Unification and Efficient Computation in The Minimalist Program

Sandiway Fong
Departments of Linguistics and Computer Science
University of Arizona
Tucson AZ 85721 USA

Introduction

From the perspective of computational modeling, in recent years, the shift from a declarative to an operational approach to the description of linguistic theories has important implications for both efficient computation and the space of possible implementations.

The declarative approach is exemplified by the Principles-and-Parameters (P&P) Approach (Chomsky 1981), in which linguistic constraints (or filters) are abstractly stated over syntactic configurations at various levels of representation, and perhaps derivationally, between levels of representation. These filters may call upon a variety of linguistic devices that may be generated in the course of a derivation, including indices, chains (of movement), and various empty categories, e.g. traces. The (not unsubstantive) problem of generating appropriate syntactic descriptions, perhaps by some top-down, bottom-up or mixed assembly of syntactic objects, is (apparently) left to the grammar designer. There is considerable freedom of implementation. For instance, apart from basic dependencies between various principles, the order of application of the constraints is undetermined. Perhaps the most straightforward realization of the P&P Approach can be found in the largely generate-and-test paradigm given in (Fong 1991).[1] The radar plot in Figure 1 illustrates the degree of overgeneration, and thus computational inefficiency,

[1] In theory, the generate-and-test paradigm, i.e. generate linguistic descriptions and explicitly rule out illicit cases, can be contrasted with a constraint-based approach, i.e. one that emits only licit linguistic descriptions. From a computational complexity perspective, the constraint-based approach offers the possibility of avoiding the inefficiency inherent in over-generating linguistic descriptions. Careful comparisons are necessary to verify that the bookkeeping necessary to maintain and manipulate constraint descriptions does not trump the inefficiency introduced by simple overgeneration. (See also note [2].) However, the author is not aware of any substantial purely constraint-based implementations of the Principle-and-Parameters Approach.

typically observed in the implemented parsing system. For the sentence *John is too stubborn to talk to Bill*, only one, the correct parse, out of 33 candidate linguistic descriptions successfully emerges from constraint testing.[2]

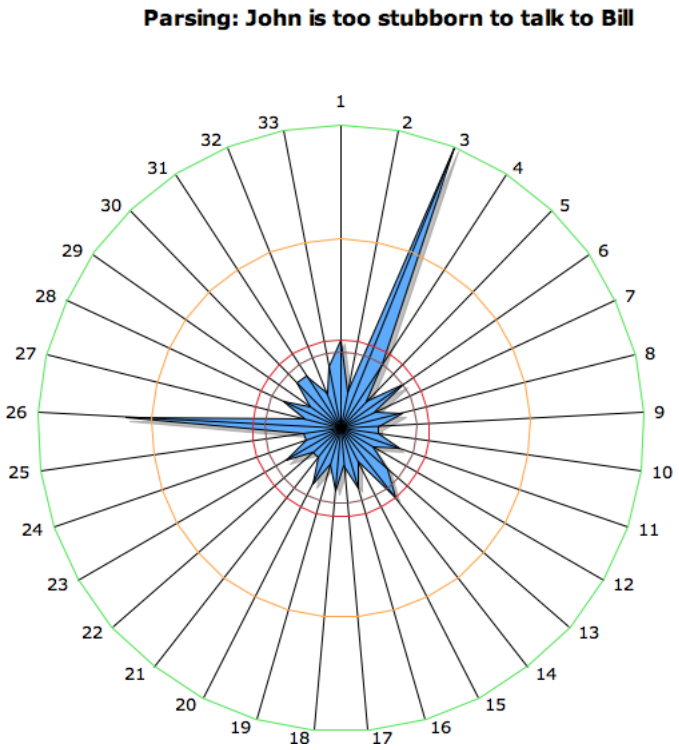

Figure 1: Example of the generate-and-test paradigm

By contrast, recent theories in the Minimalist Program, e.g. (Chomsky 2001) and thereafter, seemingly leave little wiggle room for implementational variation or inefficiency. As will be described later in this paper, the assembly of licit syntactic objects has been made precise down to a stepwise (fundamental) operational level. Linguistic constraints such as those in the P&P Approach must be re-coded or re-analyzed in terms of the interaction between just two fundamental operations, bottom-up Merge (*external*, encoding syntactic composition and *internal*, encoding sub-object displacement) and Agreement (between a recently-merged object that function as a *probe* locally seeking already-merged sub-objects, termed *goals*). Both probe and goal objects have uninterpretable features that must be valued (and discharged) in the course of assembly.

---

[2] In practice, serial constraint testing can be organized to favor the early elimination of illicit candidates (to reduce implementational inefficiency). Note that most of the candidates in Figure 1 are eliminated early on by the Case and Theta group of constraints. Only one other parse (in addition to the correct parse) makes any significant headway out of the Case and Theta group.

Computationally speaking, it is assumed that Agree applies as early as possible in the derivation as Merge proceeds to build a composite syntactic object starting from a pre-selected lexical array of basic functional and substantive objects. The resulting syntactic object (SO) is *simple* in the sense that (artifactual) devices not originally present in the lexical array cannot be introduced or built in the course of assembly, e.g. no no traces, movement chains or indices can be added to syntactic descriptions. Moreover, not only must a probe "fire" as soon as it is merged, but probe-goal search is limited to local domains defined by Phase theory. A target with an uninterpretable feature beyond the range, measured in Phases, of an intended probe will cause narrow syntax computation to crash, i.e. terminate unsuccessfully.

We must make precise the notions of efficiency and complexity to be adopted in the paper. In computer science, there are various methods of measuring complexity and efficiency. For example, depending on the situation it may be appropriate to adopt asymptotic analysis: if we assume the size of the data set, e.g. the length of the input sentence, may increase without bound, we can ignore constant factors and compare the fundamental rate of growth of the number of operations needed to compute an answer over a fixed grammar with respect to different algorithms. At a more abstract level, i.e. independent of particular grammar, it's sometimes possible to compare the generative power and computational complexity of different grammar formalisms: in general, we find there is a not unexpected tradeoff between a formalism's expressive power (what is encodable) and computational complexity. For example, traditional formal language theory tells us there exists a sliding scale of complexity from finite-state automata (equivalently, regular grammars or regular expressions), through push-down (PD) automata (equiv., context-free grammars), to nested-PD automata (equiv., context-sensitive grammars). It has been suggested that natural languages may be characterized as nestling between context-free and context-sensitive bounds of expressive power. In particular, mildly context-sensitive grammars allow the expression of limited non-context-free dependencies found in natural language but retain polynomial time parse-ability (a desirable computational property denied to unrestricted context-sensitive grammars). It has been shown that certain mildly context-sensitive grammar formalisms are all formally equivalent in expressive power (Vijay-Shanker & Weir 1994).

With respect to the P&P Approach, we can compute the degree of overgeneration as an approximation to system and implementational (in)efficiency.[3] However, since the linguistic devices employed in the P&P Approach are many and varied, a usefully limited characterization of its complexity is not practical. With respect to the Minimalist Program, a (simplified) mathematical characterization of the uninterpretable feature-checking mechanism can be found in Stabler's (1997) Minimalist Grammar (MG) formalism. This framework enjoys similar expressive

[3] Generally, overgeneration could be due to insufficient grammatical constraints or to the implementation itself. Given a sentence with a single licit parse, in the former case, extra parses will be generated. In the latter case, the system will produce multiple candidates but only the correct parse will survive constraint checking.

power and complexity to the mildly context-sensitive grammars mentioned earlier. However, it remains to be seen whether MG can encode the full range of Agree and Phase theory constraints described in (Chomsky 2001) and later publications. Furthermore, the idea of overgeneration, introduced above for the P&P Approach, is not a useful measure of efficiency in the case of the Merge/Agree system. Much like a jigsaw puzzle, there is only one defined pattern per pre-selected lexical array. Thus, correctly implemented, there should be only one way in which those objects can be fit together.[4] The notion of efficiency therefore is only meaningful with respect to the details of the Agree operation. In particular, we can evaluate the extent or depth of the search for goals by probes and count the number of agree relations computed in the course of a derivation. This paper argues that adopting the mechanism of unification for feature matching will result in improved efficiency in these terms. And that minimizing the number of operations and localizing goal search as far as possible is in keeping with the spirit and goals of the Minimalist Program. But first, we make precise the computational details of the Merge/Agree system in the next section.

Background

(Chomsky 2001) sets out the following operations as being basic to computation.

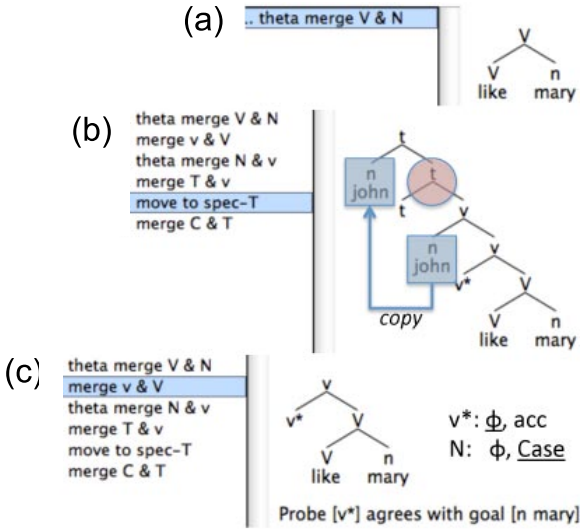

Figure 2: Basic operations

1. Merge "an *indispensible operation of a recursive system"* comes in two flavors.
    a. *External Merge* takes two syntactic objects (SO) α, β and forms the set-merged SO: {α, β}. For labeling, one of either α or β must project, i.e.

[4] This is true up to a point. For example, from the same lexical array, one can assemble both *John likes Mary* and *Mary likes John* using the same sequence of steps, depending on which nominal, *John* or *Mary*, is selected to be the subject and direct object, respectively. See the sequence of operations shown in Figure 3.

label({α, β})=label(α) or label(β). An example, with α = [$_V$ like] and β = [$_n$ mary], is given in Figure 2.a.

b. *Internal Merge* implements displacement. Selecting SOs α and γ, γ properly contained in α, form the aggregate set-merged SO {α, γ}. Furthermore, label({α, γ})=label(α).
Figure 2.b illustrates internal merge in the case of subject raising to tense (t). α = [$_t$ t [$_v$ [$_n$ *john*][$_v$ v [$_V$ [$_V$ *like*][n *mary*]]]]], and subject γ = [$_n$ *john*]. Note there are two copies of γ. No trace is generated.

2. *Agree* obtains between an active probe SO α (*active* = still having uninterpretable features), and an active goal SO β. β should be the closest locally available goal in the c-command domain of α.[5] Features of α and β are matched, and uninterpretable features of both probe and goal are deleted.

   Figure 2.c illustrates probe-goal agreement for α =v* (transitive v) and β = [$_n$ *mary*]. Goal [$_n$ *mary*] has interpretable φ-features, i.e. person (3rd), number (singular) and grammatical gender (feminine) but uninterpretable Case. The probe v* has uninterpretable φ-features and but can value accusative Case. v*'s uninterpretable φ-features are valued through matching with the goal's φ-features. The goal [$_n$ *mary*] in turn receives accusative Case from v*. After Agree, neither probe nor goal remains active, as their uninterpretable features have been valued.

A convergent derivation using Merge and Agree obtains when the initial selection of SOs has been fully utilized and no uninterpretable features remain at large.

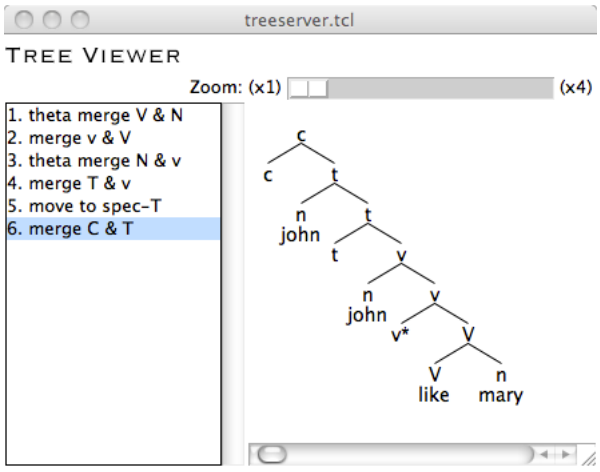

Figure 3: Parse of *John likes Mary*

Sentential structure is assumed to involve a sequence of (top-down) selection from an inventory of primitive functional elements including the c (complementizer), t

[5] The term "local" refers to the limits on search extent imposed by Phase theory on the unrestricted c-command domain.

(tense) and v (little v), V pairs. These functional categories come in various flavors, e.g. V may select for a thematic direct object, as with transitive and unaccusative verbs. V may be objectless, as with unergative verbs. V itself is directly selected by v, which may select for a thematic subject, as in the case of unergative or transitive verb (v*, which can value accusative Case), or be subjectless, as in the case of passives. Tense (t), which has an EPP or edge feature and requires a subject, may come in uninterpretable φ-complete (tensed) or φ-incomplete (infinitival) flavors; however, only φ-complete tense can value nominative Case.

An example of the parse produced for a simple transitive sentence *John likes Mary* is given on the right in Figure 3. Although there are two copies of *John*, only the highest copy is pronounced.[6] The convergent Merge/Agree derivation sequence is summarized on the left, and traced in detail in Figure 4.[7]

0. lexical array	[c][t][n john][v*][V like][n mary]
1. theta merge V & n	[V[V like][n!case mary]]
2. merge v & V	[v[v*][V[V like][n mary]]]
Probe [v*] agrees with goal	[n mary]
3. theta merge n & v	[v[n!case john][v[v*][V[V like][n mary]]]]
4. merge t & v	[t[t][v[n john][v[v*][V[V like][n mary]]]]]
Probe [t] agrees with goal	[n john]
5. move to spec–T	[t[n john][t[t][v[n john][v[v*][V[V like][n mary]]]]]]
6. merge C & T	[c[c][t[n john][t[t][v[n john][v[v*][V[V like][n mary]]]]]]]

Figure 4: Merge/Agree sequence for *John likes Mary*

In the basic case, as in Figure 4, there is a simple, one-to-one correspondence between probes and goals, and this leaves no room for computational optimization. However, in more complex scenarios, Agree relations may simultaneously hold between a single probe and multiple goals: in particular, when either the probe or goal may be φ-incomplete. In these cases, as we will see in the next section, unification can improve the efficiency of the computational system.

Unification and Computation

We will illustrate the advantage of unification on the following examples from (Chomsky 2001:4b-c):

3.
   a. there are likely to be awarded several prizes

---

[6] Inflectional morphology is not implemented in Figure 3. We assume a procedure that spells out *likes* from t (tensed) + v* (φ: 3rd-sg-fem)+ V (*like*).

[7] In Figure 4, the !*F* notation, where *F* is a feature, as in [n!case …], is used to indicate that a SO has a (currently) unvalued uninterpretable feature. In steps 1 and 3, *Mary* and *John* have unvalued Case, respectively. By steps 2 and 4, the uninterpretable Case in each case has been valued by probes v* and t, respectively.

b. several prizes are likely to be awarded

4.

a. we expect there to be awarded several prizes
b. we expect several prizes to be awarded

Examples (3.a-b) and (4.a-b) contain the same passivized embedded clause *to be award-ed several prizes*. The matrix predicate is a *raising* predicate, *be-likely* in (3.a-b) and an *Exceptional Case Marking* (ECM) verb *expect* in (4.a-b). Within the embedded clause, passive *v* is non-Case-valuing and the direct object (DO) is *several prizes*. The adjectival past participle *–ed* (PRT) is assumed to be is φ-incomplete and have uninterpretable Case (unrealized in English).[8] The embedded clause includes a subject position, overtly occupied by a copy of *several prizes* in (4.b) and by pleonastic *there* in (4.a); the embedded subject position is covert in (3.a-b). Finally, tense is φ-incomplete (infinitival) and cannot value Case either.
Thus *several prizes* must get Case from the tense that heads the matrix clause in (3.a-b), and matrix v* in (4.a-b), respectively. In fact, since matrix tense and matrix v* are the only φ-complete probes available, it (namely, tense and v*) must participate in multiple agree relations with various goals in the embedded clause (including the DO and PRT) in order for computation to converge.

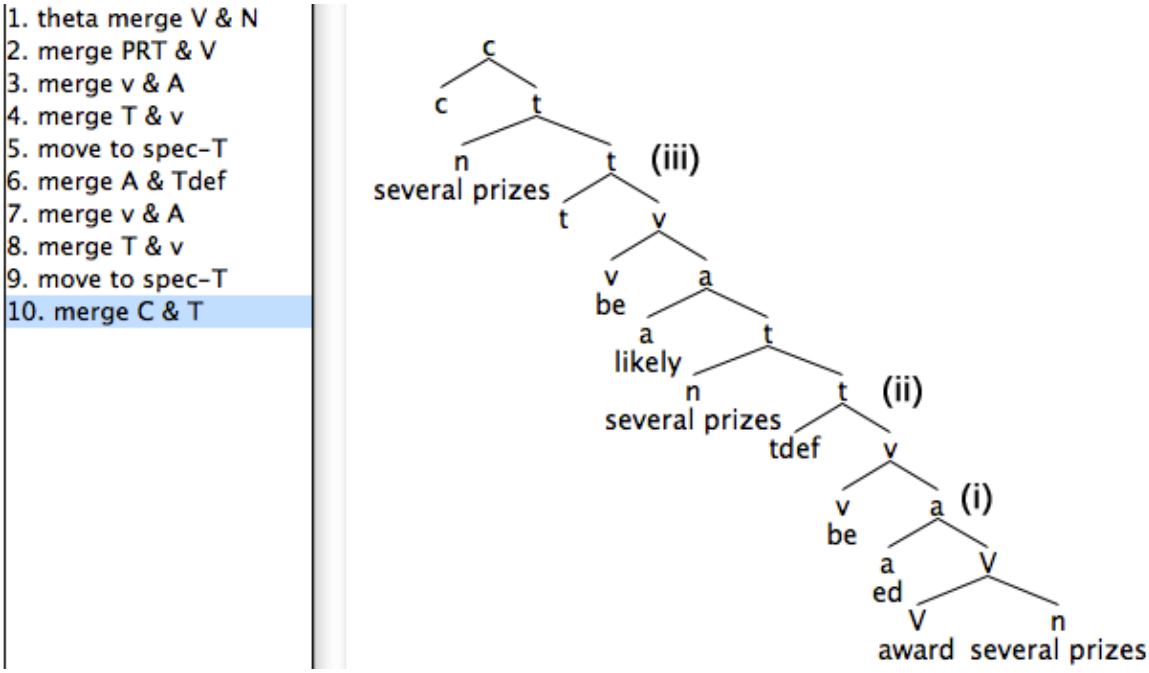

Figure 5: Several prizes are likely to be awarded

Consider the derivation of (3.b), illustrated in Figure 5. The DO *several prizes* raises to embedded tense and then to the highest position at matrix tense where it is pronounced. Along the way, the following Agree relations are computed: at position

[8] PRT is assumed by (Chomsky 2001) to have uninterpretable number and gender φ-features.

(i), adjectival –*ed* (PRT) undergoes feature matching with the DO. The PRT's uninterpretable φ-features are valued by the DO, which possesses interpretable φ-features. However, both PRT and DO lack Case. At position (ii), the embedded tense (tdef = infinitival) agrees with the DO and its uninterpretable φ-features are valued. However, tdef is φ-incomplete and cannot value Case. Since tense has a EPP (or edge) feature, the DO raises. At position (iii), matrix tense (t) probes and finds the raised DO. Since t is φ-complete, it values Case for the DO and its uninterpretable φ-features are valued. In (Chomsky 2001), in order for the derivation not to crash, t must continue to probe beyond the raised DO to value PRT's Case feature as well. Thus t is in a multiple agreement relation. However, if feature matching is implemented using unification, the matrix tense's secondary search can be avoided altogether, thus making for more local (and efficient) computation. Suppose unification is adopted. Then, at stage (i), the PRT's and DO's still unvalued uninterpretable Case features can be unified together.[9] Subsequently, in stage (iii), PRT's Case feature will be valued at the same time as the DO's, without the search extension of (Chomsky 2001).

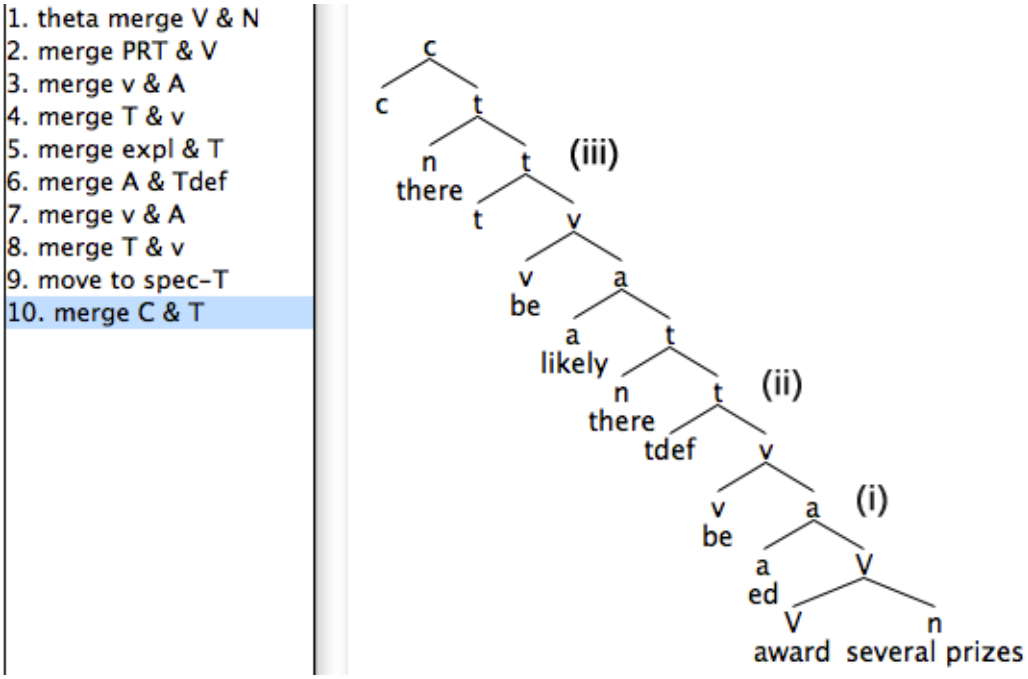

Figure 6: There are likely to be awarded several prizes

Consider next the derivation of (3.a), illustrated in Figure 6. The sequence of operations, stages (i-iii), largely parallels that of (3.b) discussed previously, except that the initial conditions will be different: in (3.a), the initial lexical array contains

[9] Implementation note: more precisely, an unvalued uninterpretable feature will have an uninstantiated logical variable as its value. Thus, when two unvalued uninterpretable features are unified, their values are represented by the same variable.

pleonastic *there*.[10] Instead of the DO *several prizes* raising to embedded tense as in (3.b), *there*-insertion via External Merge to embedded tense is triggered in (3.a).[11] At stage (iii), matrix tense (t) will probe and encounter (first) *there* at embedded tense. Both *t* and *there* have uninterpretable φ-features. Thus, unlike (3.b), matrix *t* must continue to probe all the way down until it encounters the *in-situ* DO *several prizes*, whereupon its uninterpretable φ-features, and the DO's uninterpretable Case feature, will be valued.

In the implementation, at this point several other operations must also complete for the derivation to converge properly. Prior feature matching by unification will result in adjectival –*ed*'s (PRT) Case feature being valued at the same time. Pleonastic *there*'s uninterpretable φ-feature will be also valued (as its φ-feature was already unified with matrix *t*'s). Without unification, feature matching must contain more steps. In particular, *there* must be a probe and directly enter into an agree relation with *several prizes*, after agreeing with (and probing beyond) PRT. Therefore unification provides for an efficiency gain since there will be fewer relations computed, despite matrix tense having to reach all the way down to find the *in-situ* DO.

Not only does unification requires less search to value the same uninterpretable features, it also simplifies bookkeeping from the viewpoint of recursive computation. As mentioned earlier, it is assumed that Agree applies as early as possible in the derivation as Merge proceeds. Once a probe has agreed with a goal within a (still partially assembled) syntactic object, there is no need to revisit (or later revive) that now-established relationship as syntactic object building proceeds in compositional fashion beyond initial Merge of the probe. In other words, there is no need to re-descend and check sub-object features: still unvalued uninterpretable features (linked earlier by unification) will be automatically valued at the earliest opportunity.

Consider now the derivation of (4.b), illustrated in Figure 7. Stages (i) and (ii) are similar to that of (3.b), illustrated previously in Figure 5. However, unlike (3.b), stage (iii) for (4.b) makes use of v* introduced along with the ECM verb *expect*. The probe v*, which values accusative Case, agrees with the DO *several prizes* previously raised to embedded tense.[12] In (Chomsky 2001), v* must continue to probe past the DO and value the Case feature of the PRT *-ed*. Assuming unification has already taken place in stage (i) between the unvalued uninterpretable Case feature for both PRT and DO, there is no need (in our implementation) to extend the domain of search for v* past embedded tense. Thus unification has a locality advantage with

---

[10] (Chomsky 2001) assumes *there* is φ-incomplete, containing only a person feature.
[11] In this implementation, it is assumed that External Merge is preferred (where available) to Internal Merge.
[12] Example pairs (3.a-b) and (4.a-b) differ with respect to the Case assigned to *several prizes*, nominative and accusative by matrix tense and v*, respectively. This difference is not manifested here. However, cf. *we expect him/*he to be nominated*.

respect to probe-goal search. To complete the derivation, we note that in stage (iv), matrix tense (t) probes and values the nominative Case for the matrix subject pronoun *we*.

Finally, in the case of (4.a), the derivation parallels that of (4.b) described above, with the initial conditions changed by the presence of pleonastic *there*, as with (3.a). At stage (iv) (not shown), v* agrees with *there* and must probe all the way down to have its φ-features valued by in-situ *several prizes*. Similar efficiency gains to (3.a) are realized here: in particular, *there* does not need to probe *several prizes* in this implementation.

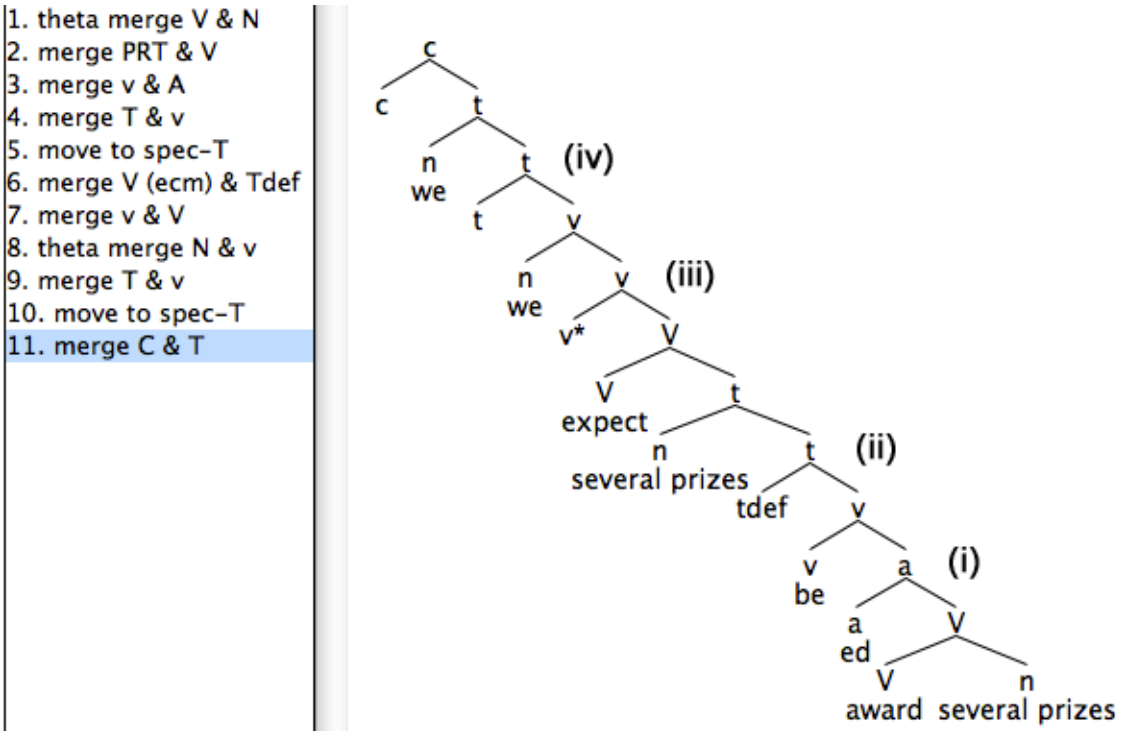

Figure 7: We expect several prizes to be awarded

References

Chomsky, N.A. (1981). *Lectures in Government and Binding*. Dordrecht: Foris.

Chomsky, N. A. (2001). Derivation by Phase. In M. Kenstowicz (Ed.), *Ken Hale: A Life in Language,* pp. 1–52. Cambridge MA.: MIT Press.

Fong S. (1991). *Computational Properties of Principle-Based Grammatical Theories*. PhD thesis, Artificial Intelligence Laboratory, MIT.

Stabler, E.P. (1997). Derivational minimalism. In: C. Retoré (ed.), Logical Aspects of Computational Linguistics (LACL '96), Lecture Notes in Artificial Intelligence Vol. 1328, pp. 68–95. Springer, Berlin, Heidelberg.

Vijay-Shanker, K. and D. J. Weir (1994). The Equivalence Of Four Extensions Of Context-Free Grammars. In *Mathematical Systems Theory*}, Vol. 27, pp. 27-51.