Introduction to Transformational Grammar, LINGUIST 601
October 1, 2004

Clausal Architecture and Verb Movement

1 Clausal Architecture

1.1 The Hierarchy of Projection

(1) a. John must leave now.
b. John must be leaving for the airport right now.
c. John must have left for the airport by now.
d. *John must is / been / being / has / had / having / ate / eaten / eating.

We have identified the location of modals with the location of finite Tense. This yields the following hierarchies:

(2) a. Tense $>$ V_{bare}
b. Tense $>$ be_{Prog}
c. Tense $>$ $have_{Perfect}$

We can conclude that the only restriction T^0 imposes on its complement is that it requires its complement to be bare. This is also true for the infinitival marker *to*.

(3) a. To leave now would not be fun.
b. To be leaving for the airport right now, John must have started early.
c. To have left for the airport by now, John must have started early.
d. *To is / been / being / has / had / having / ate / eaten / eating.....

This pattern can be summarized as follows:
T^0 must take a bare VP complement, where VP covers verbal projections headed by main verbs as well as auxiliaries.

There is now a question of which direction the selection goes in.

(4) Two Options:
a. Option 1: T^0 selects a verbal head that has the property of being inflectionally bare.
i. At point of Merge: T^0[uBare] . . . V^0[Bare]
ii. After Agree: T^0[~~Bare~~] . . . V^0[Bare]
b. Option 2: T^0 selects a verbal head that is inflectionally unspecified and specifies its inflection to be bare. This would be a property of T^0.
i. At point of Merge: T^0[Bare] . . . V^0[uInfl:—]
ii. After Agree: T^0[Bare] . . . V^0[~~uInfl~~:Bare]

(the points above generalizes to the progressive auxiliary *be* and the perfect auxiliary *have*.)

Both options are attested in the analyses of natural language phenomena - (4a) in agreement and (4b) in case-assignment. In what follows, we will adopt the option in (4a).

- A fully decompositional option is also conceivable.

1.2 Selection by Perf and Prog

Just like T^0 imposes selectional constraints on their complement, so do Perf and Prog.

(5) a. Gina has eaten / *eating / *ate / *eat the apple.
b. Gina is eating / *eaten / *ate / *eat the apple.

The above selectional constraint can be represented as follows (along the lines in (4a):

(6) a. Selection by Perf:
At point of Merge: have[...,uPerf] . . . eat[Perf]
After Agree: have[...,~~uPerf~~] . . . eat[Perf]
eat[Perf] is realized as 'eaten'.
b. Selection by Prog:
At point of Merge: be[...,uProg] . . . eat[Prog]
After Agree: be[...,~~uProg~~] . . . eat[Prog]
eat[Prog] is realized as 'eating'.

Some additional constraints:

(7) a. Perf > Prog:
Maya has been eating pizza all day.
b. *Prog > Perf:
*Maya is having been eating pizza all day.

We could, of course, encode the ungrammaticality of (7b) using selection. But the exact source of the ungrammaticality of (7b) remains unclear - it could be semantic in nature. Empirically though we end up with the following hierarchy:

(8) T^0 > (Perf) > (Prog) > V

A complete derivation:

(9) Angela must have been eating pizza all day.
a. [eat[Prog] all day]
b. be[Perf,uProg] [eat[Prog] all day]
c. Agree: be[Perf,~~uProg~~] [eat[Prog] all day]
d. have[Bare,uPerf] [be[Perf,~~uProg~~] [eat[Prog] all day]]
e. Agree: have[Bare,~~uPerf~~] [be[Perf,~~uProg~~] [eat[Prog] all day]]
f. must[Tns,uBare] [have[Bare,~~uPerf~~] [be[Perf, ~~uProg~~] [eat[Prog] all day]]]
g. Agree: must[Tns,~~uBare~~] [have[Bare,~~uPerf~~] [be[Perf, ~~uProg~~] [eat[Prog] all day]]]

Each of the heads as an uninterpretable feature and an interpretable feature - the interpretable feature specifies a property of the head and the uninterpretable feature a selectional constraint.

1.3 Formal Definitions of Merge and Agree

1.3.1 Merge

Merge is the most basic syntactic operation. We will distinguish between two kinds of Merge: set-merge, which introduces arguments, and pair-merge, which introduces adjuncts.

(10) Let K be the result of Merging α and β.

a. Set-Merge: K = set-merge(α,β) = $\{\alpha, \beta\}$.
When α and β set-merge, it is to satisfy requirements of one of them. If the requirements of α are being satisfied, α projects i.e. label(K) = label(α).

b. Pair-Merge: K = pair-merge(α,β) = $< \alpha, \beta >$.
Pair-Merge is inherently asymmetric - if the operation of Pair-Merge adjoins α to β to form $< \alpha, \beta >$, we can conclude that β projects i.e. label(K) = label(β).

(For any lexical item α, label(α) = α.

requirements of α: can be uninterpretable subcategorization features, as well as semantic requirements such as θ-roles.

(11) Let label(K) = α and β be a term of K introduced by set-merge.

a. β is a complement to α iff β is a sister of α.

b. Otherwise β is a specifier of α.

(β is a term of $K = \{\alpha', \beta'\}$ if $\beta = \alpha'$ or $\beta = \beta'$, or else β is a term of α' or β'.)

1.3.2 Agree

Agree is another important syntactic operation.

Consider agreement between a subject and its predicate. An uninterpretable feature F (ϕ-features) on a syntactic object Y (the predicate) is determined by another syntactic object Z (the subject) which bears a matching feature F (the subject's ϕ-features).

The operation of Agree will play an important role in our syntactic calculus and will extend beyond phenomena traditionally thought of as involving agreement.

(12) Agree is the operation by which a head X^0 (the Probe) with a set of unvalued uninterpretable features identifies the closest Y^0 / YP in its c-command domain with the relevant set of matching (i.e. nondistinct) interpretable features (the Goal), and uses the features of Y^0 / YP to value its uninterpretable features and vice versa.

• Even though the above definition is stated in terms of unvalued features, Agree may involve pure feature checking if the features involved are privative.

The following options are in principle available:

(13) ... indicates c-command.

a. Simple Feature Checking 1:
X[uG] ... Y[G] $\longrightarrow$ X[~~uG~~] ... Y[G]

b. Simple Feature Checking 2:
X[uG] . . . Y[uG] $\longrightarrow$ X[~~uG~~] . . . Y[~~uG~~]

c. Feature Valuing 1:
X[uF:] . . . Y[F:val] $\longrightarrow$ X[~~uF~~:val] . . . Y[F:val]

d. Feature Valuing 2:
X[uF:] . . . Y[uF:] $\longrightarrow$???

The following cases have also been used in the literature, but they do not fall under the definition in (12).

(14) a. Simple Feature Checking 3:
X[G] . . . Y[uG] $\longrightarrow$ X[G] . . . Y[~~uG~~]

b. Feature Valuing 3:
X[F:val] . . . Y[uF:] $\longrightarrow$ X[F:val] . . . Y[~~uF~~:val]

One possibility that has been explored is that the options in (14) might go hand in hand with the options in (13). The assignment of nominative to the subject by T^0 and agreement between T^0 and the subject could be seen as a case in point.

A locality constraint on Agree:

(15) *X[uF:] . . . Y[F:val1] . . . Z[F:val2] $\longrightarrow$ X[~~uF~~:val2] . . . Y[F:val1] . . . Z[F:val2]

1.4 The Location of Negation

Where can negation appear?

(16) a. John might not leave.
b. John did not leave.
c. John has not left.
d. John is not leaving.

(17) a. Lisa might not have been talking to Bill.
b. Lisa might have not been talking to Bill.
c. Lisa might have been not talking to Bill.

Out of (17a-c), only (17a) is felt to have a status different from (17b-c), and is often referred to as sentential negation as opposed to (17b-c) which are said to involve constituent negation.

Constituent negation can appear on a wide range of syntactic categories.

(18) a. PP: Minjoo was sitting not on the chair, but on the bar stool.
b. NP: Angela was reading not a comic, but a newspaper.
c. AP: One can be not happy and not unhappy at the same time.

The cut between sentential and constituent negation is ultimately a semantic one. But we can give the following syntactic characterization of the distinction:

(19) a. Sentential Negation: T^0 [Neg [....

b. Constituent Negation: every other instance of negation.

A minimal pair:

(20) a. You can't always do that. (sentential)

b. You can always not do that. (constituent)

c. You can't always not do that. (sentential and constituent)

• We will treat *not* as having its own projection NegP.

Two reasons for this:

(21) a. *n't*, a variant of *not*, can appear as a suffix on the verb:

i. Makoto hasn't left.

ii. Hasn't Makoto left?

b. a contrast between *not* and the negative adverb *never*:

i. Makoto did not leave./*Makoto not left.

ii. Makoto never left./*Makoto left never.

2 Basic Verb Movement

The Hierarchy of Projection:

(22) $T^0 > (Neg_{sentential}) > (Perf) > (Prog) > V$

According to the hierarchy of projection the perfect auxiliary *have* and the progressive auxiliary *be* are below negation. This seems right for some cases but not for others.

(23) a. Aniko might not have been eating pizza.

b. Aniko has not been eating pizza.

c. Aniko is not eating pizza.

d. Aniko did not eat pizza./*Aniko ate not pizza.

Similar facts hold with the negative adverb *never*.

(24) a. Aniko might never have been eating pizza, but for...

b. Aniko has never been eating pizza.

c. Aniko is never eating pizza.

d. Aniko never ate pizza./*Aniko ate never pizza./*Aniko did never eat pizza.

The facts about *do*-support are actually more general than the above discussion might indicate. *do*-support is triggered by polarity items other than *not* also, such as *so* and *too*.

(25) Negation and other polarity items

a. Danny does not like Sasha.
(*Danny not likes Sasha.)

b. Danny does so like Sasha.
(*Danny so likes Sasha.)

c. Danny does too like Sasha.
(*Danny too likes Sasha.)

This can be represented by modifying the hierarchy of projection as follows.

(26) The Hierarchy of Projection (revised):
$T^0 > (Pol) > (Perf) > (Prog) > V$
(Pol^0 projects PolP, sometimes also referred to as Σ / ΣP)

The generalizations that we can infer from the above set of facts (assuming the hierarchy of projection) are:

(27) a. If we can insert lexical material under T^0 (e.g. modals and *to*), everything else stays put.

b. If nothing is directly inserted under T^0 and T^0 has PerfP / ProgP as its sister, then *have/be* must move to T^0 and realize the features of T^0.

c. If T^0 has VP as its sister, then the features of T^0 are realized on V^0.

d. If T^0 has PolP as its sister, we have two subcases:

i. PolP immediately dominates PerfP / ProgP: *have/be* moves to T^0 and realizes the features of T^0.

ii. NegP immediately dominates VP: *do* is inserted under T^0 and realizes the features of T^0.

do-support has a Last Resort nature - it only applies when all other options have failed. If we insert a *do* in environments where *do*-support is not forced, we get ungrammaticality.

(28) a. *John doesn't be eating pizza. (vs. John isn't eating pizza.)

b. *John does be eating pizza. (vs. John is eating pizza.)

c. *John doesn't have eaten pizza. (vs. John hasn't eaten pizza.)

d. *John does have eaten pizza. (vs. John has eaten pizza.)

Give that *John eats pizza* is acceptable, why is the following still acceptable? and why does the *do* have an emphatic reading?

(29) John does eat pizza. (only emphatic reading is available)

One way of thinking about these facts is as follows:

(30) a. The features of T^0 need to be expressed on an overt host.

b. If a modal is merged into T^0, the features of T^0 have a host.

c. If the highest verb is auxiliary *have/be*, the verb moves into T^0 and provides a host to the features of T^0. The presence of Pol^0 does not block this movement.

d. If the highest verb is a main verb, the main verb cannot move to T^0. Now there are two options:

   i. If Pol^0 does not intervene between T^0 and V^0, the features of T^0 are realized on V^0.

   ii. If Pol^0 intervenes between T^0 and V^0, a *do* is inserted under T^0 and the features of T^0 are realized on *do*.

(roughly the proposal in Bobaljik (1995))

An implementation of the above idea given our assumptions about Agree:

Basic intuition: the realization of the features of T^0 is distinct from the categorial selection of T^0 for a bare VP complement.

Assumption 1: The features of T^0 can be realized only on a verbal head that is the sister of T^0 or the head of the sister of T^0.

Assumption 2: Modals are generated under V^0, but must combine with finite T^0 to be pronounced.

Assumption 3: *not/so/too* occupy the [Spec,PolP], *-n't* occupies the head of PolP. *never* does not involve projection of a PolP.

Assumption 4: Certain kinds of features cannot be checked by Agree alone - they require movement. Such features are called strong features, sometimes indicated by *. Thus F* would be the strong version of F.

(31) a. The [unInfl:Tense] feature is strong for *have, be* and modals and weak for main verbs.

b. The presence of a NegP does not block T^0 from causing verbs with strong uInfl:Tense features (modals, *have, be*) from moving up to T^0. Main verbs have weak features and stay put.

c. Now the features of T^0 are realized on its verbal sister or the verbal head of its sister. If a verb has moved to T^0, it counts as a sister of T^0 and the features of T^0 are realized on this head. If nothing has moved to T^0 but T^0 has a verbal complement, then the features of T^0 are realized on this verbal head.

d. In the case where the complement of T^0 is a PolP and no verb has moved into T^0, neither of the above options are available. In this case, a *do* is inserted under T^0 and the features of T^0 are realized on *do*.

A unsolved mystery:

We can say the following:

(32) a. You can't always do that. (sentential)

b. You can always not do that. (constituent)

c. You can't always not do that. (sentential and constituent)

We can also say:

(33) You didn't always not do that.

But what is the positive version of (33)? We can say (34), but that seems to be emphatic only.

(34) You did always not do that.

• it seems constituent negation blocks realization of features of T^0 on the V^0, but does not trigger *do*-support leading to ineffability. See Embick and Noyer (2001) for details.

References

Bobaljik, J. D. (1995) Morphosyntax: The Syntax of Verbal Inflection, Doctoral dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts. Distributed by MIT Working Papers in Linguistics.

Embick, D., and R. Noyer (2001) "Movement Operations after Syntax," Linguistic Inquiry 32:4, 555–598.