefit from the mildly context sensitive grammar advantages that an LTAG approach could provide. TAG adjunction constraints and feature information are similar to JG lexical features that select basic JG rule groups via algorithms that might apply, for example, for a specific verb family. However, JG applications have relied more on specialized programming accompanying basic JG rules rather than being able to use forests of linked trees to implement lexicalized applications such as what a TAG approach would provide.

Because JG separates the syntax/semantics representation from the ordered words ready to articulate, many of the complexities of TAG trees can be simplified. The JG approach does put language-specific ordering and other lexical rules into algorithms that operate on the JG syntax/semantics trees, a deliberate tradeoff for not describing both syntax and order directly in tree structures. However, as the various TAG and variant systems have been developed, amazingly complex trees are needed to allow for all of the possible variations in word order for German, Korean or Kashmiri. A similar set of trees for JG parallel with a MC-TAG or DTG system would not require explicit encoding of multiple word order variations and hence the number of trees would be reduced.

Figure 5 summarizes some of these similarities and differences between JG and TAG.

3.1 JG-TAG

Because of their similarities, TAG and JG could conceivably be combined into a new JG-TAG approach. A parser

Subjunction rule in main tree	Adjunction rule in subordinate tree	Sample output text
N ∗ N/l = N	V + N/l = PV	The mouse that the cat chased (got away).
N ∗ N/l = N	PV + N/l = SV	The cat that chased the mouse (lives next door).
N ∗ N/l = N	PA + N/l = SA	The tall elephant . . .
N ∗ N/l = N	P + N/l = PP	The ladder upon which (I was standing) . . .
N ∗ N/l = N	PP + N/l = SP	The boy from Atlanta . . .
N ∗ N/l = N	PN + N/l = SN	Harry, the class clown, . . .
V ∗ V/l = V	PA + V/l = SA	He looked up (the number).
V ∗ V/l = V	PP + V/l = SP	(He) dropped (the paint) into the burning cauldron.
PV ∗ PV/l = PV	PA + PV/l = SA	(He) went quickly (into the city).
PV ∗ PV/l = PV	PP + PV/l = SP	(The city) needed water for survival.
SV ∗ SV/l = SV	PA + SV/l = SA	Unfortunately, . . .
SV ∗ SV/l = SV	PP + SV/l = SP	Without a doubt, . . .

Figure 4: Some basic JG relationships via interjunction; nodes are linked via the /1 annotation.

Multiple linked trees:

- JG: syntax/semantics tree representation(s) for a sentence
- TAG: grammar is represented in multiple trees but output is single parse tree plus derivation tree

Expanded domain of locality:

- JG: interjunction rule pairs but still context-free
- TAG: initial and linked auxiliary trees with adjunction constraints, mildly context sensitive with LTAG

Syntax, semantics and language-specific ordering:

- JG: separation of syntax/semantics from lexical ordering; more complex algorithms attached to rules but with simplified trees
- TAG: variations on TAG for complex word order but more complex and numerous trees required

Figure 5: JG and TAG: contrasts involving similar features.

could then be developed following TAG's XTAG model. Because JG representations have separate trees (Lytle, 1979) for syntax/semantics vs. ordered output words, such a system would be simplified from standard XTAG with fewer trees needed for a complete grammar.

The first enhancement needed to create a JG-TAG system would be to attach a junction operator to each nonterminal node. We will represent this junction attached right to the end of the node label. Because the subjunction operator is the "∗" character, JG-TAG would also need to change the foot node sign from a "∗" to a "#". Even though a subjunction operator would not be attached to a foot node, we will also propose another use of the "#" in another enhancement. Generalizing junctions allow for creation of an arbitrary number of conjuncts without spawning new nonterminal nodes. Figure 6 shows how *bears* could be added to the conjunction rule for *lions and tigers*. In this case the auxiliary tree would use the foot node indicator "#" attached to the non-terminal N node. The output tree shows the result of this n-ary adjunction capability. The lexicalization of *and* in these sentences is assured by the conjunction operator for the noun nonterminal node.

Another enhancement for JG-TAG is to allow for subordinate tree structures and their creation and processing. This is where the capabilities of MC-TAG are essential. One auxiliary tree would create the relative pronoun node on the side of the main clause and the other auxiliary tree would create the mirror node on the subordinate relative clause side. To avoid confusion with subscripts that are used in trace nodes in standard theory trees, we propose that the link number between these mirrored linked nodes be a superscript. Figure 7 illustrates this process with the sample working tree for the sentence *The cat caught a mouse*. The two MC-TAG trees would operate on the working tree by first adjoining with the node for cat and

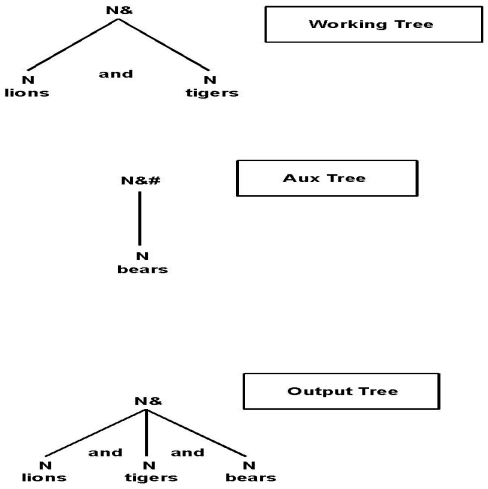


Figure 6: Adjoining trees in JG-TAG to form a three-node conjunction concept tree.

then creating the subordinate relative clause, linked by the superscript "1" with the main tree. The tree that adjoins into the main clause to create the $N * N = N$ rule would be defined as an auxiliary tree and the subordinate clause would be defined as an elementary tree in order to create the new tree. The SV node of the main clause would also be marked as the starting point for processing the resulting JG tree forest. The overall process would be as follows:

1. The left JG-TAG auxiliary tree will left-adjoin with the *cat* node to create a $N * N/1 = N$ subtree with *cat* and *that* associated with the nodes.

2. The right JG-TAG elementary tree *that/1 the dog chased* will then be added to the working tree forest and the relative pronoun noun node with the superscript link stays linked with the N/1 node in the main working tree.

3. The resulting new working tree has the main tree, where processing of the tree would begin, marked as the start tree (*the cat that/1 caught a mouse*).

4. The subordinate tree is accessed during tree traversal by going from the main clause mirrored link to the subordinate clause.

3.2 JG-TAG: prospects and challenges

Early NLP applications of JG used junction rules plus specialized programming to examine rule contexts for triggering language-specific transfer or lexical rules. Currently JG tree processing programs are limited to a proprietary control language used inside recent applications. A JG-TAG system would allow a standard XTAG-like system to be developed that could provide a parsing capability for JG trees, allowing wider access and easier comparisons with existing systems using other theories.

Another benefit of such a system would be its ability to represent a greater portion of the grammar of a language with fewer TAG trees. For example, if the JG concept tree in Figure 3 were represented in JG-TAG, only one tree would be needed to cover both cases where the substituted noun nodes include a "+WH" feature or not. This same tree could also work for nonstandard orderings of an SVO sentence as OSV.

A JG-TAG system would provide an excellent framework to represent subcategorization frames for different verb classes using supertag trees. TAG has always excelled in providing context sensitivity to a basic rule system and a lexicalized JG grammar implementation would allow JG structures that have previously been represented programmatically to be described in a more easily visualized and maintainable data structure format. The verb class JG-TAG trees would also simplify the lexical rules by attaching them to specific verbs and allowing them to be limited to the context of a specific verb.

One of the exercises in creating such a system would involve the format of lexical rules that would be attached to the JG-TAG trees. Each JG-like rule in the tree specifies left-to-right, right-to-left or discontinuous ordering. Recall that the JG approach involves in-situ wh-elements and a specific traversal order without creating target nodes for movement. Thus the algorithm for deciding traversal would reflect, but not implement, movement. The documentation and implementation papers for the JG ordering algorithms and transfer language used in an early machine translation project could be a good starting point for a JG-TAG system (Melby 1974, Gessel 1975).

Another challenge would be matching and using features attached to JG nodes with the TAG feature capabilities. TAG unification features that prevent more than one tense-bearing verb to be attached usually would be implemented by JG lexical agreement rules. However, the feature unification approach from TAG provides a straightforward manner to keep track of main and auxiliary verbs and their inflections as a sentence is created from the tree.

Mandatory, optional and null adjunction constraints allow the relationships between the various TAG tree sets to be carefully defined, linked together and maintained. Expert rule systems generally need these kinds of constraints in order to assure tractable development and maintenance. These same capabilities would be very advantageous to link together JG tree fragments that would define a working grammar for a particular language.

The power of the MC-TAG trees that encapsulate semantic relationships would then output not just a surface ordered derived tree but an order-independent syntax/semantics representation less dependent on the

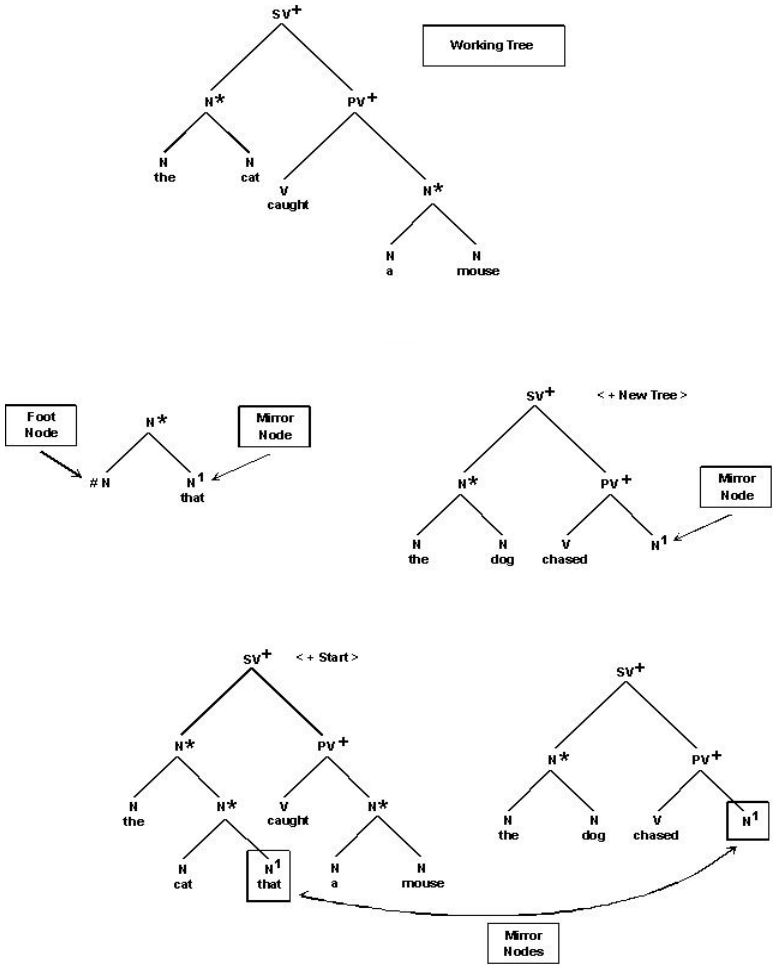


Figure 7: JG-TAG MC trees for a relative clause; the top working tree is operated on by the JG-TAG multicomponent tree that would attach the relative structure at the appropriate node and subordinate tree. The N node is structure-shared between the trees; superscripts specify inter-tree links.

derivation tree for semantic relationships. The JG trees are not at as low a semantic level as the derivation tree but provide structure related to the original utterance (e.g. active vs. passive) and are very rich in specific syntax and semantic relationships (e.g. themes and verb classes with thematic roles (Millett, 1975)) between the concepts of the utterance. Comparative and quantifier structures have a particularly rich semantic structure in JG (Lytle 1985) and a JG-TAG system could facilitate comparison of the capabilities of a JG-based text-understanding application to other standard approaches. A JG-TAG system could also provide a standardized application and coding framework for using Junction Grammar.

4 Conclusions

As TAG formalisms have been applied to natural languages, their advantages over context-free phrase structure rules have become more apparent. Many useful refinements to the basic TAG formalism have supported a wide variety of structures. Meanwhile JG embodies rather different assumptions than do traditional theories: a separation of linguistic data via conceptual and articulation trees, junction operators on non-terminal nodes, multiple-linked tree structures, and flexible traversal of lexical rules. The appreciable overlap of approaches with TAG and JG has prompted this discussion on combining the benefits of both theoretical systems to represent and process Junction Grammar trees. The advantages of the mildly context sensitive lexical JG-TAG system proposed in this paper can expand the domain of locality for JG trees, simplify lexical rules by attaching them to supertag class trees and draw on the extensive NLP experience using TAG based systems to benefit JG. TAG could likely also benefit from junctions, ordering, and multiple tree enhancements from Junction Grammar.

Acknowledgement We wish to thank Eldon Lytle for comments on drafts of this paper. All errors or inaccuracies are the authors'.

References

Anne Abeillé and Owen Rambow. 2000. Tree Adjoining Grammar: An overview. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing*, pages 1–68. CLSI Publications.

T. Becker, A.K. Joshi, and O. Rambow. 1991. Long distance scrambling and tree adjoining grammars. In *EACL-91: Papers presented to the 5th Conference of the European Chapter of the Association for Computational Linguistics*, Berlin.

Floyd H. Billings. 1972. Proposals for ordering well-formed syntactical statements. Master's thesis, Brigham Young University.

Hunter M. Breland and Eldon G. Lytle, 1990. *Computer-assisted writing skill assessment using WordMAP*. American Educational Research Association and the National Council on Measurement in Education, Boston, MA. ERIC Document Reproduction Service No. ED 317 586.

Brian Gessel. 1975. The formulation and computer adaption of synthesis grammars. Master's thesis, Brigham Young University.

Daryl K. Gibb. 1970. An application to mechanical translation of a variable recursive algorithm based on the operations of union and intersection. Master's thesis, Brigham Young University.

Karin Harbusch and Peter Poller. 2000. Non-isomorphic synchronous tags. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing*, pages 147–166. CSLI Publications.

Eldon G. Lytle and N. C. Mathews, 1986. *Field Test of the WordMAP Writing Aids System*. Panaca, NV, May. Official publication of the Lincoln County School District.

Eldon G. Lytle. 1971. *Structural Derivation in Russian*. Ph.D. thesis, University of Illinois.

Eldon G. Lytle. 1974. *A Grammar of subordinate structures in English*. Mouton, The Hague.

Eldon G. Lytle. 1975. Junction Grammar as a base for natural language processes. *American Journal of Computational Linguistics*. Microfiche no. 26.

Eldon G. Lytle. 1979. Junction Grammar: Theory and application. In William C. McCormack and Herbert J. Izzo, editors, *Sixth LACUS Forum*, pages 305–343. Hornbeam Press.

Eldon G. Lytle. 1985. Come on up. In Adam Makkai and Alan K. Melby, editors, *Linguistics and Philosophy: Essays in honor of Rulon S. Wells*, volume 42 of *Amsterdam studies in the theory and history of linguistic science: Current Issues in Linguistic Theory (IV)*. John Benjamins, Amsterdam.

Alan K. Melby. 1974. Formulating and testing syntactic transfers. Master's thesis, Brigham Young University.

Alan K. Melby. 1976. *Computer Generated Intonation in Synthetic Speech*. Ph.D. thesis, Brigham Young University.

Alan K. Melby. 1978. Design and implementation of a machine-assisted translation system. In *Proceedings of the Seventh International Conference on Computational Linguistics*, Bergen, Norway.

Alan K. Melby. 1985. Generalization and prediction of syntactic patterns in Junction Grammar. In Adam Makkai and Alan K. Melby, editors, *Linguistics and Philosophy: Essays in honor of Rulon S. Wells*, volume 42 of *Amsterdam studies in the theory and history of linguistic science: Current Issues in Linguistic Theory (IV)*. John Benjamins, Amsterdam.

Ronald P. Millett and Eldon G. Lytle. 2004. Language included: Book 1: Sentence pictures, picturing grammar, picturing sentences. eBook format.

Ronald P. Millett. 1975. Junction Grammar verb classes in Spanish. Technical report, Brigham Young Univeristy. Junction Grammar Papers.

Ronald P. Millett. 1976. A pitch contour generating algorithm based on a Junction Grammar linguistic model. Technical report, Brigham Young Univeristy. Junction Grammar Papers.

Royden S. Olsen and David M. Tuttle. 1973. The effect of 'language trees' on the acquisition of generative capacity in a second language. Technical report, Brigham Young Univeristy. Junction Grammar Papers.

Owen Rambow and Y. S. Lee. 1994. Word order variation and tree-adjoining grammars. *Computational Intelligence*, 10:386–400.

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-tree grammars. In *Proceedings of ACL-95*.

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