Tree-Adjoining Grammars: A Model of Syntax Representation

Luke Palmer

December 6, 2005

In 1957, Noam Chomsky introduced four formal classes of language with which to study human discourse: *regular* languages, *context-free* languages, *context-sensitive* languages, and finally *unrestricted* or *recursively enumerable* languages. Each class in this list is a subset of the next. Since their introduction, these languages have been widely studied in the contexts of mathematics, linguistics, and computer science.

This article will present another fairly mature class of languages that has remained out of sight for everyone but computational linguists. This is the class of languages generated by the *tree-adjoining grammars*; the so-called *mildly context-sensitive languages*. This class is important as it allows us to *lexicalize* the grammars for a natural language. To lexicalize a grammar means to associate exactly one word to each reduction rule.

In Section 1, we introduce the widely-studied context-free grammars, and present their use in natural language as well as their shortcomings which led to the introduction of tree-rewriting grammars. Section 2 describes tree-

rewriting grammars with a focus on the tree-adjoining grammars. Finally, Section 3 presents a link between the regular, context-free, and tree-adjoining classes (the "language hierarchy") and the process by which children acquire language.

1 The Rise and Fall of Free Context

A great deal of research has gone into the context-free languages. They were first proposed in order to describe human language, but they quickly took off in the area of computer science, used to process programming language syntax, network protocols, and natural language-like commands for games. They have also been used in natural language processing, but more powerful techniques have become more popular in recent years. Nonetheless, we will focus on their applicability to natural language, which will give us a good idea why they had to be phased out.

Informally, a context-free grammar is a set of rules that takes "rewritable" symbols eventually into sequences of words in the target language. For example, this is a small version of a common context-free grammar:

S	Start symbol
NP VP	Rule (1)
the dog VP	Rule (3)
the dog V NP	Rule (2)
the dog V the cat	Rule (4)
the dog chased the cat	Rule (5)

Figure 1: Derivation of "the dog chased the cat"

$$S \rightarrow NP\ VP \tag{1}$$

$$VP \rightarrow V\ NP \tag{2}$$

$$NP \rightarrow \text{“the dog”} \tag{3}$$

$$NP \rightarrow \text{“the cat”} \tag{4}$$

$$V \rightarrow \text{“chased”} \tag{5}$$

The capital letters represent rewritable symbols[1], and the things in quotes represent words in the target language. We can see how the string "the dog chased the cat" was generated from S (the so-called *start symbol*) in Figure 1. The order in which the rewrites are made does not matter (in the fourth step I could just as well have used rule (5) to attain "the dog chased NP"). Rules may also be applied more than once during a derivation (in the fourth step I could have used rule (3) again to attain "the dog V the dog").

It is not very interesting to use a grammar to generate sentences. Most of

[1] S: Sentence; NP: Noun Phrase; VP: Verb Phrase; V: Verb.

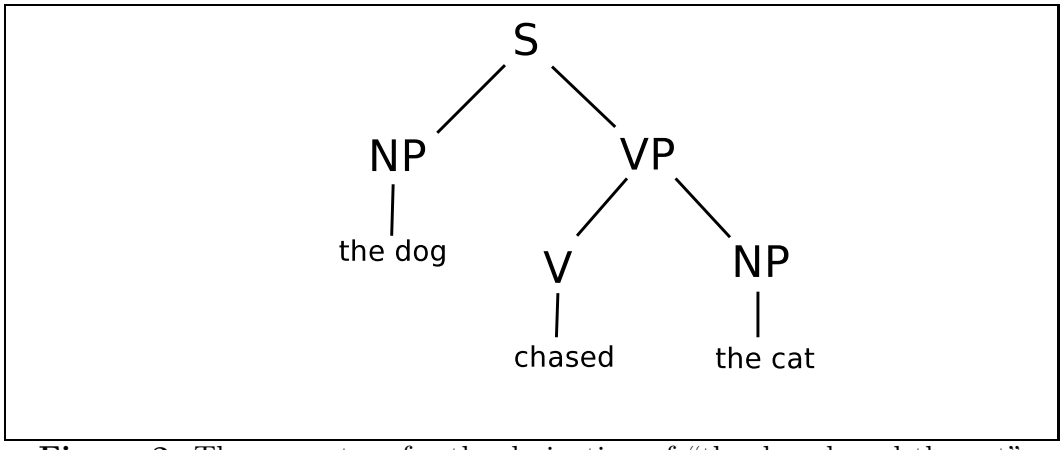

Figure 2: The parse tree for the derivation of "the dog chased the cat"

the time you will get nonsense[2]. It is also possible to use a grammar to *match* sentences; i.e. to see if a particular sentence could have been generated by a given grammar. But that isn't very interesting either, by itself. What is interesting when you check that a sentence is generated by a grammar is *how* it was generated. This process creates a *parse tree* of the sentence, which classifies phrases and relates them to each other in a logical way. The parse tree plays an even bigger role in the tree-rewriting grammars of Section 2. The parse tree for the derivation in Figure 1 can be seen in Figure 2.

After experiencing great success in the theory of programming languages, context-free grammars began to show their weaknesses in natural language processing. For instance, they fail at one of the most fundamental underlying themes in language: agreement. To ensure that the subject agrees with the verb in number, you must define two very similar VPs: VP_{sg} (singular) and VP_{pl} (plural). To ensure that the subject agrees with the verb in person,

[2]But you can be assured that it will be syntactically correct nonsense!

you must define three very similar *VP*s, and to combine the two properties, you *multiply* the definitions, resulting in *six* almost-duplicated productions. This is clearly getting out of hand.

It was disputed and unknown at the time whether all natural languages were actually context-free. In 1985, Shieber[9] showed convincing evidence that Swiss German was not context-free. This increased the pressure on academics to come up with a better representation of natural syntax. Fortunately for them, one had already been around for ten years.

2 Tree-Rewriting Grammars

Context-free grammars were all about rewriting strings to (or from) a single start symbol, and to create a parse tree in the process. Tree-rewriting grammars instead start with one or more start trees and rewrite them using rules until one of them matches the input.

The first form of tree-rewriting grammar was the *tree substitution grammar*. Here, the start trees contained *substitutable nodes*, which were childless nodes marked with a root type (for example, *NP*). New trees could be derived from old ones by substituting a tree with the given root type for that node. This provided the people who constructed grammars more control over the derived trees, and it was a generally encouraging development until it was shown that the tree substitution grammars derive the same set of languages as the context-free grammars.

Section 2.1 describes the formalism of tree-adjoining grammars, and section 2.2 describes the integration of features, a method for controlling what kinds of substitutions can be made.

2.1 Tree-Adjoining Grammars

The tree substitution grammars were modified to allow an additional operation, namely *adjunction*, which formed the *tree-adjoining grammars* (TAGs) [6]. In a tree-adjoining grammar, you again specify a set of initial trees, which correspond roughly to the different general forms a sentence can take on. An example base tree for English is shown in Figure 3. Then you specify *auxiliary trees*, which are a bit tricky. An auxiliary substitution takes a subtree S and substitutes a subtree T, making S a subtree of T. It is best illustrated by example. Figure 4 shows an auxiliary substitution on the base tree in Figure 3. The shown substitution allows the string "I taught him a lesson" to derive "I really taught him a lesson".

TAGs are (finally!) not equivalent to the context-free grammars. They are strictly more powerful, generating the set of "mildly context-sensitive languages". Chomsky's context-sensitive grammars are still more powerful. However, TAGs have had much more success in practice than the context-sensitive grammars because the formalism is easier to work with. It is hard to intuit the generated structure of a context-sensitive grammar just by looking at it, while it is hard to miss the generated structure of a TAG. TAGs also readily provide a way to encode semantic information.

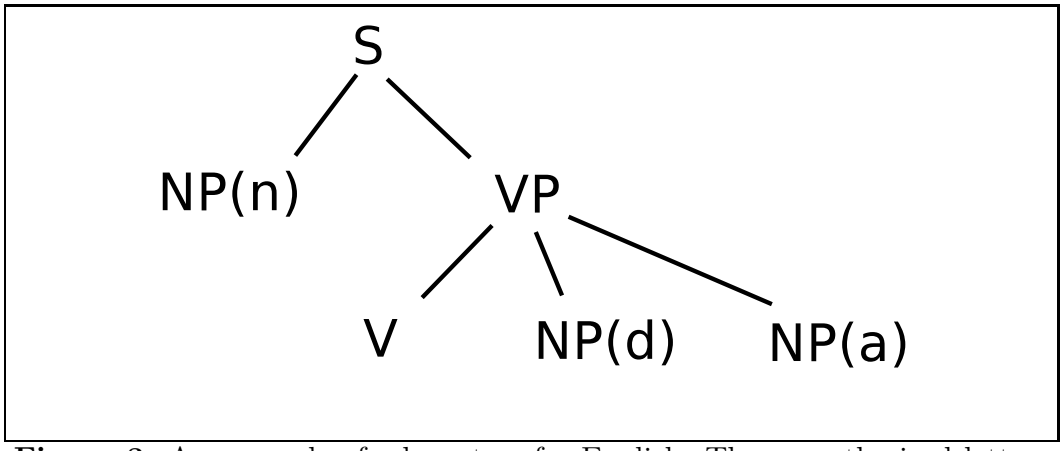

Figure 3: An example of a base tree for English. The parenthesized letters after NP refer to the case that declined pronouns would take on: *n* for *nominative*, eg. "**I** gave John the ball"; *d* for *dative*, eg. "John gave **me** the ball"; *a* for *accusative*, eg. "John gave **me** to the asylum".

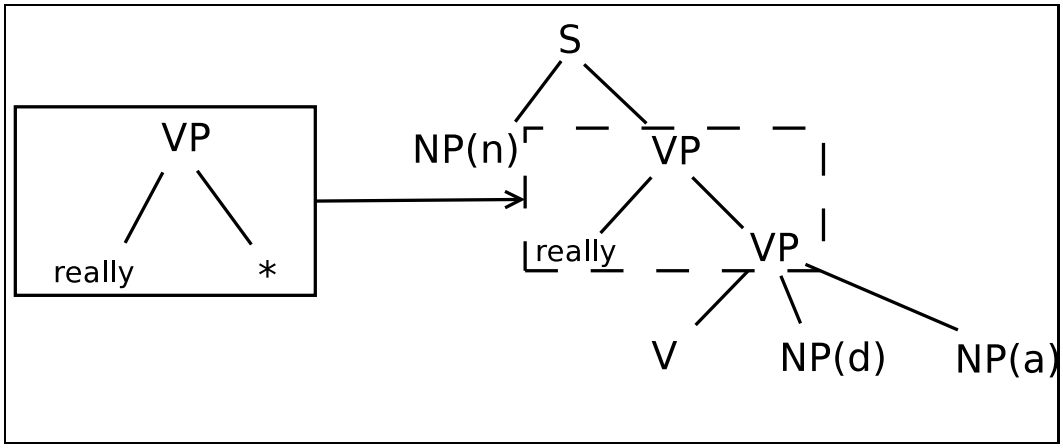

Figure 4: An auxiliary substitution on the base tree shown in Figure 3. The auxiliary tree is on the left, the resulting tree is on the right. The star in the auxiliary tree represents where to put the subtree that is being replaced by this substitution.

The tree-adjoining grammars were ultimately introduced in order to lexicalize any specified grammar. That is, it is desirable to associate one rule with each lexeme[3]. This is because modern linguistic theory unifies to some degree the lexicon and the syntax, by treating each lexeme as its meaning together with a "minisyntax" for its usage.

2.2 Feature-based TAGs

TAGs have some of the same problems that were pointed out for context-free grammars in section 1. In order to allow the sentence "John has seen the ball" but disallow "*John seen the ball", node types will unnecessarily multiply. Two solutions were proposed for this: *Constrained TAGs* and *Feature-based TAGs*.

Constrained TAGs allow you to specify a finite set of adjunctions that can occur at a given node, rather than the usual unconstrained operation where any matching adjunction can occur. These haven't been widely successful, and the engineer in me feels uneasy about them, wanting things to be extensible. You don't want to have to go back through all your rules when you add a new auxiliary tree to see if it should or should not be allowed to be substituted in each place.

Feature-based TAGs (FTAGs), introduced by Vijay-Shanker in 1987 [10], instead allow you to associate two *feature sets* with each of the nodes, which

[3]A lexeme is one unit of meaning, usually (but not always) a single word. For example, "to give" (and its various inflected forms) is one lexeme, but "to give up" is another.

must unify with each other at every position in the derived tree. These are commonly called the *top* and *bottom* features, which encode properties of the structure above and below, respectively, the node in question. When performing an adjunction of an auxiliary tree A for a source subtree S, S's top feature must unify with A's root's top feature, and S's bottom feature must unify with A's *'s bottom feature. Another way of saying this is that the top feature says what the tree is expecting in that position, and the bottom feature says what it actually has. Adjunctions must be made in order to make those two things the same.

This formalism allows us to represent the "seen" example above simply, as shown in Figure 5. The feature on VP in this example is `tense` which represents whether it is a tensed clause (i.e. has a subject). The initial tree on the left expects a tensed VP, and provides an un-tensed VP on which it must be based. The reader should verify that all the features unify by adjoining the auxiliary tree on the right to the VP on the left. The stars in the auxiliary tree's features mean "anything" or "don't care".

3 Child Language Acquisition

Now that we have the mathematical formalisms behind these various classes of languages, we can see how they relate to the human mind. What follows is a summary, in the layman's terms established in this introduction, of Robert Frank in [4].

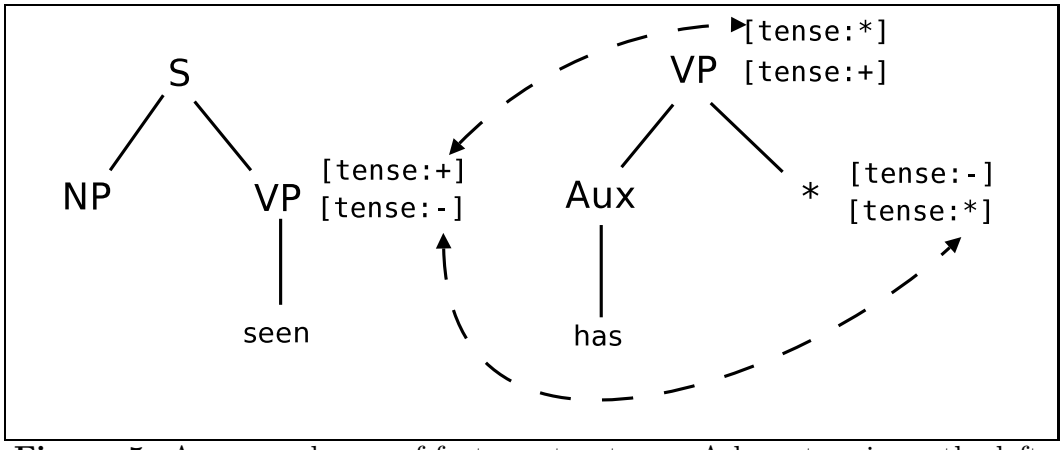

Figure 5: An example use of feature structures. A base tree is on the left, which is not syntactically correct because the `tense` features do not unify on *VP*. An auxiliary tree is on the right, which can be adjoined to the *VP* on the left to make it unify.

Frank points out a natural ordering in children's language acquisition, that utterance of sentences of type 1 below precedes utterance of type two, which in turn precedes type three:

1. They're taking a vacuum cleaner to wipe and puppy dog's running. (Conjunction)

2. Tell Iris that I wet my bed. (Complementation)

3. I'm calling the man who fixes the door. (Relativization)

The same sort of ordering applies to children's assignment of correct meaning to the following sentences [2]:

1. The sheep that kissed the monkey kissed the rabbit.

2. Cookie Monster tells Grover to jump over the fence[4].

3. Cookie Monster touches Grover after jumping over the fence[5].

We would like to understand why this ordering occurs. It had been argued previously that children hadn't yet acquired the proper grammatical structures, but Frank argues instead that they are not able to perform certain types of computations. Recall the tree substitution grammars (TSGs) introduced in Section 2. If you restrict those grammars not to allow recursion, but instead simply conjunction (forming the NTSGs), then we can explain the first type of sentences acquired in learning. If the base trees are all constructed in the manner of Figure 6, the misunderstandings in comprehension above can be understood as children's use of "that" as a synonym for "and" (which was shown by [2]). The reader should verify that the sentence "John has eaten an apple and Fred has eaten peaches and a candy bar" fits into this structure, but that "John was saying he has eaten an apple" would require recursion.

Allowing recursion (the TSGs), we arrive at the second type of sentences here, but prepositional phrases still cannot be handled[6]. Finally, allowing adjunctions (the TAGs), we can generate an adult language. Frank's showed

[4]Children who could comprehend the previous sentence would act out that Cookie Monster was the one jumping

[5]A similar misunderstanding happened here, where children thought that Grover was jumping over the fence. Admittedly, this sentence could be interpreted ambiguously even by adults.

[6]Lexically, that is. Recall our requirement to associate exactly one lexeme with each tree.

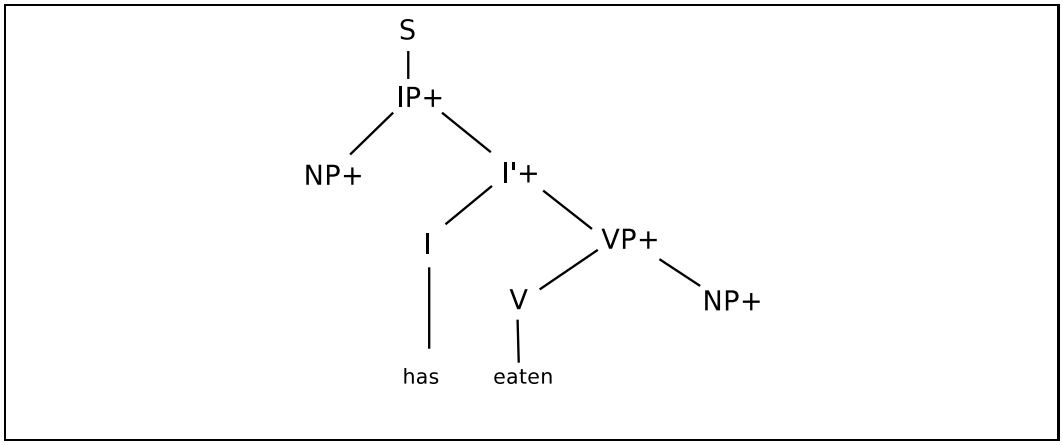

Figure 6: A base tree using conjunction rather than recursion. The plus after each symbol means that it can in fact be one or more such symbols joined with "and".

that the NTSGs are equivalent in generative capacity to the regular languages. Recall that TSGs are equivalent to the context-free languages and that TAGs form a subset of the context-sensitive language. Thus child language acquisition can be modeled by Chomsky's language hierarchy!

This is quite an incredible finding. It suggests that there is some mathematical machinery being developed in a child's brain as it matures, rather than just acquiring new vocabulary and word uses. This finding forms yet another argument for the pervasive lexicon; that is, if we drop the requirement to lexicalize the grammar, then we can represent most sentences as context-free, and many as regular, and there would be no such necessary ordering.

But how, then, do we explain the fact that Figure 6 is not lexicalized? Elman [3] explains that as children are learning, they treat combinations of

words as single lexical units. That is, "has-eaten" and "ate" are learned synonyms rather than inflections.

However, if such computational development accurately represents our process of learning language, it seems that our ability to utter and comprehend certain types of sentences would be quantized; all sentences of a certain type would emerge simultaneously. Though this has not been studied[7], it doesn't seem very likely. Frank conjectures that "the more powerful tree rewriting mechanisms are unavailable simply because they demand too much of the child's resources, [and] they could become available when other resources are freed." This claim is supported by Joshi's complexity metrics [7] for tree transformations.

Cognitive scientists keep finding more about the brain's ability to make generalizations. I claim that it is the linguistic center that is in charge of making such abstractions. Mathematics, the most abstract and symbolic field, is simply a transformation language. Before a mathematical idea is widely accepted, a suggestive and clear notation—even a linguistic metaphor—must be created for it. It is possible to think of all our abstract thought as computations similar to the tree transformations we have seen in this article. If so, then truly it is just the ability to use language that separates humans from the other animals on the Earth.

[7]Read: I am not aware of a study...

References

[1] Abeillé, Anne and Rambow, Owen. 2000. Tree Adjoining Grammar: An Overview. *CSLI Lecture Notes no. 107*: 1–68.

[2] Bloom, Lois, Margaret Lahey, Lois Hood, Karin Lifter, and Kathleen Fiess. 1980. Complex sentences: Acquisition of syntactic connectives and the semantics relations they encode. *Journal of Child Language 7*: 235–261.

[3] Elman, Jeffrey L. 1990. Finding Structure in Time. *Cognitive Science 14*: 179–211.

[4] Frank, Robert. 2000. From Regular to Context-Free to Mildly Context-Sensitive Systems: The Path of Child Language Acquisition. *CSLI Lecture Notes no. 107*: 101–120.

[5] Gardent, Claire and Parmentier, Yannick. 2005. Large Scale Semantic Construction for Tree Adjoining Grammars. *Logical Aspects of Computational Lingusitics 2005*: 131–146.

[6] Joshi, Aravind K., Leon Levy, and M. Takahashi. 1975. Tree Adjunct Grammars. *Journal of Computer and System Sciences 10*: 136–163.

[7] Joshi, Aravind K. Processing Crossed and Nested Dependencies: an Automaton Perspective on Psycholinguistic Results. *Language and Cognitive Processes 5*.

[8] Jurafsky, Daniel and Martin, James H. 2000. *Speech and Language Processing*. Prentice Hall, NJ.

[9] Shieber, Stuart B. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy 8*: 333–343.

[10] Vijay-Shanker, K. 1987. *A study of Tree Adjoining Grammars*. Doctoral dissertation, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, December.