Towards a Reduced Commitment, D-Theory Style TAG Parser

John Chen*
K. Vijay-Shanker
Department of Computer and Information Sciences, University of Delaware
Newark, DE 19716, USA
Email: {jchen,vijay}@cis.udel.edu

Abstract

Many traditional TAG parsers handle ambiguity by considering all of the possible choices as they unfold during parsing. In contrast, D-theory parsers cope with ambiguity by using underspecified descriptions of trees. This paper introduces a novel approach to parsing TAG, namely one that explores how D-theoretic notions may be applied to TAG parsing.

Combining the D-theoretic approach to TAG parsing as we do here raises new issues and problems. D-theoretic underspecification is used as a novel approach in the context of TAG parsing for delaying attachment decisions. Conversely, the use of TAG reveals the need for additional types of underspecification that have not been considered so far in the D-theoretic framework. These include combining sets of trees into their underspecified equivalents as well as underspecifying combinations of trees.

In this paper, we examine various issues that arise in this new approach to TAG parsing and present solutions to some of the problems. We also describe other issues which need to be resolved for this method of parsing to be implemented.

1 Introduction

Ambiguity is a central problem in the parsing of natural languages. Within the framework of TAG, a number of approaches have been adopted in trying to deal with this issue. Traditional approaches share the characteristic of exploring all of the choices in the face of ambiguity. To make such a scheme effective, techniques such as structure sharing and dynamic programming are commonly used. In contrast, we explore a different style of parsing that copes with ambiguity through the underspecification of tree structures.

This style of parsing deals with descriptions of trees instead of trees themselves. Notable examples of such parsers include [Marcus, Hindle, and Fleck, 1983], [Gorrell, 1995], and [Sturt and Crocker, 1996]. We refer to these as D-theory parsers. Following them, we construct an underspecified representation that captures a set of parses for the input seen so far.

The primitive of dominance is used in the descriptions that our parser manipulates. Its use in parsing has been pioneered by [Marcus, Hindle, and Fleck, 1983], and further investigated by the others such as those listed above. It has also been used within the confines of TAG elementary trees, as stated by [Vijay-Shanker, 1992]. Nevertheless, in our context of using domination to handle ambiguity in TAG parsing, we differ from both of these bodies of work. We differ from the latter because our use of domination is not limited to use within TAG elementary trees; it is crucially used here to underspecify ambiguity in the attachment of elementary trees. Thus the domination statements that our parser makes are domination statements between nodes in different elementary trees. We differ from prior work in D-theory parsing because our parser incorporates a specific grammatical formalism and grammar in order to specify syntactically well-formed structures. Moreover, it uses new kinds of representations that have not been considered in D-theory parsing in order to specify ambiguity.

This paper discusses several novel issues and presents our solutions to many of the issues that arise in adopting the use of dominance to parsing TAGs. For example, there is considerable bookkeeping that is necessary to ensure whether a particular underspecified representation conforms to a possible legal TAG derivation. Furthermore, TAG's extended domain of locality presents a problem because it hinders the degree of underspecification

*Supported by NSF grants #GER-9354869 and #SBR-9710411.

that is necessary in order to delay disambiguation decisions. We therefore argue that sets of elementary trees need to be underspecified into single representations. Another problem is that the notion of standard referent, used in D-theory parsing as the default model of a given underspecified representation, needs to be reconceptualized in order to accommodate the novel use of domination that is proposed here.

This paper is organized as follows. We first consider some broad distinguishing characteristics of our parser in Section 2. Subsequently, we define some terminology in Section 3 as a preparatory for the following sections, which delve into various aspects of of our parser. Section 4 gives a high level outline of our parser. Section 5 explicates the mechanism whereby descriptions of trees are combined. Section 6 deals with computing the standard referent. Section 7 discusses the issue of combining a set elementary trees into a single representation. Finally, concluding remarks are given in Section 8.

2 Parsing Framework

Our parser draws on typical characteristics of D-theoretic parsers, such as [Marcus, Hindle, and Fleck, 1983] and [Gorrell, 1995]. Such characteristics, as well as their relation to our parser, are discussed in this section. A basic feature of D-theoretic parsers is that the input sentence is parsed strictly from left to right and a description is constructed that captures a set of parse trees for the input seen.

One of our goals, which is shared by many other D-theoretic parsers, is that of incremental interpretation. This means at each point in the parse, a partial semantic representation is constructed from the sentence prefix that has been input so far, as delineated by the underspecified representation.

The notion of standard referent, as discussed in [Marcus, Hindle, and Fleck, 1983] and [Rogers and Vijay-Shanker, 1996], is included in our parser. Recall that an underspecified representation specifies a set of parse trees, namely those trees which are consistent with every assertion that is embodied by the underspecified representation. The standard referent is that parse tree which is taken to be the default choice from the set of trees that are consistent with a particular underspecified representation. The need for a standard referent would arise from a requirement of incremental interpretation. However, here it is mainly used in determining how structures may be combined.

Our parser subscribes to the notion of informational monotonicity. This means that, during parsing, the assertions that the parser makes are considered to be indelible. This is important because, at any point during parsing, we find that not all of the attachment possibilities that may be expressed by a dominance based description language can be incorporated into the underspecified representation that our parser constructs. Certain ambiguities, for example, may not be expressible in our underspecified representation. Therefore, there is the possibility that one of the assertions that the parser makes will turn out to be incompatible with evidence provided by later input. In that situation, the parser is forced to retract that assertion, leading to backtracking. In the literature on D-theory parsers, it has been commonly conjectured that such backtracking correlates with garden path effects.

3 Terminology

Our terminology borrows from TAG and D-Tree Grammar literature. From TAG, the grammar consists of a set of lexicalized trees. The frontier of each elementary tree is composed of a *lexical anchor*; the other nodes on the frontier will either be *substitution* or *foot* nodes. An interior node is commonly considered to be constituted of a top and bottom feature structure. In contrast, we use the terminology introduced by [Vijay-Shanker, 1992] in that an interior node is considered as a pair of *quasi-nodes*. A domination link connects the *top* and *bottom* quasi-nodes. The path from the root to the lexical anchor of an elementary tree will be called the *trunk*.

We now define the terms component and lowering node. The notion of component is borrowed from D-Tree Grammars, as defined in [Rambow, Vijay-Shanker, and Weir, 1995]. A *component* is a maximal subpart of an underspecified representation that has the property that all of its nodes are connected by immediate domination links. Consider α_0 in Figure 1 for example. It consists of two components: the first contains the nodes S, NP_0, and the top VP quasi-node, while the second contains the bottom VP quasi-node (VP_0), the V node, and the NP_1 node. A *lowering node* is the root of a component. It can either be the root of an elementary tree or a bottom quasi-node. We call such nodes lowering nodes because in D-theory, adjunction or substitution at these

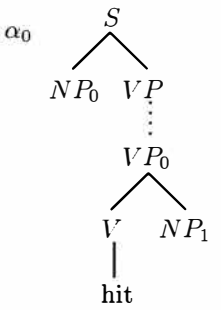

Figure 1: A lexicalized TAG tree with two components and two lowering nodes.

nodes would mean that these nodes would be lowered, i.e., later input causes some structure to be placed above them. The elementary tree α_0 contains two lowering nodes, S and VP_0.

4 Parsing Algorithm

With the terminology so defined, the disquisition of our parsing algorithm can now proceed. At any given time during parsing, we assume that there is a single *underspecified representation* R corresponding to the sentence prefix which has been seen so far. It describes a set of trees that are obtained by combinations of structures anchored by the input encountered. For now, we assume that R is treelike except that some of the nodes may be connected by domination links instead of immediate domination links. The immediate domination links arise only from the immediate domination links that appear in the composed TAG trees. The domination links arise from either those that appear in elementary TAG trees or between nodes in different trees. At the start of parsing, R consists of a single substitution node S.

A provisional description of the parsing process is outlined as follows.

- The next word is input. There will usually be more than one elementary tree corresponding to this word.
- Those elementary trees that are *structurally* incompatible with the current representation R are removed from consideration. This is addressed in Sections 5.1 and 5.2.
- The remaining elementary trees must be incorporated into the current representation R to produce the new underspecified representation. If only one elementary tree corresponding to the current input word remained after the second step's filter, then this would be straightforward. On the other hand, there may be cases where more than one tree remains. We propose to address the latter situation by grouping such sets of elementary trees into one underspecified representation. This would allow us to delay the choice of elementary tree until later while still being consistent with informational monotonicity. Nevertheless, it may still be the case that the remaining set of elementary trees corresponding to the current word cannot be grouped together. In this case, we would have to make a choice. We propose that other sources of information would need to be recruited in order to further constrain the set of viable elementary representations, as well as their attachment sites. These sources may include semantics, or lexical frequencies as determined from a corpus.

In what follows, we begin a more detailed discussion of certain essential aspects of our parsing algorithm. We start by considering what structural possibilities the parser needs to consider when it tries to combine the current underspecified representation R with an elementary tree η that is anchored by the next input word.

5 Control Strategy

This section describes how representations may be combined. Associated with each representation are expectation lists that encode how it expects to fit into a complete sentence structure. We first describe the nature of expectation lists. Next, we develop the control strategy that is used to check if two representations may be combined, through the use of expectation lists. We also consider what actions may take place if two expectation lists are incompatible.

5.1 Expectations

At any given point in the parsing of the input sentence, subsequent input may clearly not be inserted into the part of R that is to the left of that path running from the root of R to the leaf l. Extending the notion of trunk, which has hitherto only been defined for elementary trees, we state that the *trunk* of an elementary representation R is the path from the root to the last input node. Thus, when inserting new structure into R, only the nodes and domination links on and to the right of the trunk need to be considered. These nodes are contained in what we call the *right expectation list* of R.

Analogous observations hold about the elementary TAG trees. Namely, at each point in parsing, an elementary tree η, representing the next input word, is combined with R. When considering how η may be combined with R, it suffices to consider only those nodes and domination links that appear on or to the left of that path from the root to the anchor. These nodes are contained in the *left expectation list* of η.

Now we will consider how to compute the left expectation list of R and the right expectation list of η. In the relevant subparts of R and η, we need to examine nodes that can be involved in combinations: lowering nodes, substitution nodes, and foot nodes. In our expectation lists, we consider expectations of two types: substitution expectations or lowering expectations. A substitution expectation corresponds to a substitution node or a foot node. Conversely, a lowering expectation corresponds to a lowering node.

Each expectation is characterized as optional or obligatory. Substitution expectations are always obligatory. Lowering expectations may either be optional or obligatory. Suppose there exists a domination link that states that node a dominates node b. The lowering expectation corresponding to node b is obligatory if the labels of a and b do not match.

Expectation lists are *ordered* lists of expectations. The order of expectations for one representation corresponds to the sequence in which the parser will try to assemble it with the other representation. The algorithm that is presented in Figure 2 returns an ordered list of expectations when given an underspecified representation or elementary tree and a direction d that specifies whether a left or right expectation list should be found. It first examines the anchor and then makes its way up the trunk of the representation, one node at a time. At each such node n, the algorithm sees if n corresponds to an expectation. If so, this is recorded. It also checks if n has any siblings in the dth direction. If so, each such sibling is scanned in a depth first fashion for any expectations.

Consider the expectations in the right expectation list for α_0 in Figure 1. The algorithm starts at the anchor "hit" and then makes its way up to the V node. The V node has a sibling, NP_1, which is examined in a depth first fashion for any expectations. This yields the first expectation, $NP_{1(oblig,subst)}$. The algorithm continues up the trunk to the bottom VP quasi-node (VP_0), which is associated with another expectation $VP_{(opt,low)}$, which is duly recorded. Continuing further up the trunk, the algorithm finds one more expectation corresponding to the root S, which is $S_{(opt,low)}$. Similarly, the left expectation list of α_0 would be $< VP_{(opt,low)}, NP_{0(oblig,subst)}, S_{(opt,low)} >$.

5.2 Combining Representations

As the parser proceeds through the input sentence from left to right, the parser tries to combine the underspecified representation R with an elementary tree η corresponding to the current input word. Expectation lists encode how a representation to the left may be combined with an elementary tree to the right. We say that the underspecified representation R can be combined with the elementary tree η when the right expectation list of R is *compatible* with the left expectation list of η.

An outline of a proposed procedure to determine the compatibility of two expectation lists is delineated in this section. This procedure has two steps. First, one expectation is taken from the left expectation list of η and the right expectation list of R is searched for a matching expectation. Second, given that the two representations partially combine according to how these first two expectations match, it is seen whether the two representations can be fully combined.

[1] If d is left (right), then *not* d is right (left).

[2] Node p is a *parent* of a node c if p is connected to c by a domination or immediate domination link. For example, a top quasi-node is the parent of the corresponding bottom quasi-node.

[3] The *child* relation is defined analogously to the parent relation in the previous footnote. For example, a bottom quasi-node is the child of its corresponding top quasi-node.

Order_expect(n:node, d:left or right)
  Case n lies on trunk
    If n is a lowering node
      Record n as a lowering expectation
    Loop through each *sibling* i of n that appears to the d of n from *not* d[1] to d
      Order_expect(i, d)
    If n has a parent p[2]
      Order_expect(p, d)
  Case n does *not* lie on trunk
    If n is a substitution node
      Record n as a substitution expectation
    Else (n is an interior node)
      If n is a lowering node
        Record n as a lowering expectation
      Loop through each *child*[3] i of n that appears to the d of n from *not* d to d
        Order_expect(i, d)
      If n is an adjoining node
        Record n as a lowering expectation

Figure 2: Algorithm that returns an ordered list of left or right expectations.

Finding the First Pair of Matching Expectations

We take the first expectation e_1 from the left expectation list of η. e_1 may either be a substitution expectation or a lowering expectation. If it is the former, we look for a matching lowering expectation in R. Conversely, if it is the latter, we look for a matching substitution expectation.

It may occur that, although e_1 is an obligatory expectation, there is no expectation in R that matches. We need not consider this particular situation further, because the two representations are obviously incompatible. Alternatively, although there may be no matching expectation for e_1, e_1 is an optional expectation. In this case, we can skip to the next expectation on the left expectation list of the elementary tree.

Matching the Rest of the Two Expectation Lists

Suppose that a matching expectation e_m (in R) for e_1 (in η) is found. Now there are two tasks that must be accomplished. First it must be ensured that this matching expectation can indeed lead to a valid combination of the two representations that are under consideration. This is accomplished as follows. Suppose that n_m and n_1 are the nodes corresponding to the expectations e_m and e_1, respectively. Now let p_m be the path from the root of the underspecified representation R to n_m. Conversely, let p_1 be the path from the root of the elementary tree η to n_1. The task is to find out if the components along the path p_m can be interspersed with the components along the path p_1. This is done by iteratively matching the expectations corresponding to the nodes that lie on path p_m with those that lie on the path p_1, starting from those expectations corresponding to nodes that lie closest to e_m and e_1. Second, any remaining unmatched obligatory expectations in η should be matched, in order of their appearance on the expectation list, by recursively calling the aforementioned procedure.

Representing Ambiguity of Attachment

Suppose e_1 is a substitution expectation and we find a matching lowering expectation e_m in R. That e_m must be the corresponding attachment site for e_1 does not immediately hold. In Figure 3(b), for example, the auxiliary tree could be adjoined at either of the VP's in R. To capture such attachment ambiguity, rather than equating n_1 and n_m (the node corresponding to the first expectation in the expectation list for R that matches e_1), we will say n_1 dominates n_m. On the other hand, in Figure 3(a), there is no ambiguity in attachment of the new elementary tree. In this case we can make a stronger statement by equating n_1 and n_m.

In general, the type of attachment ambiguity that we underspecify is where e_1 is a substitution expectation that matches more than one lowering expectation and no obligatory expectations occur between the first and

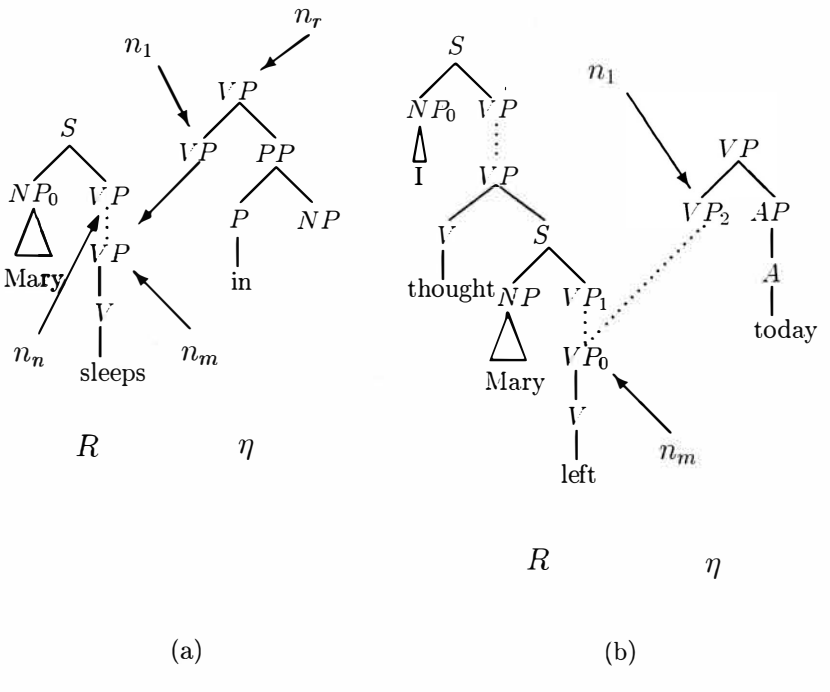

Figure 3: Example of case 1 of the algorithm to match expectation lists. (a) is the subcase of unambiguous attachment. (b) is the subcase of ambiguous attachment.

the last matching lowering expectations on the left expectation list of η, inclusively.

Whether such an ambiguity exists is determined as follows. Suppose e_1 is a substitution expectation. Furthermore suppose its combination with e_m, the first matching lowering expectation, does in fact lead to a valid combination of representations. We may find another valid combination by scanning further in the right expectation list of the underspecified representation R, starting from e_m, to find another matching expectation $e_{m'}$ (before encountering an obligatory expectation that does not match e_1) and verifying that it leads to another valid combination of structures, using the same method that we performed for the first matching expectation e_m.

Our underspecified representation captures this ambiguity as follows. Notice that no matter where the elementary tree η is adjoined, it is the case that the node n_1 will always dominate the lowest attachment site n_m. Therefore, in cases of such ambiguity, we stipulate such a domination as shown in Figure 3(b). Notice that while the example in Figure 3(b) corresponds to adjoining, a similar situation arises even if n_1 is a substitution node.

Representing the ambiguity in this way has various repercussions. For example, if the elementary tree that is being ambiguously added is an auxiliary tree, then we should make sure that if disambiguation should subsequently occur, an additional assertion should be made in order to guarantee a proper TAG adjunction. In detail, suppose that γ is the auxiliary tree, and n_1 is the foot node of γ. When disambiguation occurs, n_1 is equated with some bottom quasi-node n_b. In that case, an additional stipulation should be made that the top quasi-node n_t that corresponds to n_b dominates the root of γ. Another consequence is that an underspecified representation R that encodes such ambiguities is no longer the treelike structure that was assumed earlier. This necessitates alterations in our algorithm for computing expectation lists, as will be discussed in Section 6.

Details Behind Matching the First Two Expectations

As noted previously, matching one expectation list against another entails finding a sequence of pairs of corresponding substitution and lowering expectations. Whether two expectations match depends upon various factors, such as whether the substitution expectation occurs in R or η, and whether the nodes corresponding to the expectations are substitution nodes, foot nodes, or quasi-nodes.

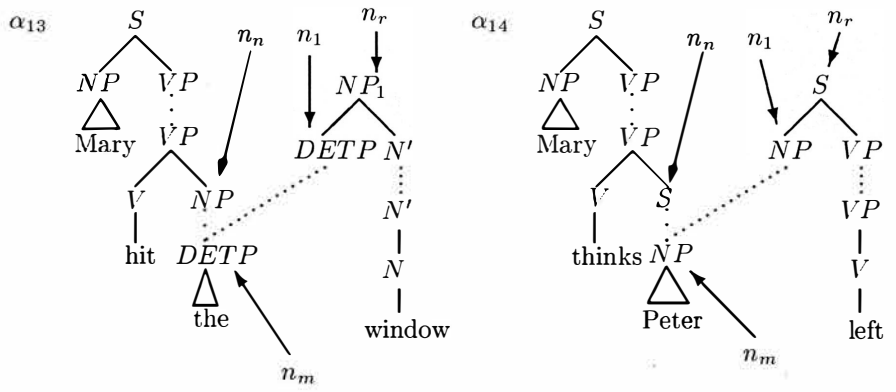

Figure 4: Case where stating that NP_0 dominates NP_1 is justified.

Let e_1 be the first expectation from the left expectation list of the elementary tree η. The expectation e_1 may either be a substitution expectation (corresponding to a substitution node or a foot node) or a lowering expectation (corresponding to a lowering node). We handle the tests for satisfaction of these two kinds of expectation as case 1 and case 2, respectively. In the following we take e_m to be the first expectation in the right expectation list of R that matches e_1. Furthermore, let n_m and n_1 be the nodes corresponding to the expectations e_m and e_1, respectively.

Case 1: e_1 is a Substitution Expectation. As stated previously, nodes n_1 and n_m are equated if there is no ambiguity in the placement of n_1. Otherwise, the ambiguity is encoded into the representation with node n_1 dominating n_m. Now consider the situations where n_1 is either a substitution node (Case 1a) or a foot node (Case 1b).

Case 1a: n_1 is a Substitution Node. If n_1 is a substitution node, then n_m must be the root of some elementary tree. This follows because the TAG operation of substitution requires that the root of an elementary tree be equated with a substitution node and that entire tree must have already been integrated into R. Node n_m may be dominated by another node n_n in R. n_n may be a substitution node (Case 1a-i) or a foot node (Case 1a-ii). We proceed to consider the cases where n_1 is to be equated with n_m.

Case 1a-i: n_n is a Substitution Node. Let n_r be the root of η. We can say that it is dominated by n_n if $n_r \in LC(n_n)$ [4]. See α_{13} of Figure 4 for an example. Otherwise, our original assertion of equality between n_1 and n_n was in error. The use of domination of n_r by n_n is to allow for further material to the right to be inserted between these two.

Case 1a-ii: n_n is a Foot Node. Suppose n_n is a foot node. If $n_r \notin LC(n_n)$ then n_n does not dominate n_r. Otherwise, n_n may or may not dominate n_r; the rest of the expectation lists would have to be checked in order to find out if there is the possibility of ambiguity in the placement of n_r. See α_{14} of Figure 4.

Case 1b: n_1 is a Foot Node. If n_1 is a foot node, then n_m must be a bottom quasi-node. Analogous to Case 1a, this follows because the TAG operation of adjoining requires that the foot node be equated with a bottom quasi-node. Again, where there is no ambiguity in the placement of n_1, we consider whether n_n is a foot node (Case 1b-i) or a top quasi-node (Case 1b-ii). Note that node n_n cannot be a substitution node because a substitution node cannot occur along the path between a top and bottom quasi-node.

Case 1b-i: n_n is a Foot Node. This case is similar to Case 1a-ii.

Case 1b-ii: n_n is a Top Quasi-node. If n_n is a top quasi-node, then it dominates n_r. This is to ensure that a valid TAG adjunction is performed. See Figure 3(a) for an example.

Case 2: e_1 is a Lowering Expectation. Again suppose that n_m and n_1 are the nodes corresponding to the expectations e_m and e_1, respectively. This time e_m must be a substitution expectation. Note that the possible attachment of n_m to more than one node in η does not occur. Because n_1 is a lowering node, it can only be a bottom quasi-node (Case 2a) or a root of an elementary tree (Case 2b).

Case 2a: n_1 is a Bottom Quasi-node. If n_1 is a bottom quasi-node, then n_m must be a foot node. It cannot be a substitution node because that would mean that a substitution node exists along the path between

[4] $LC(\nu)$ stands for left corner, by which we include anything that can appear in the left periphery of a tree whose root is ν.

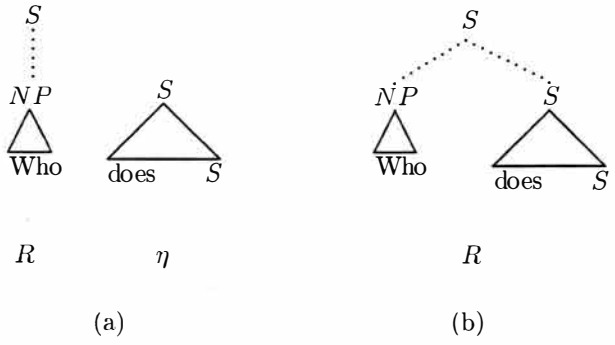

Figure 5: (a) Representations that are selected when the input "Who does..." is encountered. (b) The same representations combined with domination.

a top and bottom quasi-node. Node n_1 is dominated by another node n_k in elementary tree η, which can only be a top quasi-node. Because n_k is a top quasi-node, it dominates the root of the auxiliary tree that has n_m as a foot node. This follows from the fact that we want to preserve TAG adjoining.

Case 2b: n_1 is the Root of an Elementary Tree. If n_1 is the root of the elementary tree η, then n_m can either be a substitution node or a foot node. In this situation, node n_m is made to dominate n_1. Equality is not asserted because it may be possible for material from the right to be inserted between n_m and n_1.

When the Relationship between Two Representations is Locally Unknown

It is also possible that there is no expectation e_1 in the left expectation list of the elementary tree η which might suggest how to combine it with the underspecified representation R to its left. Furthermore, the expectations in that list may have all been optional. In this situation, although no assertion of domination can combine R and η, it is possible that subsequent input may be able to combine them. Call this Case 3. It occurs for example in the context of the sentence "Who does John annoy?" After the prefix "Who does..." has been seen, the parser considers combining the two representations that are shown in Figure 5(a), only to find that it is impossible.

As a general strategy for case 3, we will assume that the disparate representations are combined using dominance where both of the representations are dominated by the lowest node which we know for certain will dominate both. An example is shown in Figure 5(b). This strategy would entail complications to the procedure of building expectation lists and checking for the compatibility of expectations. Notice that this strategy is akin to buffering the uncombinable items. There is therefore the problem of determining what, if anything, is the proper buffer size.

6 Computing the Standard Referent

In D-theory, the notion of standard referent plays an important role. The standard referent is a minimal model (tree) of a description. It is usually obtained by minimizing the path lengths of all of the domination links. Such a tree would be useful in incremental interpretation. For us the method for obtaining the standard referent is also useful in determining the order in which the components of the current underspecified representation are to be combined with new additional input. Recall that the underspecified representation that our parser produces allows for non-treelike forms in order to describe some forms of attachment ambiguity. On the other hand, the algorithm that was given to compute expectation lists requires a treelike description as input.

Consider the description α_8 in Figure 6. The problem here is that obtaining the expectation list for α_8 is not possible because an expectation list requires a unique ordering of domination. In particular, whether XP_3 dominates XP_4 depends on whether or not XP_3 is placed above XP_4. If the standard referent is implicitly assumed to be the default structure, the question arises as to what the standard referent of α_8 is. Such cases have never arisen in previous work on D-theory. [5] In defining the notion of standard referent here, we have

[5] See [Rogers and Vijay-Shanker, 1996] for discussion on the determinism hypothesis in D-theory parsing and the uniqueness of the standard referent.

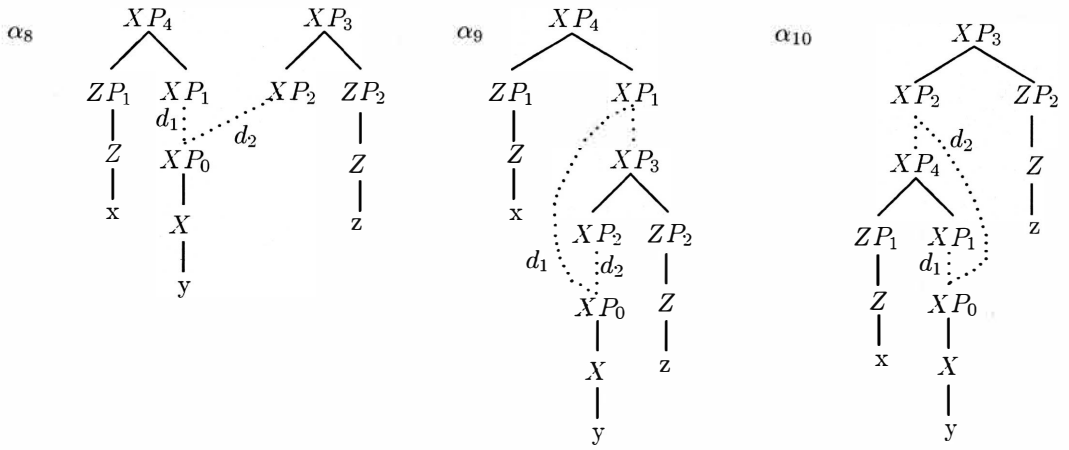

Figure 6: An example of our parser's underspecified representation (α_8), its standard referent (α_9), and another representation compatible with α_8 (α_{10}).

to consider disambiguating the attachment ambiguity that is expressed in structures such as α_8. Because the standard referent is usually taken to be the default structure, its definition could also be used to give an ordering for the expectation list.

The basic strategy that we would like to use in order to determine the standard referent is to minimize the lengths of the domination links. This strategy, however, does not always yield a single standard referent, as seen in α_8 of Figure 6. In this case, there is no unique standard referent because the sum of the lengths of the domination links d_1 and d_2 does not vary with the placement of the auxiliary tree that is rooted by node XP_3. [6] The difficulty arises when two domination links reach the same node. We call such pairs of domination links *ambiguous*. In α_8 the pair of domination links d_1 and d_2 are ambiguous because they both dominate the same node XP_3.

In order to guarantee that a single standard referent is obtained, the process of determining a standard referent is divided into two stages. First, all of the ambiguous domination links are disambiguated by minimizing their lengths sequentially according to a predetermined order, yielding an unambiguous representation R'. Second, a single standard referent is determined from R' by minimizing the sums of the lengths of the domination links.

Suppose d_1 and d_2 are any ambiguous pair of domination links in R. d_1 and d_2 state that certain nodes U and V dominate a node X. Clearly, U and V must come from different elementary trees, say β_U and β_V. There must have been an order in which β_U and β_V were introduced into R. This is the order in which the ambiguous domination links d_1 and d_2 are ordered.

Each domination link is assigned a value that specifies the time at which the elementary tree in which the link's parent node resides was introduced into R. In defining the order for an expectation list, the domination link with the higher value is ordered before one with a lower value. Consider for example the underspecified representation α_8 of Figure 6. There are two ambiguous domination links d_1 and d_2 with $value(d_1) < value(d_2)$. Consequently, d_2's length is minimized before d_1's length. The resulting underspecified representation is α_9.

There are two motivations behind our method. First, computing the expectation list of the result of computing the first step of the standard referent yields a sequence of expectations that does not omit any possible attachment points. For example, the expectation list of α_9 encodes the possibility of material being inserted between XP_1 and XP_3, something that would not happen had XP_3 been postulated to dominate XP_4, as in α_{10}. Second, this strategy typically corresponds to right association, as described in [Frazier, 1995]. Specifically, minimizing more recently created domination links first means that more recently added subtrees will be attached as low as possible. For example, the standard referent for the representation in Figure 3(b) has the adverb "today" modifying the lower verb "left." Insofar as right association has been postulated as a

[6] Another possible standard referent of α_8 would be found by equating the two pairs of nodes XP_1 and XP_2, and XP_3 and XP_4, which would result in a parse tree where, for example, ZP_1 and ZP_2 are siblings. Nevertheless, it does not result in a tree obtained by any sequence of TAG operations, assuming that the component that is rooted at XP_4 and the component that is rooted at XP_3 come from different elementary structures. We avoid this undesirable result by never attempting, in computing the standard referent, to equate two nodes simply because they both happen to be asserted as parents of a single node.

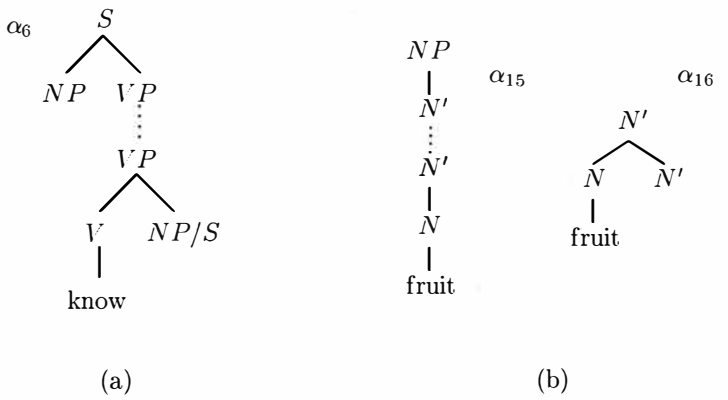

Figure 7: (a) Elementary representation corresponding to two elementary TAG trees. (b) Elementary trees that require underspecification in order for parser to avoid backtracking.

general strategy for human parsing, modulated perhaps by semantic and discourse factors, it corresponds with this strategy of computing the standard referent, which may be used as the baseline structure for semantic interpretation.

7 Elementary Representations

Underspecification plays an important role in reduced commitment parsing. Too great an underspecification means that few constraints are placed on what the final structure of the sentence will look like. It diminishes the predictive ability of the parser and enlarges the range of potential structures that are considered at any given time. Conversely, too little an underspecification leads the parser into making incorrect predictions and reduces the number of sentences that can be parsed without backtracking.

The enlarged domain of locality that TAG provides causes the latter sort of problem in this context. Let us select the strategy of choosing only one elementary tree at each step in the parse. Consider the problems it raises when trying to parse the following pair of sentences: "I know the answer" and "I know the answer is wrong." Crucially, after having seen the word "know" the parser cannot predict which elementary tree to select because the extended domain of locality of TAG requires two different elementary trees for "know," each corresponding to a different subcategorization.

In order to alleviate this problem, it is useful to coalesce different elementary trees into a single representation because at this point, there is no way to tell which is the correct tree to choose. We merge the two elementary trees for the verb "know" by having the substitution node corresponding to its object ambiguously specified as NP or S. See Figure 7(a). We call a single set of assertions with the added proviso of node label underspecification (which represent a set of elementary TAG trees) an *elementary representation.*

The expedient of node label underspecification is also useful in avoiding backtracking in several other situations. Another example is the ambiguity that is associated with PP attachment. Suppose that a PP may either attach at the N' level or the VP level. The problem is that there is a separate elementary tree for each case. With node label underspecification, these separate elementary trees may be coalesced into one elementary representation. Using such elementary representations, we can delay the PP attachment decision until all of the necessary disambiguating information is encountered, as in Figure 8.

Other situations require underspecification of structure in addition to node label underspecification. In general, such cases occur when there is an optional argument or an optional modifier. For example, the verb "wash" optionally subcategorizes for an NP. Another instance occurs with the sentence prefix "The wedding marked..." There is ambiguity in whether the reduced relative or the main verb tree for "marked" ought to be chosen. Notice that parsing free word order languages also introduces this problem of tree selection. The ambiguity in selection of tree structure may even cross part of speech boundaries. This is exhibited in the sentence pairs "Fruit flies like an orange" and "Fruit flies through the air." The relevant trees for the word "fruit" are shown in Figure 7(b).

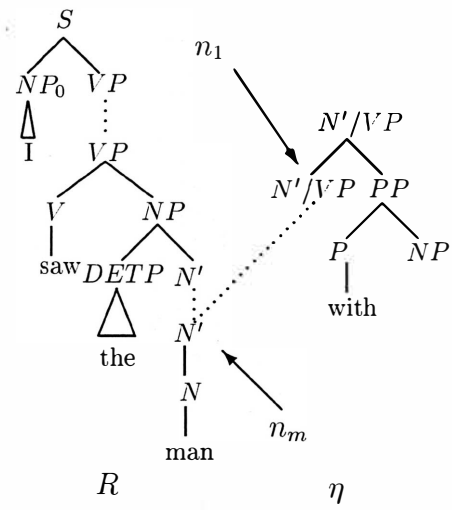

Figure 8: Node label underspecification allows us to represent PP attachment ambiguity.

We are exploring various possibilities for representing this structural ambiguity. One possibility is to license only those parts of an elementary tree up to a certain projection. Take the trees in Figure 7(b) for example. Recall that in the case of seeing the word "know," we underspecified the argument label because it was unavailable. Similarly, upon encountering the word "fruit" we underspecify the set of two elementary trees because only projection up to the N level is licensed. The method suggested by [Evans and Weir, 1997] can perhaps be used to coalesce these structures.

8 Conclusions and Future Work

In this work, we consider parsing TAG using underspecified descriptions of trees featuring dominance and underspecified node labels. The prediction component of an Earley parser corresponds to our assertion of domination. The completion component of an Earley parser does not have an analogue in our parser; it is a manifestation of how an Earley parser considers all possible parses while our parser leaves the situation underspecified. Specifically, in order to underspecify the attachment, we find that in most of the cases in Section 5.2, our parser does not immediately perform TAG operations of substitution and adjoining. That would require equating nodes from two different trees. Instead, our parser asserts domination that leaves us uncommitted to a particular attachment site. Nevertheless, when our parser recovers a parse tree from the underspecified representation, it is necessary for our parser to ensure that the resulting tree corresponds to one derived using substitution and adjoining.

In contrast with traditional work in TAG parsing, we envision our parser to be one that delays attachment decisions. However, we would not like our parser to leave the situation completely underspecified, but to reduce the search space through the judicious use of extra information. This is our notion of reduced commitment, the idea that the underspecified representation that the parser constructs encodes only those relations that the parser strongly believes must hold for the entire sentence. This extra information may include semantics, information about thematic roles, selectional restrictions, or frequency data that are collected from a parsed corpus. It is a subject for future work.

While our parser draws elements from D-theory, this work also extends beyond current work on D-theory. Current work on D-theory parsers do not use any explicit grammar formalism or grammar. Our choice of TAG was motivated by several factors. First, TAGs in general have the desirable properties of factoring recursion and localizing dependencies, which provide argument and structural requirements that are needed by D-theory style parsers for making its assertions regarding structures. Second, TAGs encode linguistic knowledge in a declarative format, making it easier to develop and modify than a more procedural design. Third, extensive TAG grammars have already been developed which could be quickly adapted to the deterministic parser that is proposed here.

Our use of TAG notwithstanding, we use dominance in a different manner than is customary with D-theory parsers. D-theory parsers usually construct a treelike description. On the other hand, some of the descriptions that our parser constructs are not necessarily treelike. This was necessary to capture attachment site ambiguity. In these cases, a statement is made that the tree which may be ambiguously placed dominates the lowest possible attachment site. Consequently, this site would be dominated by two structures, each of which is ambiguously ordered with respect to domination.

This ambiguous representation had a direct impact on our computation of the standard referent, another area where we deviate from normal D-theory parsers. We have provided a means of determining a standard referent under these exceptional circumstances. Our solution appears to produce a structure that conforms to right association, which is not the case for all other D-theory parsers. Not only may our standard referent aid in incremental interpretation but in our case it also aids in deciding how representations may be combined.

We have also discussed how representations may be combined through the use of expectation lists. We have provided a taxonomy of expectations. These help define a number of cases that illustrate when domination may be postulated between components of different elementary trees. We have enumerated many of these cases and given examples of their application.

Finally, we have argued that in incorporating TAG into D-theory style parsing, the assumptions that are standardly made about TAG trees can interfere with the goal of delaying attachment decisions. To alleviate this situation, we have suggested how several TAG trees should be coalesced into one elementary representation. An elementary representation represents elementary TAG trees that differ only in their node labels. This underspecification essentially allows the parser to delay the choice of the specific elementary TAG tree until further information can be obtained. We have also enumerated other cases where node label underspecification is insufficient. We have postulated a solution based on limiting how much of a projection of an elementary tree that we would license in a given situation. This is another subject for further investigation.

References

[Evans and Weir, 1997] Evans, R. and Weir, D. (1997) Automaton-Based Parsing for Lexicalized Tree Adjoining Grammars. In *Proceedings of the Fifth International Workshop on Parsing Technologies.* Cambridge, Massachusetts.

[Frazier, 1995] Frazier, L. (1995) Issues of Representation in Psycholinguistics. In J. L. Miller and P. D. Eimas (Eds.), *Speech, Language, and Communication* (pp. 1-27). Academic Press, New York, New York.

[Gorrell, 1995] Gorrell, P. (1995) *Syntax and Parsing.* Cambridge University Press, United Kingdom.

[Marcus, Hindle, and Fleck, 1983] Marcus, M., Hindle, D., and Fleck, M. (1983) D-Theory: Talking about talking about trees. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics* Association for Computational Linguistics, Cambridge, Massachusetts.

[Rambow, Vijay-Shanker, and Weir, 1995] Rambow, O., Vijay-Shanker, K., and Weir, D. (1995) D-Tree Grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, Cambridge, Massachusetts.

[Rogers and Vijay-Shanker, 1996] Rogers, J. and Vijay-Shanker, K. (1996) Towards a Formal Understanding of the Determinism Hypothesis in D-Theory. In H. Bunt and M. Tomita (Eds.), *Recent Advances in Parsing Technology* (pp. 59-78). Paper presented at IWPT '93. Kluwer Academic Publishers, Boston, Massachusetts.

[Sturt and Crocker, 1996] Sturt, P. and Crocker, M. (1996) Monotonic Syntactic Processing: A Cross-Linguistic Study of Attachment and Reanalysis. *Language and Cognitive Processes.*

[Vijay-Shanker, 1992] Vijay-Shanker, K. (1992) Using Descriptions of Trees in a Tree-Adjoining Grammar. *Computational Linguistics*, 18(4), 481-517.