Sprachwissenschaft in Frankfurt

Arbeitspapier Nr. 17

Economy

Daniel Büring

Frankfurt a.M.
Dezember 1996

Schutzgebühr DM 5,-

Johann Wolfgang Goethe-Universität Frankfurt a.M.
Institut für Deutsche Sprache und Literatur II

Contents

Formulating economy conditions...we derive a fairly narrow and determinate notion of most economical convergent derivation... Precise formulation of these ideas is a rather delicate matter, with a broad range of empirical consequences (Chomsky 1995:201)

0. Introduction*

Among the mechanism and concepts proposed or considered in the papers collected in Chomsky 1995, the concept of economy presumably presents the most radical departure from earlier conceptions of Chomskian generative grammar. Most of the proposals within the Minimalist Program (henceforth MP) can - from a broader perspective - be conceived of as mere reformulations or modifications of principles known from the framework as presented in the *Lectures on Government and Binding* (Chomsky 1981, LGB) and *Barriers* (Chomsky 1986) (or attempts thereat): X-bar theory is reduced to bare phrase structure, locality is uniformly accounted for by minimality, government, argreement, and certain other tree geometric configurations are subsumed under spec/head agreement, and, finally, the operative levels are restricted to LF and PF.

As opposed to that, economy principles, first discussed in Chomsky 1991 (distributed as early as 1989) have no predecessors in Chomsky's earlier work.[1] Maybe as a consequence of that, Chomsky 1995 presents a broad range of possible economy principles, which are subject to

---

* This article is part of the DFG project 'Ökonomieprinzipien der Syntax' at Frankfurt University. Thanks go to Günther Grewendorf and Joachim Sabel there. My colleagues at Cologne University have helped me to see clearer many aspects of the Minimalist Program in our spring 1996 reading group. Special thanks to Alexandra Zepter who read the manuscript and proved with her comments to be a real expert in formal syntax.

[1] In a recent LINGUIST list discussion (issue 7.458ff), Yehuda Falk pointed out that predecessors of economy can be seen in the evaluation procedures of earlier generative grammars, i.e. procedures which were assumed to be active in the language learner to chose between competing and descriptively adequate grammars. Yet economy as a part of the 'steady state' rather than the Language Acquisition Device appears to be completely novel in Generative Grammar.

constant modification, reformulation and sometimes abandonment during the course of the exposition. Accordingly, other authors, too, radically differ in the concepts and principles they invoke or take to be operative, both among each other and from the proposals discussed in the MP.

The aim of this paper is to present the core ideas and concepts of economy in a self-contained way and discuss their properties and consequences (as for the general concepts of the MP, which I presuppose here, see e.g. Marantz 1995). Moreover the paper is intended as a guide to this aspect of Chomsky 1995, overcoming, I hope, some of the difficulties readers might have in facing Chomsky's rather unorthodox style of presentation. I have tried to present the core arguments, discuss the examples found in Chomsky 1995 and provide the reader with as many references to the original text as possible.[2]

Hopefully this paper can also serve as a basis for further inquiry into questions about economy: What are the operative principles? In which way can they be reduced so as to avoid redundancy within the principles of grammar? Which formal properties do they have to have? And finally, how strong is the case for economy conditions actually?

Towards this aim, I have decided to start - after a brief exposition - with some rather general remarks on the architecture of grammars that use economy principles (section 2). The concept of transderivational constraints, which plays a crucial role in the MP, especially in its economy part, is introduced and discussed. In the following section 3 I present the version of economy as it takes shape at the end of chapter 4 of Chomsky 1995, the final version, if you like. The idea is to start with those principles that are doubtlessly relevant at the end of the discussion, omitting those that only serve expository purposes and are reformulated or abandoned during earlier chapters. This, I hope, will facilitate the readers understanding of the 'state of the art'. I should warn the

---

[2] I refer to the single chapters as MPLT (*A Minimalist Program for Linguistic Theory* = ch.3) and C&T (*Categories and Transformations* = ch.4). Page numbers refer to the book edition (Chomsky 1995) throughout.

reader, though, that section 3 presents nothing more than *my* conception of what the final version looks like. Since Chomsky 1995 leaves quite some room for interpretation, my presentation of those ideas may be far from accurate. I therefore incorporated quite some *but*'s, *if*'s and *maybe*'s in the form of side remarks (indicated by different typeface) and footnotes, which should presumably be skipped upon first reading.

Following that, section 4 discusses those principles that are discussed during the course of exposition but do not seem to have survived until the end of the book. It should help readers to familiarize with certain concepts found in the literature and at the same time see how and to which degree they are actually subsumed by the version of the theory presented in section 3.

Section 5 concludes the discussion and tries to weigh the arguments presented by Chomsky that speak in favor of incorporating transderivational constraints such as *Procrastinate*. The appendix 6 then discusses one of the more technical details, namely the question of locally implementing transderivational constraints. I give a partial formalization of economy systems in order to facilitate discussion of their properties. This discussion is rather sketchy and presumably raises more questions than it settles. Nevertheless I believe that these questions have to be posed and dealt with if the discussion about computational complexity is to make sense.

1. Economy and the Theory of Grammar

Let me start with a rather straightforward (though not necessarily clear) issue: economy of representation (MP 151, 200): Interface representations may not contain superfluous symbols; no expletives at the level of Logical Form (LF), no traces at Phonological Form (PF) and so forth. This aspect of economy is part of the general *Principle of Full Interpretation* (FI for short, MP:200). Though its exact content is open to debate, I will not be concerned with it here.

Our central concern is *economy of derivation*. The idea

is simple enough: In a framework such as LGB everything could move everywhere, unless this movement (or the resulting configuration) was explicitly banned by some wellformedness principle. Certain movements had to take place (e.g. object raising in passives) but only because avoiding them would have lead to violation of some independent principle (in the case of passives the Case Filter or the Extended Projection Principle, EPP), thereby making the structure without movement ungrammatical. Other movements were conceived of as truly optional: particle shift, topicalization, dative shift, scrambling, object shift in the Scandinavian languages among others.

Within the MP movement has to be *triggered*. A grammar within the LGB framework has to exclude the possibility that in English, multiple *wh*-movement takes place; or that German scrambling crosses finite clause boundaries; or that verbs in embedded questions move to COMP. An MP compatible grammar has to make sure that movements take place at all. For one central dogma of the MP is:[3]

(1) Laziness: Do not move!

There is *prima facie* only one reason to move despite Laziness, namely *checking of features*. It is not clear what features there are, but among them we certainly find categorial features (N and V), case, agreement (= phi-features) and a feature *wh* or *Q*.

A garden variety case of triggered movement goes like this: Some functional head, say AgrS, has some feature, e.g. agreement: 3rd person singular. In order to generate a wellformed sentence, some element with that very feature (i.e. a DP) has to enter into a *checking relation* with AgrS during the course of derivation, where a checking relation is by assumption the spec/head configuration. So DP moves to SpecAgrS and the pertinent features are checked. Successful feature checking in the case of agreement features results in deleting those features on the

[3] The 'principles' introduced here are not part of grammar (though they could be, and some authors take them to). Here, as in Chomsky 1995, they only serve expository purposes. For the virtually active principles, see sections 3f.

functional head $AgrS^0$. In case those features do not get checked at some point, the derivation *crashes*. A non-crashing derivation is called *convergent*. Thus, Laziness is only violated in order to avoid crashing of a derivation. And vice versa: DP can only move to SpecAgrS because that feature had to be checked. No feature, no movement! We say that movement is *feature driven*.

(2) Movement serves the sole purpose of feature checking

This is called the principle of *Last Resort* (LR) in the MP, although it is not defined at any point.

> At first glance (2) is just a consequence of the combined efforts of (1) and the concept of convergence: Suppose that checked features are the crucial prerequisite for convergence, and that Laziness is understood as a global principle: 'How many superfluous moves have been made in this derivation?' Then it follows that a derivation can only be 'truly lazy' or 'the laziest' if no element has moved without the need for feature checking. And in fact Last Resort is often used in that way: as a description of the overall effect resulting from convergence and Laziness. We will see below, however, that this is not the interpretation Chomsky has in mind, but that's already one of the economical subtleties of the MP. In this section we will therefore pretend that 'α moves for the sake of convergence' and 'α moves to check its own features' are equivalent statements.

Let us start with the case of overt movement. As for covert movement it is assumed later in C&T that it doesn't affect entire words ('categories' in Chomsky's terminology) but only features. We therefore ignore this case here. Overt - as opposed to covert movement - is triggered if the pertinent feature has to be checked and deleted at Spell-Out. Such a feature is called *strong*. Features that aren't strong are *weak*. Strong features occur by assumption only on non-lexical categories, i.e. AGR, TENSE, COMP, DET and - surprisingly - *v*, the category of light verbs. Whether or not some feature on a functional head is strong is a question of parametric variation.

Strong features thus trigger overt movement. And vice

versa, weak features never trigger overt movement. The reason for the latter is the principle *Procrastinate*.

(3) Procrastinate: Covert movement is cheaper/ more economical than overt movement.

Due to Procrastinate covert movement is always preferred, if possible: Ideally you don't move at all (Laziness); only move for the sake of feature checking (Last Resort); and if you do, do it covertly (Procrastinate); overt movement only if strong features are involved.

> It might come as a surprise that strong features justify a violation of Procrastinate. In MPLT it is assumed that strong features cannot be interpreted at PF and therefore cause a crash. Procrastinate is thus violated in order to ensure convergence, just like Laziness. In C&T this explanation is abandoned. The ban against strong features in the covert component is simply stipulated (MP:233).

2. Introducing Economy and Transderivational Constraints

Before we come to the details of the MP economy system it is useful to clarify some terminology. And this in turn requires some theoretical prolegomena.

Suppose we simply augment an LGB-system with (3), the Procrastinate principle. Would that make much of a difference? Of course, optional movements wouldn't happen (unless new features are introduced to trigger them, see below). Wh-movement would take place so as to satisfy the wh-criterion (May 1985, Rizzi 1990), A-movement would take place for DP to get case, and various head movements would take place because the pertinent heads are affixes that would otherwise violate Lasnik's filter (what Baker 1988 calls the *Stray Affix Filter*). In general, things wouldn't change grossly.

How would we define Laziness? Maybe as in (4)

(4) * $\alpha_1 \ldots t_1$, unless
   a. there is an X that entertains a checking relation R with α, and
   b. there is no Y that entertains R with t

The filter in (4) has to hold for each movement step. Or it has to be checked for every two-membered chain link.

One can also imagine a completely different definition for Laziness, such as in (5).

(5) No derivation may contain more movements than necessary.

What does *necessary* mean? Maybe...

(6) A movement M is *necessary*, if without M the structure would end up ungrammatical.

(6) contains a counterfactual conditional. How can we know whether the structure would end up ungrammatical without movement M? The answer is: Try! We thus replace (6) by (7).

(7) A movement M is *necessary* if there is no grammatical derivation without M.

The problem with (7) is that there is presumably always some derivation that can avoid the pertinent movement. What we really want is that *this* derivation wouldn't end up grammatical, if we didn't move. We therefore need an *equivalence class* of derivations, none of which may get by without movement M. That's what Chomsky calls a *Competitor Set*.

Within the LGB framework the notion of Competitor Set would have been straightforwardly definable using the notion of d-structure: derivation D' is a competitor to D iff D' and D have the same d-structure. With d-structure gone we have to use its MP counterpart, the *Numeration*:

(8) For each derivation D, CS(D), the *Competitor Set* of D, contains all D′ whose initial numeration equals that of D.

Then the final version of *necessary* would be:

(9) Movement M is *necessary* if there is no derivation D′ in the Competitor Set of D that is grammatical and does not contain M.

Our principle (5) can now be reformulated as (10).

(10) A derivation D is ungrammatical if it contains unnecessary movements.

Hold it! These definitions are circular: In (9) necessity is defined using the notion of grammaticality, and in (10) grammaticality is defined using the notion of necessity: In order to classify a step as *necessary*, one has to consider the competing derivations, more precisely the grammatical ones among them. But in order to know whether or not a competing derivation D′ is grammatical one has to know - due to (10) - if it contains unnecessary steps. And therefore we have to consider the competing grammatical derivations...

There are two ways out of this dilemma. First, we can suspend with the clause '...that is grammatical...' in (9). Then every derivation D′ is a competitor as long as it is built on the same numeration, regardless of whether D′ is actually wellformed. But that is certainly dangerous. I can always leave out some movement if I don't have to care for grammaticality. In fact there is *always* a competing derivation that doesn't need the movement in question (presumable one that doesn't involve movements at all) and therefore movement would *always* be prohibited. That's what Chomsky calls an unwelcome result. We thus forget about this possibility.

Or, the second possibility, we replace *grammatical* in '...that is grammatical...' with some other property. Such a property should be '...that is grammatical, economy considerations aside.' Chomsky calls this property

convergence. We thus redefine:

(11) Movement M is *necessary* if there is no derivation D' that converges but doesn't contain M.

(12) A derivation is *convergent* if it meets the operative wellformedness constraints (all features checked, ECP, subjacency, X-bar or whatever)

(13) A derivation is *economical/grammatical* if it is convergent and contains no unnecessary step.

These are almost all ingredients for a working economy system. Still somewhat vague is the notion of *movement* in (11). To see this, take a concrete example: Suppose we want to know whether moving some wh-word was necessary. There are now at least two ways of interpreting (11):

(14) Movement M is *necessary* if
  a. ... there is no derivation in the Competitor Set that doesn't contain wh-movement.
  b. ... there is no derivation in the Competitor Set that contains movement of *this* word into *this* position

Interpretation (14.a) seems nonsensical: We do not want to consider any old wh-movement within the structure, but only those we are just concerned with. (14.b) looks better in that respect. Given a structure S we ask whether the movement under discussion can be avoided in any of the competing derivations. However, we are still not precise about what 'movement' is, or more precisely, when two movements in two derivations are called the same.

A straightforward way to deal with this would be to define movements as a transition from one phrase marker to the other. For example, movement of the subject from SpecV to SpecAgrS would be identified with a pair of phrase markers, roughly $[_{VP}$ Subj V Obj] and $[_{AgrS}$ Subj AGR $[_{VP}$ *t* V Obj]]. We then interpret (14.b) in terms of (14.c):

  c. ... there is no derivation in the Competitor Set

> that doesn't contain the transition from structure S to structure S'

where S and S' are the phrase markers before and after movement M, respectively.

Presumably necessity of movement, and - in turn - economy, can be defined in these terms, although non-trivial problems arise. Let us, however, turn to a completely different and considerably simpler implementation of economy principles first, one that is assumed in - though not throughout - the MP. This version will suffice to illustrate the general issues. We will then return to the peculiarities of making reference to movement steps in subsection 2.2. below.

The idea is this: If two derivations D and D' are within the same Competitor Set and one of them, say D, contains one or more unnecessary steps (i.e. steps that D' doesn't contain), it follows that the total number of movements in D must be bigger than the total number of movements in D'. The most economical derivation is thus the one with the smallest overall number of derivational steps. We thus abandon the concept of being *necessary* and define economy directly as follows:

(15) Derivation D is grammatical if there is no derivation D' in the Competitor Set to D such that
  a. D' converges
  b. D' contains less steps than D

This condition is by no means equivalent to the one envisaged before. In principle both D and D' may contain superfluous steps, as long as it is the same number. But (15) does what it should: It prevents elements from moving if they don't have to.

If there is a derivation D' (in the sense of definition (15) above) which is 'shorter', D is not grammatical. We say that the shorter derivation D' *blocks* the longer D.

Constraint like (15) (as well as (11)/(13) from above) are genuine *transderivational* constraints (TDCs). They can

only be formulated using classes of competing derivations, our Competitor Set. Note that transderivational constraints and economy constraints are not necessarily the same. There are economy conditions that are not transderivational (e.g. the final version of the Minimal Link Condition - see below - or Relativized Minimality[4]). Likewise there could be transderivational constraints that are not economical (replace for example *less* in (15) by *more* to get an anti-economical transderivational constraint). Constraints that are not transderivational will be called *intraderivational constraints* (IDCs).

Since there are no transderivational constraints in LGB-type grammars it is worthwhile to take a closer look at the architecture of systems that incorporate transderivational constraints. One of their characteristics is the distinction between two different concepts of wellformedness, in this case convergence and what I will refer to as grammaticality. The system thus looks as in (16).

(16)

(i) numeration
|
(ii) generation
|
(iii) interface conditions
(=IDCs that define *convergence*)
|
(iv) economy conditions
(=TDCs that define *economy/grammaticality*)

It follows from our above considerations that there have to be at least some intraderivational constraints within the MP that make up (iii), the interface conditions. Otherwise some non-convergent derivation (e.g. one in which everything remains in situ) would inevitably block everything else within the Competitor Set. Within the MP

---

[4] Rizzi's (1990) Relativized Minimality can be read as an economy constraint, since it virtually forced shortest movement (see MP:181 for this point). Yet it is - in Rizzi's original formulation as well as in the final version presented in the MP - formulated purely intraderivational.

these are first of all *Full Interpretation* and some other conditions (e.g. θ-assignment[5], empty numerations among others). But that does not necessarily imply that *all* intraderivational constraints are ordered before (iv). And in fact it is assumed within the MP that some intraderivational constraints operate *after* the economy conditions. The complete picture looks thus more like in (17).

(17)

(i) numeration
|
(ii) generation
|
(iii) interface conditions
(=IDCs that define *convergence*)
|
(iv) economy conditions
(=TDCs that define *economy*)(i) Numeration
|
(v) other constraints
(=IDCs, that define *grammaticality*)

(v) contains e.g. the ECP in Collins 1994. Chomsky uses the word *gibberish* to refer to derivations that pass (iv) but are nevertheless deviant. The terminology we are using here, and in contradistinction to the MP, makes a difference between *economical* and *grammatical* derivations.

But what difference does it make whether some intraderivational constraint belongs to group (iii) or to group (v)? Why, that is, must there be such a thing as gibberish? The answer is rather prosaic: constraints in (iii) codetermine convergence, but those in (v) don't. Suppose some derivation D has left a DP within VP, lacking case. It thus violates one of the interface conditions in

---

[5] Not the θ-criterion, which no longer exists. Chomsky writes: 'There is no need for the ... θ-Criterion at LF. A convergent derivation might violate them, but in that case it would receive a defective interpretation.' (MP:200) But later one it is explicitly stated that receiving a θ-role and possibly also assigning ones θ-roles is a necessary prerequisite for convergence. In that sense they are intraderivational constraints, belonging to (iii) in the sense of schema (17) below (MP:347; MP:315f see also the discussion on page 27).

(iii) since it contains unchecked features, but it wins out on economy since it has avoided one movement (DP to SpecAGR). As opposed to that D′ is just like D except that DP has raised and all features are checked. Then the fact that D is shorter, i.e. more economical, doesn't matter at all. D doesn't *converge* in the first place (= it doesn't meet (iii)), hence it is not in the Competitor Set relevant for (iv). D′ has dropped out of the running before economy considerations even arise. D′ cannot block D.

An *ungrammatical* derivation - call it D" - on the other hand (e.g. gibberish) may well block one that meets all constraints in (v). That is, D" kicks out D in the economy contest (iv) only to find that D" itself is ungrammatical in terms of (v). In a sense there are no survivors at all: D is uneconomical (since it loses out to D"), but D" is ungrammatical. We will see examples of this situation in section 3.2. below.

Summing up we find that there are various reasons why a sentence can be starred:[6]

(18) Sentences that a system such as (17) classifies as 'not belonging to language L':
  a. sentences that do not converge
  b. sentences that are not economical
  c. sentences that are not grammatical

Finally we should note that there can be different reasons why a derivation does not converge. Besides the interface conditions ((iii) in (17)) there are conditions that are checked in the course of generation. In the LGB days those

---

[6] Cf. MP:220: 'A language [sic!] L ... generates three relevant sets of computations: the set D of derivations, a subset D_C of convergent derivations..., and a subset D_A of admissible derivations of D. FI determines D_C, and the economy conditions select D_A. ...economy considerations hold only among convergent derivations; ... Thus, D_A is a subset of D_C." (MP:220). Chomsky ignores principles in the sense of (v) in (17) here, i.e. those that tell sentences from gibberish. It should be noted though that gibberish has no psychological counterpart such as 'semantically deviant'. Sentences so classified are usually just as unacceptable as those that are uneconomical or simply non-convergent. Gibberish is just a genuinely technical notion, despite its colloquial appearance.

were d- and s-structure conditions as well as conditions that had to be met with every single step of the derivation, e.g. subjacency, the Head Movement Constraint (HMC), X-bar conditions (structure preservation) and so forth. The MP prohibits d- and s-structure constraints, so we are left with 'each-step'-conditions such as subjacency. Chomsky calls these *derivational constraints* since they are checked during the derivation (or rather: during the generation in the sense of (17)). Constraints that apply at only *one* level of representation (FI, ECP...) are called *representational*.

Note that the terminology might lead to confusion here: *Derivational* contrasts not with *transderivational*, but with *representational*. I will therefore use the word *stepwise* in the remainder where Chomsky would use *derivational*:

(19) a. stepwise (Chomsky's derivational; applies with every step during generation) vs. representational (applies at designated level(s) of representation only)
b. transderivational (compares different derivations) vs. intraderivational (looks only at the derivation under construction)[7]

We thus revise (17) once more, yielding (20).

[7] At first glance, 'stepwise' and 'representational' are relevant subgroups of intraderivational constraints only, for it seems that transderivational constraints, by definition, cannot be stepwise (in fact they aren't representational either, comparing entire derivations rather than single representations). As we will see below, however, there are in fact stepwise transderivational constraints.

(20)

(i) numeration
|
(ii) generation
(respecting stepwise IDCs)
|
(iii) interface conditions
(=IDCs that define *convergence*)
|
(iv) economy conditions
(=TDCs that define *economy*)
|
(v) other constraints
(=IDCs that define *grammaticality*)

Stepwise constraints as in (ii) are mostly built into the definition of Move/Attract at the end of C&T, among them the Minimal Link Condition (MLC) and Last Resort, as well as the X-bar-substitutes of the bare theory.[8]

There are some rather exotic variants within the MP such as transderivational stepwise constraints (see 2.2., 3.2.1., and 6), or stepwise constraints that show effects only in (v) (see 3.3.2.). Both do not seem essential, though, and we will come back to those later on.

One more remark on stepwise constraints. In LGB the issue never arose as to what it means that a derivation violates some stepwise constraint. Take subjacency: A completed derivation may contain one or several subjacency violations. In a sense, grammar has to keep track of the bounding nodes/barrier that were illicitly crossed during the derivation, possibly using something like:

(21) A derivation D violates subjacency if there is some step in the derivation (some phrase marker) in D, in which a trace is separated from its

[8] Whether a stepwise constraint is built into the operations that generate derivations - e.g. Attract/Move - or formulated as a condition on the output of that operation is much of a muchness. Chomsky seems to envisage a theory in which stepwise constraints solely occur within Attract. The only candidate for a freestanding stepwise constraint at the end of C&T seems to be the extension requirement.

antecedent by some barrier.[9]

Definition (21) sounds like a representational constraint again, but note that there is a universal quantification over steps in the derivation, which is always equivalent to a stepwise constraint. What is important is that subjacency violations must be 'noted' during the generation, but the generation goes on (this is essential at least in those theories that predict degrees of strength in subjacency violations, considering a single violation as 'mildly ungrammatical' or not ungrammatical at all).

Within the MP it is assumed that violating a stepwise constraint causes the generation to stop. Or, to be precise, *that a step that would violate a stepwise constraint cannot be generated in the first place, because the stepwise constraints are part of the definition of the pertinent operations (Move/Attract or Merge)*. For example, a derivation that violates the Minimal Link Condition cannot be generated to begin with.[10] If that happens to hold for *every* stepwise constraint within the MP (see again 3.3. for a potential exception) we could derive an interesting corollary:

(22) Stepwise constraints cannot interact with economy conditions because they block convergence.

In other words, if a stepwise constraint is formulated in the way given in (21) it must belong to category (iii) in the sense of (20).

Another characteristic feature of all economy based formalism is that they involve the concept of Competitor Set. Above it was suggested that defining Competitor Sets via d-structure or numeration is the most straightforward way, but this is not really true. In general the Competitor Set defines what counts as a 'purpose' within the grammar. Hence, if identity of features is a defining property of

---

[9] This definition fails to register the *number* of subjacency violations, but that can easily be overcome.

[10] That implies that there are two classes of ungrammatical expressions: sentences that cannot be generated in the first place, due to the internal workings of (ii). And sentences that can be generated but are subsequently filtered out by either (iii), (iv) or (v). Subjacency violations, if in fact reducible to the MLC, belong to the former category. There is no straightforward way to have the grammar produce those sentences and give them, say, a question mark. Whether or not this result is problematic depends in part on one's conception of the competence-performance relation, see also appendix 6.

Competitor Sets this expresses the idea that checking a feature (a.k.a. movement) is a useful purpose. If some derivation involves such checking, it would be unfair to compare it to another one that doesn't contain the triggering feature and say: 'But this derivation is more economical!'

This might sound trivial but note that every economy condition is based on exactly this: distinguishing between useful or motivated and unmotivated or superfluous movements. And what counts as useful is determined by the definition of the Competitor Set: useful movements correlate with some defining property of the competitor set. Thus Chomsky claims that 'intelligibility' is not a useful purpose. Accordingly, semantics have no influence on competitor sets, hence movement cannot take place for solely semantic reasons (unless these are reflected by some feature, of course).

Some researchers (e.g. Fox 1995, Kitahara 1993, Reinhart 1994) take it that semantic effects *are* a useful purpose. For example, according to Fox quantifier raising is motivated by the need to change scope relations. And changing scope relations *is* a useful purpose in itself. Accordingly Fox defines the Competitor Set using truth conditions: Two derivations are competitors only if their LFs yield the same truth conditions. Two derivations with different truth conditions (say with and without raising of an object quantifier over a subject quantifier) are not within the same Competitor Set, hence neither can block the other.

In general, then, movements (in general: operations) in an economy system can be triggered or justified by almost anything, not just feature checking. The Competitor Set is the place where this is defined (for more discussion see Sternefeld 1996).

2.1. Excursus: Economy, Transderivational Constraints and Optimality

In the above examples we have concentrated on a single economy principle, Fewest Steps (FS). We have *de facto* identified the notion of economical with 'containing the

fewest number of steps'. But grammars may well contain several transderivational constraints, just as there can be several interface conditions or several grammaticality requirements. Within the MP at least two other transderivational constraints are discussed, the Minimal Link Condition and Procrastinate (which is not to say that these principles wind up as transderivational constraints at the end of C&T or at all). Take the following version of Procrastinate

(23) Procrastinate
Derivation D meets Procrastinate if there is no competing convergent derivation D' that features a smaller number of overt movements than D.

We are not interested at this point in how to formally implement this principle. What is relevant here is the question how Procrastinate and Fewest Steps - revised here - interact.

(24) *Fewest Steps:*
Derivation D meets Fewest Steps if there is no competing convergent derivation D' that contains a smaller number of movements than D.

Procrastinate and Fewest Steps look akin but they are logically independent. One can imagine a situation where two derivations D and D' are in the same Competitor Set, and only D meets Procrastinate while only D' meets Fewest Steps (say D' has one overt movement, D has two covert ones, every D" has three or more). What will happen? Maybe both derivations would be classified as uneconomical and receive a star. That would follow from a definition of economy like in (25).

(25) A derivation is economical if it meets Procrastinate and Fewest Steps.[11]

---

[11] One might ask whether Procrastinate should be called an economy principle (after all, postponing things does not usually save energy). Strictly speaking (25) defines the notion 'meets all transderivational constraints'. I will not attempt to define conditions for being an

But there is another possibility, one that is not being pursued in the MP:[12] One transderivational constraint might overrule the other. One might say, for example, that Fewest Steps is only relevant if Procrastinate is met to begin with. We would thus impose an ordering upon the transderivational constraints.

(26)

(i) numeration
|
(ii) generation
(respecting stepwise IDCs)
|
(iii) interface conditions
(=IDCs that define *convergence*)
|
(iv) economy conditions
(iv-i) Procrastinate
(iv-ii) Fewest Steps
|
(v) other constraints
(=IDCs that define *grammaticality*)

Notice that such an ordering is meaningless among intraderivational constraints: Whether, say, the ECP applies before or after the θ-criterion or the other way around makes no difference at all. But with transderivational constraints, ordering makes a crucial difference. In effect an ordering as in (26) states that Procrastinate (co-)determines the Competitor Set for Fewest Steps. The question of how transderivational constraints are ordered is thus as crucial as the question whether an intraderivational constrain figures in (iii) or in (v) (see

---

economy condition, which I am inclined to think would be to define a rather meaningless notion.

[12] Is it? In MP:201f it seems that Chomsky implies some ordering among the transderivational constraints; wrt. the MLC her writes: 'We thus assume that, given two convergent derivations D_1 and D_2 *both minimal and containing the same number of steps*, D_1 blocks D_2 if its links are shorter.' (MP:201f, my italics).

According to that the MLC would get to chose between derivations only if these derivations have already met Fewest Steps (and presumably Procrastinate). MLC would thus be low-ranked. But no such ordering or ordering effects are discussed later on.

above).

Intraderivational constraints can, however, be transformed into (almost) equivalent transderivational constraints (see Kim, Kolb, Müller & Sternefeld 1995, p.37, FN6).[13] Take the ECP. A standard formulation as an intraderivational constraints is given in (27).

(27) ID-ECP
Derivation D meets the ID-ECP, if at LF(D) all traces are properly governed.

The transderivational version looks like in (28).

(28) TRANS-ECP:
a. (auxiliary definition) ID-ECP: For any derivation D, ID-ECP(D) is the number of traces at LF(D) that are not properly governed.
b. TRANS-ECP: D meets the TRANS-ECP if there is no competing convergent derivation D′ such that ID-ECP(D′) < ID-ECP(D).[14]

This TRANS-ECP could now be ordered somewhere between the other transderivational constraints in (iv) in (26). The ordering among the transderivational constraints is called their *ranking*. The earlier a constraint appears in the derivation, the higher it is ranked. A system with ranked transderivational constraints is called an *optimality system*. The mother of all optimality systems is Optimality Theory (OT) as developed by Prince & Smolensky 1993. Syntactic applications can be found in Grimshaw 1993, Müller 1995 among others.

Optimality systems are thus a subset of the set of transderivational systems. Typical optimality systems in phonology have no or almost no intraderivational

---

[13] I said 'almost', due to the fact that intraderivational constraints can 'wipe out' a derivation entirely, i.e. no element in the Competitor Set meets the pertinent constraint, the numeration yield no wellformed output. Transderivational constraints always deliver an output, i.e. a non-empty set of optimal derivations.

[14] Note that the TRANS-ECP crowns the derivation with the smallest number of ECP violations. Such a system also allow for *ties*, just as desired: Two equally good/bad derivations do not block each other.

constraints: Each input leads to at least one output.

2.2. Local vs. Global Economy

In C&T Chomsky repeatedly claims that economy principles, in particular Procrastinate, should be implemented in a *local* fashion. Local here means that Procrastinate does not - as assumed above - look at the complete set of convergent competing derivations (the *global implementation*), but only at the next step within a derivation, deciding whether this step can be procrastinated until after Spell-Out or even altogether avoided. According to Chomsky such a local interpretation is preferable on conceptual grounds because it is taken to be computationally less complex. We will return to this issue in more detail in the appendix 6.

Suffice it to say at this point that there is no qualitative reduction in computational complexity if one assumes that, say, Procrastinate operates locally. The reason is plain to see: If Procrastinate, at a given point of the derivation, has to check whether some movement is necessary in order to ensure convergence, that means that it has to 'precalculate' all possible continuations of the derivation in order to see whether there is a convergent one among them that procrastinates movement. In other words, it has to locally calculate the Competitor Set. Again, the reader is referred to the appendix for details.

At this point we are interested in a technical detail with the local version of Procrastinate. Note that the local implementation does not take into consideration the overall number of overt movements but only the question whether a particular movement is necessary (overtly or at all). In other words, we are back at classifying individual movements as necessary vs. superfluous, or, in this case, urgent vs. postponable.

Above we said that the easiest way of precisely defining the notion of movement is in terms of a pair of phrase markers P and P', where P' is the result of moving some element in P. We also said that this will turn out problematic. Let us see why. Why can't we define (local) economy as exemplified with Procrastinate in (29)?

(29) Given a phrase marker P, movement M in P, resulting in phrase marker P′, is only allowed if there is no derivation D such that
  a. D contains P and
  b. D does not (overtly) contain P′

Clause (29.a) defines the Competitor Set. Since a derivation is nothing but a sequence of phrase markers, the set of continuations of P can safely be equated with the set of derivations containing stage P.[15] (29.b) then simply requires that the structure that would result from the movement under consideration is not found in D, at least not overtly.

This interpretation presupposes, though, that the order among all operations is fixed, a presupposition which isn't obviously true. Suppose for example that some derivation contains two movements that are in principle independent of each other, say wh-movement of DP to SpecC and I-to-C movement of the verb. The question is now: Can we move DP before V?

(30) a. John did see whom
  b. whom John did see t_{DP}
  c. whom did John t_{Aux} see t_{DP}

Now, there's a competing derivation in which V is moved first:

(31) a. John did see whom
  b. did John t_{Aux} see whom
  c. whom did John t_{Aux} see t_{DP}

Suppose this derivation converges. The transition from (30.a) to (30.b) does not happen in (31). Of course, there

---

[15] As we will see in the appendix, things are more complicated since the above definition ignores (i) the numeration (or what is left of it after P has been constructed), and (ii) the facts that a particular stage of a derivation does not necessarily consist of a single phrase marker, but of several which are being constructed in parallel. We will account for these complications in the appendix, but the general considerations remain the same, so for ease of exposition we can use the simplified assumptions made in the text.

is a very similar transition, (31.b) to (31.c), but formally it is not the same transition, since neither the input nor the output structure (P and P′ in the sense of (29)) are identical. Thus, derivation (31) avoids a transition that is included in derivation (30). Accordingly that transition is unnecessary in (30) and (30) is ungrammatical, according to (29) (although it is convergent). The more economical derivation blocks the less economical.

But vice versa, (30) blocks (31), too, again because of (29). The transition from (31.a) to (31.b) does not take place in (30), is thus unnecessary. The net result is that (30) and (31) mutually block each other. In general, if two derivations differ only in the order of application of some movement, they block each other. That's an unwelcome result, so (29) can't work that way.

Is local economy impossible to implement, then? We could try to define the notion of movement in a yet more elaborate way, say, by making reference to launching site, landing site, the moved element, and/or the movement path (i.e. the set of nodes that are c-commanded by the landing site and dominate the launching site). However, none of this 'book keeping' seems very minimalist. Furthermore we must take into consideration that the 'basic tree' is not unambiguously determined by the numeration. That is, an element α may occupy different positions in competing derivations even without movement (e.g. when there are multiple embeddings with numerous AGRs and DPs that do not differ in their relevant feature specification). Thus movement of the Subj to some AGR_x in one derivation might correspond to movement to some different AGR_y in a competing one. Yet we wouldn't want one to block the other.

It thus seems that we must ensure that the order of application of movements is in fact completely determined. For substitution this is already the case: By the extension requirement, substitution (i.e. movement to Spec) can only take place at the root node. However, this does not hold for adjunction, in particular not for head movement. We would thus have to imply an extraneous ordering, e.g. head

movement before substitution before (phrasal) adjunction. Given that, we can in fact use the above definition (29). In the concrete example, (30) would be ruled out by the Extended Extension Requirement and (31) would be the only - hence most economical - derivation (see the appendix 6 as to how this would work with several phrase markers which are constructed in parallel).

In sum, the local implementation of transderivational constraints is quite problematic (note that the constraint is still transderivational in the sense that it compares possible continuations). In subsubsection 3.2.1. we will look at the empirical consequences of the local implementation, appendix 6 deals with its properties wrt. computational complexity.

3. Economy at the End of Chapter 4

Let us now have a look at the theory of economy that takes shape at the end of Chomsky 1995.[16] It has two basic parts (and, it seems, only those): The operation *Attract*, which has a number of intraderivational constraints built into it, and the transderivational principle Procrastinate (see FN 35 and section 4.3., though).

[16] Towards the very end Chomsky discusses a modell that dispenses with AGR phrases in favor of multiple specifiers. Though this is largely irrelevant for considerations of economy in principle, some technical changes, especially in the definition of Equidistance, follow from that. I will ignore this option in the following text. Unless indicated in further footnotes, no substantial differences wrt. the issues discussed follow.

3.1. Attract

Attract is the mirror image of *Move* in earlier stages. Instead of the movement operation that takes an element α to, say, SpecAgr we now have the operation Attract that 'drags' α to SpecAgr. Two conditions govern this operation: i) α must have features that match those of AGR; and ii) α has to be the 'closest' category carrying that feature. 'Closest' here means something like 'separated from AGR by the smallest number of nodes'.

This new definition does not recur to the kind of features attracted. In C&T, Comsky distinguishes between [+interpretable] features and [-interpretable] features. Only the latter are dangerous at LF and must therefore be checked and deleted. Given Last Resort, a category would only move if its features are [-interpretable], regardless of the status of the features of the checker. Given Attract, the features of the checkee may be [+interpretable] or [-interpretable]. It thus doesn't matter that from the perspective of the moved category a [+interpretable] feature doesn't need to be checked (remember that [+interpretable] features do not cause trouble at the LF interface).[17] If a DP moves for reasons of case, or because its category is being attracted: So be it!

This suggests a new look at the examples that used to be blocked through Greed. In MP:216 principle

(22) Move raises α only if morphological properties of α itself would not be satisfied otherwise in the derivation

is illustrated using the following examples (MP ex. (23b), (24a)).

(32) a.* John seems (that) is intelligent
b.* there seems that a lot of people are intelligent

In (32.a) *John* overtly moves to matrix SpecAgrS to check the strong [D]-features of AgrS (plus weak AGR and Case as 'free riders'). In (32.b) *a lot of people*

[17] In other words, Greed (cf. section 4.2) is no longer a relevant principle.

would have to raise covertly to matrix AgrS to replace the expletive and to check agreement and case of AgrS. In these cases neither *John* nor *a lot of people* need to make this movement for their own sake. Their features have all been checked before. Movement would thus be altruistic. With Greed blocking such altruistic movement, (32.a) and (32.b) were thus correctly blocked (MP:261).

With Attract such an account is no longer viable. AGR wants [D] and case/agreement features, respectively, the embedded subjects have them, there is no closer feature, so 'movement ho!' Why then are the examples in (32) out? Presumably because *John* as well as *a lot of people* have already had their case features checked and deleted in the embedded SpecAgrS (case features are [-interpretable] hence delete). So raising them to matrix AgrS would not help to check that AGR's case features. The case features of matrix AgrS thus remain unchecked and cause the derivation to crash at LF (note that this presupposes that embedded AGR always has case checking features, which is not obvious).

This explanation furthermore presupposes that matrix AgrS needs to check case (and agreement). Plausible though this might seem, the discussion on MP:200 suggest otherwise. There it is stated that

(33) there seems to a strange man that it is raining outside

converges, though as 'semigibberish'. That would imply that AgrS does not need to check case and/or agreement. Given that possibility it appears unclear again what blocks (32.b). I can see no reason, though, why (33) should be classified as semigibberish rather than non-convergent. If we make that assumption, (32.a) through (33) crash, just as desired, without Greed.

Following Chomsky all those properties are built into the operation Attract: The attracted element must bear the relevant feature and it is the closest one that does so (MP:253f). Furthermore there is a residue of Last Resort, namely that feature checking is necessary from the point of view of the attractor. This, too, is supposed to follow independently, namely from the fact that all features on

functional heads are [-interpretable].[18] Hence their features have to be checked or the derivation will crash, i.e. for them, movement is necessary.

If the feature is strong, attraction has to happen before Spell-Out. By the extension condition this means that a phrase marker cannot be Merged into a larger structure if its root bears a strong feature (it can of course 'project', i.e. adjuncts, complements and specifiers can be added; MP:234f).

The 'attractor' for a moving category XP is not the head itself, say AGR, but the projection of AGR to which XP is going to be attached, in this case AGR'. To accomplish this, Chomsky introduced the notion of *sublabel* (MP:268), which we define slightly more precise in (34).

(34) S is a *sublabel* of K if the maximal X^0 projection of the head of K, $H(K)^{0max}$, contains an element with the feature S (α contains β if it does not exclude it, i.e. if some segment of α dominates β).

Suppose K is AgrS', dominating TP with T moved to AgrS, i.e. $AgrS^{0max}$ = [T AgrS]. Then all (not yet deleted) features of T or AgrS are sublabels of AgrS', in this case [assign nominative], [D-], [V-] and some set of agreement features. AgrS' attracts DP if and only if i) DP can enter into a checking relation with one of these sublabels of AgrS', and ii) DP is the closest element that can enter that relation (see MP:297, def.(84)).

Condition ii) is the only residue of the Minimal Link

---

[18] This is supposed to follow, since only [-interpretable] features are deleted. Suppose a DP moves to SpecAgr to check feature φ, say [1st person sg]. φ on DP is [+interpretable] hence not deleted. If φ on AGR were [+interpretable], too, feature checking would have no effect. Both categories remain unchanged. Featurewise, the output is the same as the input. That, by assumption, renders movement superfluous and hence prohibits it, given Laziness. Hence AGR's feature must be [-interpretable] (MP 282f).

This argument seems rather weak, though. First it is merely stipulated that movements are superfluous if they don't result in feature deletion. It boils down to stipulating that [+interpretable] features cannot attract. Second, we know there to be [-interpretable] features on, say, DPs. Suppose these are attracted by [+interpretable] features on AGR, then movement would still result in feature deletion: on DP. So even if the above reasoning is valid it doesn't follow that the [-interpretable] feature has to be on the functional category.

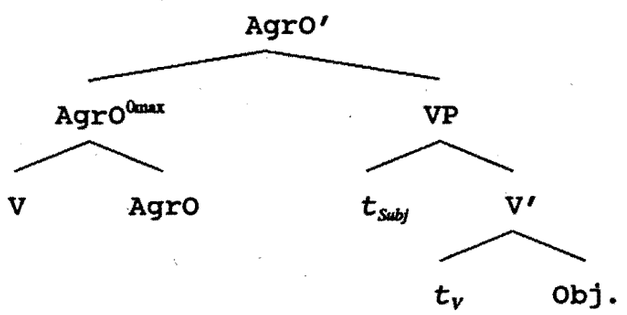
Condition (see section 4; Chomsky keeps referring to it as the MLC, though). But the perspective has drastically changed. We don't move first and then ask whether the movement was minimal. Rather, movement can't happen if it isn't minimal.[19] That is, we calculate in advance whether the result would meet Relativized Minimality or not.

A technical problem arises here, for the target of movement doesn't exist before movement (it is only created with movement). Hence we cannot use the to-be target in our advance calculation but have to calculate closeness from that node that already exists, e.g. AgrS'. Closeness is thus defined as follows (see also MP:299):

(35) β is closer to HP than α, if β c-commands α and there is no X^0-chain whose head is $H(HP)^{0max}$ or a daughter of $H(HP)^{0max}$, such that ß is within the minimal domain of that chain.

This definition is the mirror image of *equidistance*, the notion used before. Its complexity results from the fact that the landing site itself cannot be addressed directly, but only its to-be sister, the 'attractor' in the sense of (34). Suppose HP = AgrO' with V adjoined to AgrO:

(36)



We want to move the complement of V, the object, to SpecAgrO, i.e. attach it to AgrO'. It used to be the case that the minimal domain of the chain (V, t_V) contained both SpecAgrO and SpecV. As a consequence of this, SpecV wasn't

[19] Using the terminology introduced in section 2., we move towards a grammar without stepwise constraints, i.e. without constraints that can incriminate steps within a derivation. Presumably each stepwise constraint can be reformulated in this way, see also FN 8.

closer to the object's base position than SpecAgrO (in the technical sense) and could safely be skipped. Moving the object is possible, just as desired.[20]

Now we determine the set of equidistant positions from the perspective of AgrO' (HP in the sense of (35)) by determining the minimal domain of the V-chain that ends in $AgrO^{0max}$ (Chomsky calls this the 'neighborhood' of HP, MP:299). This domain is the set {SpecV, Obj.} (note again that SpecAgrO isn't there yet). It is irrelevant that Obj., the launching site is in this domain. The only important thing is that SpecV belongs to the domain, for thereby it is not - by definition (35) - closer to AgrO' than Obj. As a matter of fact, SpecV is never closer to AgrO', regardless of the launching site. It is a neighbor and therefore not a potential intervener.

If V hadn't moved to AgrO, the X^0-chain in the sense of definition (35) would be the trivial chain ($AgrO^0$). Its minimal domain contains VP and SpecAgrS, if there, but not SpecV.[21] Accordingly SpecV is closer to AgrO' than anything SpecV c-commands, among them Obj. It blocks movement of Obj. to SpecAgrO (this is Holmberg's generalization: overt object movement is contingent on overt V-movement out of VP). We continue to derive that without head movement no position can be skipped.

Now we can define Attraction:

(37) Attract F:
K *attracts* F if F is the closest category that bears a feature that can enter into a checking relation with some sublabel of K (cf. MP 297).[22]

---

[20] (35) requires that V be the daughter of $AgrO^{0max}$, not just some part of $AgrO^{0max}$. This ensures that with cyclic head movement, e.g. V-to-AgrO-to-T, SpecV is not equidistant to T', even though T^{0max} dominates the head of the chain (v, t_v). This is the counterpart of the earlier requirement that chains are maximally two-membered: domains do not grow with successive cyclic head movement.

[21] This case is not covered by definition (87), MP:299, which is one of the reasons why I've presented the more complicated definition in (35).

[22] In MP:297 only features are attracted, i.e. (37) is replaced by '... if F is the closest feature that...'. The fact that categories move along in overt syntax is supposed to be a reflex of *generalized*

> The way Attract is defined, it is a static relation within the phrase marker. Some formulations within the MP suggest, though, that Chomsky regards Attract as the *process of movement* rather than the configuration that has to hold for movement to take place. For example, following the definition that corresponds to (37) (MP:297) he says: 'The operation forms the chain *(α,t)*'. Accordingly (37) would not be the *definition* of Attract, but a precondition for it. This is also suggested by the formulation 'We define *Attract F* in terms of the condition [(37)]...', MP 297). Accordingly, *Attract* consists of 'attraction' as defined in (37) plus Move. In that case we should replace '...attracts...' in (37) by '...can attract...' so as to allow for features that do not immediately trigger movement (e.g. weak features).

Why is it important, that the MLC and Last Resort are incorporated into Attract ('closest' and 'checking relation' in (37))?[23] The idea is that the minimal step chosen by (37) will not necessarily be wellformed (or rather: will not necessarily lead to a converging derivation). The MLC picks the minimal step, not the minimal one among the promising.

Let us look at a neutral example first and then turn to one where this assumption is crucial (MP:294ff). Suppose a derivation reaches stage (38):

(38) (guess) [Q' they remember [which book Q [John gave *t* to whom]]]

Q and Q' indicate interrogative COMPs, i.e. COMPs that need to check a *wh*-feature (note that (39.a) has a grammatical non-Q reading *...remember that...*, which is irrelevant here). The lower complementizer Q is already satisfied. It

---

pied-piping.

[23] This question has two subquestions: (i) Why is it important that MLC and LR are no longer transderivational constraints? And (ii) Why is it important, that they prohibit certain derivations in advance rather than kicking them out later on at the interfaces (i.e. why are they stepwise instead of representational constraints?)? The answer to question (ii) has to do with Chomsky's idea that stepwise constraints are more desirable than representational ones because they reduce computational complexity: the derivation need not be finished, nor need it be computed for the Competitor Set. See also appendix 6 below. The main text is concerned with question (i) only.

has had its feature checked with the DP *which book*. Q′ still needs to be checked. There are two candidates for this checking, namely the two *wh*-phrases *which book* (whose *wh*-feature is [+interpretable] and hence not deleted after checking with Q) or *to whom*.[24]

(39) a. (guess) [[which book]$_1$ Q′ they remember [t_1′ Q [John gave t_1 to whom]]]

b. (guess) [[to whom]$_2$ Q′ they remember [[which book]$_1$ Q [John gave t_1 t_2]]]

(39.a) meets all preconditions for Attract F. In particular there is no A′-position intervening between *which book* and t_1′ that would count as closer in the sense of (37). All other things being equal, (39.a) will yield a convergent derivation, albeit interpreted as gibberish.[25]

(39.b) on the other hand would not yield gibberish. But that structure cannot be derived using Attract F: in (38) *which book* c-commands *to whom*, it can enter into a checking relation with C′ (as in (39.a)), and it is not in the same minimal domain as Q′ = COMP (or the chain (Infl, t_{INFL}), assuming that I-to-C movement took place), in short: it is closer. As a consequence of that, C′ in (38) cannot

---

[24] According to Chomsky there is a third option, namely *do*-insertion so as to form a yes/no question:
(i) do they remember which book John gave to whom
Chomsky (MP:294) assumes that (38) is an embedded question, hence the *(guess)*. The idea seems to be that (i) is a possible continuation of (38), though one that is bound to crash after the next step, since there is no way to integrate the remaining elements in the numeration (*guess* and functional heads) into the derivation. Crucially both options - *do*-insertion and *wh*-movement - must count as equally economical in the sense of (37) see below. Alternatively one could simply assume that *do* must be part of the numeration. Then (i) wouldn't be a competing option.

[25] Note that (39.a) counts as wellformed (convergent and economical) though not acceptable (grammatical). It is a bit unclear, though, why it should be gibberish. With respect to a similar example (MP:291) Chomsky assumes that a free-standing Q-COMP is interpreted as an embedded yes/no-question, which yields 'presumably gibberish'. (39.a) would then mean something like (i):
(i) * Guess which book they remember whether John gave to whom!
(i) is certainly deviant, but not for semantic reasons: (i) - and thus (39.a), too - should mean the same as (ii):
(ii) Guess for which book they remember whether John gave it to whom!
It is again unclear what defines the class of gibberish expressions and whether its members share any empirical property apart from being unwanted.

attract the PP *to whom* (or its *wh*-feature), moving PP to SpecQ' is thus impossible.

The unacceptability of (39.b) could as well be derived by a transderivational constraint: The derivation (39.a) converges and is thus a competitor, gibberish or not. (39.b) violates the Minimal Link Condition, (39.a) doesn't, so the latter blocks the former. But Chomsky doesn't want it that way. Rather, (39.b) is plainly underivable because of the definition of Attract, regardless of whether or not there is a converging competitor.

This is crucial in example (40) (MP:295), where transderivational and stepwise intraderivational MLC do not yield the same result.

(40) seems $[_{IP}$ that it was told John $[_{CP}$ that IP]]

To complete this derivation we must find a subject for the matrix clause. By assumption the numeration is empty. We can move either *it* or *John* to SpecAgrS.

(41) a. * John$_1$ seems $[_{IP}$ that it was told t_1 $[_{CP}$ that IP]]
b. * It$_1$ seems $[_{IP}$ that t_1 was told John $[_{CP}$ that IP]]

Both options yield unacceptable sentence. We know, why (41.b) is out: *it* has already checked and deleted its case features with the embedded SpecAgrS (case is [-interpretable]). Matrix AgrS can attract the categorial features of *it* ([D-], the EPP effect) as well as its phi-features. But the [assign nominative] feature of matrix AgrS/T remains unchecked and hence undeleted. Since it is [-interpretable], (41.b) crashes at LF.

(41.a) on the other hand would be convergent and grammatical.[26] *John* can check all the features of matrix AgrS/T. The fact that raising out of finite clauses as in (41.a) is nonetheless impossible follows only from the Minimal Link Condition as formulated within Attract: *it* is

---

[26] Note that *John* must bear a [nominative] feature since it is the passivized object of *told*, cf. also the grammatical
(i) it seems that John was told that IP ,
which is based on the same numeration as (41a/b).

closer to matrix AgrS than *John*, it is not equidistant, and it could enter into a checking relation with AgrS's [D-]-feature. Thereby it is the only 'visible' element for Attract. It will be moved, regardless of the fact that the result must lead to a crashing derivation. If the Minimal Link Condition were to transderivationally compare converging derivations only, (41.b) couldn't block anythying. It is thus crucial that the Minimal Link Condition works as a stepwise constraint within Attract.[27]

> We might contemplate the question whether Attract, or rather, the Minimal Link Condition built into it, is a transderivational constraint nevertheless. In a sense it seems to be, because by invoking the notion of 'closest' feature it quantifies over moving competing elements. Yet it is not, because it only considers competing moves, not competing derivations. A constraint will be called transderivational only if it looks 'ahead' into competing derivations up to a well defined point of 'completeness', in our case LF (or the point of cancellation, in case no applications of Merge or Move are licit). Considering only the alternatives to the next step can most likely be reformulated in terms of positions rather than movement steps, as it is done in (37). We therefore reserve the term transderivational constraint for those principles that compare complete derivations in the sense alluded to above. The Minimal Link Condition is thus a stepwise intraderivational constraint.

3.1.1. Attract and Merge

Before we turn to the principle of Procrastinate, a genuine transderivational constraint, let us briefly consider an extension of Attract which Chomsky proposes, namely: The

---

[27] In FN 72 (MP 297/387) Chomsky points out that (41.a) could also be derived using a global, transderivational economy condition. To see how consider again (i), familiar from FN 26.

(i) It seems that John was told that IP.

The idea is that (i) contains a shorter move - raising *John* from the embedded object position into the embedded subject position - than the derivation yielding (41.a), where *John* is raised from the embedded subject position to matrix SpecAgrS. Thus, global economy, too, would block (41.a), this time with (i) as the better competitor. But Chomsky tries to avoid global economy where possible (see the discussion in 6), which is why he prefers the version discussed in the main text. It remains a potentially significant fact, though, that there are no examples in MP that *empirically* favor the stepwise as opposed to the transderivational version of the Minimal Link Condition.

farthest is next! If at a certain point of a derivation the question arises whether to merge or to move, a merged object counts as closer than the moved one. In the cases considered, we have strong features that can in principle be checked by either a moved or a base generated (=merged) element (MP:311).

(42) a. I wonder whether Q he left yet.
b. I wonder if Q he left yet.
c. There is a book on the table.

Whether, if and *there* are base generated in their surface positions and check/delete the strong feature of Q and T/AgrS, respectively. Suppose these elements count as 'close,' then Attract F would pick them as a checker for these features. But as Chomsky immediately notes, there are problems with this extension. He therefore considers restricting it to non-arguments. In the next section we will see that the welcome effects seem to follow from Procrastinate and do not require any modification.[28]

3.2. Global Economy: Procrastinate

We just said that the Minimal Link Condition - as implemented within Attract - is not a transderivational constraint. It only involves a local comparison between positions in the existing phrase marker. We now turn to global economy.

The Procrastinate principle has already been introduced above. It owes its existence mainly to the abandoning of s-structure because it is used to encode s-

---

[28] The pertinent passage is not entirely clear. I said above that Merge is to be preferred over Move. This interpretation, as pursued in the main text, is suggested by the statement that adopting such a condition would wrongly '...force the subject to have accusative case...' (MP:311) (the idea is that V's [assign accusative] feature could then most economically be checked by the subject merged to V'; this derivation could of course never converge, but Attract F doesn't care about convergence).

But maybe the passage is just about enriching Merge by compatibility restrictions similar to those in Attract, e.g. that categories can only be merged if they enter into either a checking or a complementation relation. We will come back to this in section 6 below.

structural word order variations. If a feature is weak, overt movement is unnecessary for convergence, hence blocked by Procrastinate. If a feature is strong, non-movement will be fatal, so movement is permitted, though dispreferredly so. Hence, if a language has overt *wh*-movement, obligatory SpecAgrS (EPP), V-to-I or object shift, those features are strong. If things remain in situ, features are weak.

Not only is this a descriptive rather than an explanatory device, it also hardly requires a transderivational constraint. Suppose e.g. that Attract F in (37) above is simply augmented by '...and that sublabel is a strong feature or Spell-Out has already passed.' That, too, would ensure that only strong features trigger overt movement, but it works purely intraderivationaly1.[29]

In this section we will concentrate on some more subtle effects of Procrastinate, by-products, if you will, which are not readily captured be something like the above reformulation of Attract. Consider the following examples (MP 344ff, ex. 170, 169, 171a/ b, 176, 178):

(43) a. there seems [*t* to be [someone in the room]]
b. * there seems (to me, often) [$_{IP}$ someone to be *t* in the room]

(44) a. I expected [someone to be [*t* in the room]]
b. * I expected [*t* to be [someone in the room]]

(45) a. it seems [that someone was told *t* [that IP]]
b. * it seems [that *t* was told someone [that IP]]

The examples in (a) and (b) in (43) through (45) are based on the same numeration in each case. All three examples present a similar problem: We need to fill three A-positions but we've only got two DPs to fill them. Hence we

---

[29] One might object that a condition like 'Spell-Out has passed' is ugly, non-minimalist in spirit and reactionary in that it sneaks in the notion of s-structure through the backdoor. True enough, but exactly the same holds for Procrastinate as it stands. As long as languages show word order variation, the point of Spell-Out is significant in describing them and will enter the descriptive apparatus at some point; period.

must apply A-movement in at least one case so as to have one DP check two positions. What determines where to merge and where to move?

Let us focus on that stage of the derivation where such a decision must be made. For (43) this is (46):

(46) AgrS to be someone in the room

AgrS has a strong [D-] feature (EPP) that could be checked either by movement of *someone* or merger of *there*. As (43) shows we have to go for the latter option: *there* must be merged and subsequently raised to matrix AgrS (since categorial features are [+interpretable] and can hence enter into multiple checking). But why not raise *someone*? Chomsky's answer is: Because that would violate Procrastinate. To merge means to not move, and to not move is good.

(44), too, goes through stage (46). Here the choice is to merge the pronoun *I* or to raise *someone*. We know that Procrastinate favors the former option. That would yield (44.b), the unacceptable result. So there must be a superordinate reason why *I* cannot be merged here, despite being procrastinating. This reason lies in θ-theory. *I* in (44.b) cannot receive a θ-role: *seem* doesn't have one, *expect* does, but it cannot assign it to a chain (i.e. no raising into θ-positions). Hence (44.b) crashes. It is not part of the Competitor Set and can thus not block the non-procrastinating (44.a).

Note that this explanation wouldn't work if (44.b) were just convergent gibberish, because gibberish participates in economy competitions. Nor would the explanation work if Procrastinate, just like the Minimal Link Condition, wouldn't care for convergence, still preferring (44.b). (44.a) is chosen only because it is the best way to construct a derivation that finally converges.

Something very similar happens in (45). The question here is whether - at stage (47) below - to raise *someone* or to merge *it*.

(47) AGR was told someone that IP

Again Procrastinate votes for merging *it*, thereby avoiding movement. But the consequences would be disastrous: *it* would check the [assign nominative] feature of the embedded AGR and subsequently be raised to check the [D-] feature of matrix AgrS. It would not, however, check the [assign nominative] feature of matrix AgrS, because it doesn't have a [nominative] feature anymore. After only a little while, *someone* would try to check its [nominative] feature, but in vain, since the embedded AgrS has already checked and deleted its [assign nominative]. The derivations thus crashes. Since crashing derivations don't compete, the road is clear for the apparently uneconomical (45.a): Starting with (47), *someone* is raised to embedded AGR, and *it* is merged into the matrix. Just as in (44) Procrastinate become relevant only if the procrastinating derivation converges in the first place.[30]

We are thus dealing with a truly global economy principle: Avoid movement as far as this is possible guaranteeing convergence. Had Procrastinate been an intraderivational principle like the MLC, blind to the future of the derivation, the outcome would be drastically different: (44.b) and (45.b) would be grammatical, (44.a) and (45.a) ungrammatical. It's the data that tell us that the MLC is intraderivational (ex. (40)/(41)) and that Procrastinate is transderivational (ex. (43) through (45)). Roughly: If a principle can be violated for the sake of convergence, it must be transderivational.

---

[30] There might be a converging derivation that results in (45.b), though. After (47) *it* is merged into SpecAgr and checks AGR's [assign nominative] feature. Later, as we know, matrix AgrS calls its demand for a [D-] feature:

(i) AGR seems that it was told someone that IP

By the Minimal Link Condition *it* must be the element that is raised, yielding (ii):

(ii) It seems that *t* was told someone that IP

Spell-Out applies and (45.b) result. After Spell-Out we have to check the [assign nominative] feature of matrix AgrS and the [nominative] feature of *someone*. So matrix AgrS attracts. *It* has checked an deleted its case feature, moreover it is in AgrS's minimal domain. Accordingly, *someone* - or its formal features - is the closest case feature around and should be attracted by Agr. It is raised covertly, both features are checked and deleted, and the derivation converges.

Three more remarks. First notice that only (43) is really about Procrastinate. (44) and (45) are simply cases where we have to explain why Procrastinate does *not* have effects we might expect it to have. If there were no Procrastinate we'd still predict the (b) examples to be out and the (a) example to be fine. The interesting case is just (43) where an apparently perfect derivation - (43.b) - is ungrammatical only because economy has found a yet better competitor. Generally, an argument in favor of economy must have this structure: S should be okay, given everything we assume. But there is some competing and convergent S′ that actually meets economy condition E better than S, which therefore blocks S. The fact that S is out, though convergent, provides an argument in favor of E.

Second, it does not seem to be generally true that raising in a is blocked whenever there is the possibility of merging. Consider the examples in (48.a), pointed out by Chris Wilder (p.c.).

(48) a. A rumor was in the air that there was a man in the room.
b. There was a rumor in the air that a man was in the room.

(48.a) is unproblematic: When the derivation reaches the point illustrated in (49), *there* is inserted, respecting Procrastinate.

(49) was a man in the room

However, at the very stage (49), *a man* must be raised in order to derive the equally grammatical (48.b). But this raising should be blocked as a Procrastinate violation by the option of inserting *there*, as before. Since (48.a) is convergent, it should block (48.b). The latter is predicted to be ungrammatical, contrary to fact. It thus seems that the general upshot of Chomsky's analysis of (43) - raising is postponed until all expletives are used up - is empirically wrong.

Third, there is a potential problem with the interaction of global and local economy conditions. In

subsection 3.1.1. we discussed the idea that the Merge vs. Move preference is part of the definition of Attract (MP:311f). Attract being an intraderivational constraint it does not care for convergence, as discussed at length. Then in all those cases discussed in this section, Merger should be applied where locally possible, irrespective of later (non-)convergence. That is, *it*, *I* and *there*, respectively, must be merged into the embedded SpecAgrS position and both (44.a) and (45.a) (just as the respective (b)-examples) should be out (note that Procrastinate would be irrelevant because it only considers those derivations that can be generated by Attract to begin with).

It therefore seems desirable to abandon the Merge-over-Move preference as built into Attract and leave the pertinent cases (base generated *wh*-elements and expletives, see (42) above) to be explained by global Procrastinate alone.[31]

3.2.1. The Local Implementation of Procrastinate

According to Chomsky, Procrastinate doesn't exactly work the way we described it above. We don't compute every derivation that can be computed from a given numeration and than pick those that are convergent and involve the smallest number of Procrastinate violations (this I called the *global implementation* of Procrastinate above). Rather we ask with every step of the derivation: Which of the steps that are possible next at this point, and that can ultimately lead to a convergent derivation, is the most economical one? That is, the Competitor Set is only calculated when the question 'to move or not to move' arises. In our above examples, Procrastinate is invoked no earlier than at stage (46) of the derivation, repeated here.

(50) AgrS to be someone in the room

AgrS has a strong [D-] feature, which attracts (or could

---

[31] All the more so since Procrastinate in any event cannot be reduced to Attract entirely because the latter doesn't see the 'weak'/'strong' difference. See the remarks at the beginning of this subsection, though.

attract) *someone*. So we wonder - as before - whether to move *someone* or to merge *there*. Starting from structure (50) (plus the remainders of the numeration) all possible continuations are derived. If among them there is one that converges and does *not* involve movement of *someone* to AgrS, movement is blocked by Procrastinate. This we call the *local implementation*.

According to the terminology of section 2, the local implementation of Procrastinate is stepwise (the principle is invoked with every movement operation), but still transderivational (it compares complete convergent derivations). It is local only in that it takes a certain derivational stage rather than the initial numeration as its starting point. But from this point on it computes all possible alternatives, excludes the non-converging ones and then checks whether the movement at hand could be procrastinated until after Spell-Out, which is a genuinely transderivational mode of operation. Some problematic aspects of this have already been dealt with in subsection 3.2. above. We will return to some of the conceptual issues in the appendix 6 below.

At this point we want to focus on the fact that the local implementation of Procrastinate is *not* empirically equivalent to the global one. According to the latter the total number of overt movements determines (un)economy: The converging derivation(s) with the least overt steps, zero at best, win(s). According to the local implementation the following situation can arise: At an early stage S of a derivation some movement M is at stake. With M taking place, the derivation could ultimately converge, using a total number of *n*, say five, overt movements. Call this total derivation D. There exists, however, a competing derivation D′ that includes S but does not include movement M, at least not overtly. D′, too, converges, but it involves *n+x* overt movements, say ten. According to the local implementation of Procrastinate the latter option must be chosen, blocking the former. It is irrelevant that D is the more economical derivation from a global point of view, because D′ locally avoids movement. At the time D′ faces its sixth Procrastinate violation, D is no longer a competitor to it.

In general it holds that the global implementation picks the derivation with the smallest overall number of overt movements, while the local implementation picks the derivation were the first overt movement is delayed as long as possible, which means de facto: it prefers a Procrastinate violation higher in the clause to one lower in the clause.

Although Chomsky doesn't explicitly mention it, (43) above seems to be an illustrative example. The question there was what to do after (46), repeated here.

(51) AGR to be someone in the room

We could merge *there* or move *someone*, resulting in (52) and (53), respectively.

(52) a. there to be someone in the room
b. AGR seems there to be someone in the room
c. there seems *t* to be someone in the room

(53) a. someone to be *t* in the room
b. AGR seems someone to be *t* in the room
c. * there seems someone to be *t* in the room

As before the second option must be excluded for it yields an unacceptable outcome. This was done by saying that step (52.a) blocks step (53.a) since overt movement (of *someone*) is dispreferred by Procrastinate. If we look at the entire derivation we find that (52) does not contain less overt movements than (53). Both involve one (relevant) overt movement (raising of *someone* and *there*, respectively) and one covert movement (raising of the case- and agreement features of *someone* to matrix AgrS). The difference is that overt movement in (52) occurs later (hence higher in the tree) than in (53). At stage (51) the local implementation of Procrastinate prefers the continuation in (52.a), ignoring the virtual number of Procrastinate violations in the complete derivation. Global Procrastinate would presumably judge (52) and (53) as equally economical and hence grammatical.

Chomsky doesn't mention this empirical reason for adopting the local implementation, though. Rather, his concern is the *reduction of computational complexity*. The idea is that it is less complex to compute all possible continuations to a given derivational stage and compare them wrt. the next step than to compute all derivations that are possible from a given numeration and compare them wrt. the total number of violations they involve. We will return to this point below.

Just a brief remark here: The local implementation of Procrastinate is more complicated than one might first think. We assume that after each overt movement Procrastinate checks whether that movement was unavoidable (alternatively we can assume that Procrastinate checks *before* the movement whether the movement would be unavoidable). To do so it calculates the Competitor Set, which simply means: It tries out all applications of Attract/Move and Merge possible given the phrase marker(s) already formed and the numeration. The operation Attract/Move, as used in the calculation of the Competitor Set, is restricted as usual, e.g. in the form (37), incorporating the MLC and Last Resort. Move cannot, however be restricted by Procrastinate, otherwise calculating the Competitor Set would lead to an infinite regress (because in calculating the Competitor Set, Procrastinate has to calculate the Competitor Set, which requires calculating the Competitor Set...).

In other words, while in doing the actual derivation, Procrastinate is closely interwoven with Move and Merge, it is irrelevant in computing the Competitor Set. As a consequence of that, the Competitor Set is no longer definable as a set of convergent derivations: convergent derivations under the local implementation respect Procrastinate. Derivations in the Competitor Set respect all interface conditions plus the conditions built into Attract (MLC, LR), but not Procrastinate. Put in other words, locally implementing economy principles obviates the need to distinguish being convergent from being economical (since convergence presupposes fulfillment of local economy principles). But in return for that two different concepts of convergence are required: Convergent as being

generatable using Attract/Move and Merge and meeting the interface conditions (that is the 'old' convergence, i.e. the prerequisite for being a competitor), and convergence as being generatable using Attract/Move, Merge, *respecting Procrastinate*, and meeting interface conditions (this corresponds to the notion of economical in section 2 above). We will show how this can be accomplished formally in the appendix 6 below.

3.3. Residual Issues

3.3.1. Successive Cyclic Movement

Interpreting the MLC and LR as part of Attract/Move has crucial consequences wrt. successive cyclic movement: it ceases to exist. Or, to be more precise: It only takes place if the intermediate landings are forced by feature checking independently (as with complementizer agreement in Irish, participial agreement in French, and of course successive A-movement, which is triggered by EPP features in different clauses, see the discussion in MP:299ff). We can of course force feature driven intermediate landings in languages as English as well, by postulating morphologically invisible agreement, [D], or [W] features in, say, SpecC. This option is suggested, though not pursued, by Chomsky: 'it remains an open question, whether such visibility [i.e. intermediate complementizer agreement as in Irish, DB] is only an accident of morphology, revealing the workings of a process that is more general, perhaps universal, even if morphological reflexes are not detected...' (MP 267).

In general, however, long extractions do not require intermediate traces, and - minimally thinking - should not have them. Consider e.g. *wh*-islands. In LGB they were thought to result from the interaction of subjacency or the ECP and phrase structure. By subjacency (and possibly the ECP) a *wh*-phrases ß had to touch down between IP and CP if it wanted to leave the latter. For reasons of phrase structure, SpecC was the only position to serve this purpose. If it was occupied by some other element α, the escape route was blocked, the intermediate landing was impossible, subjacency was violated and the structure thus

marked (similar for topic islands and relative clauses). Within the MP it is α itself that blocks leaving an island (by the MLC). If there is no α to begin with, extraction can proceed in one fell swoop. If there is an α, ß will not even be attracted (unless α is in the 'neighborhood' of the attractor, a case not considered for A'-movement), so the issue of intermediate landings doesn't arise.

Prospects are worse still for intermediate traces in adjoined positions: 'successive-cyclic adjunction is even more problematic. The condition [Last Resort as defined in Attract, DB] could be restricted to A-movement, or perhaps modified to include satisfaction of the MLC alongside feature checking, though consequences are rather complex' (MP:267). Since the MLC is to become part of Attract itself only shortly after, only the former, rather non-minimalist solution would remain.

3.3.2. Mismatch

So far we have said that *features* are checked, where features included case, agreement, category and so on. But more precisely we should say that *feature values* are checked, namely checked for compatibility.

(54)	**Feature**	**Value**
	Case	nominative (accusative, null...)
	Agreement	3.Ps.Sg.
	Category	N, V, D

The question now is: Does a node attract a *feature* or a certain *feature value*? Chomsky adopts the latter option. Thus, if T has the case feature [nominative], it cannot attract an DP that bears the case feature [accusative].[32] Accordingly, the MLC only regards 'attractees' with the

---

[32] We should also be precise and say that T has the feature [assign nominative]. Quite in general, features on attracting categories are not *identical* to those on attracted categories, but their mirror image. T doesn't carry nominative, no more than strong AgrO has the category D (we should rather say it has the feature [get D]). The only exemption are agreement features which seem to be of the same type on all categories involved. Needless to say that the distinction between attracting features and attracted features complicates the notions of attraction and (mis)match, though not in an essential way. We will ignore these matters here.

same feature *values*.

Knowing this, let us reconsider (40), repeated here.

(55) seems [$_{IP}$ that it was told John [$_{CP}$ that IP]]

Above we assumed that *it*, being closer to matrix AgrS, blocks attraction of the farther but appropriate element *John*. But note that *it* doesn't have a [nominative] feature, because this feature has been checked and deleted with the embedded AgrS. It must thus be the case that AGR attracts the [D] feature of *it*, which in fact is closer to it than that of *John* (MP:310).

In general it holds that only matching feature values can be attracted. Once an element α has been attracted by and moved to F, however, all features (not feature values!) shared by α and F are checked. Thus if [T AGR] attracts some DP (by the [D] feature) and this DP bears [accusative], we get a *mismatch*: (MP:309) DP's case feature, though not attracted for its own sake, is checked against T's case features, once it has entered the checking relation (Chomsky calls such features which are moved though not attracted *free riders*). And since [accusative] and [assign nominative] do not match, the derivation is *canceled* (MP:309).

Surprisingly, canceled derivations are ungrammatical but convergent (or at least can be). Accordingly they can block alternative convergent derivations, if these are less economical (MP:309).[33] Following the terminology of section 2, *match* is an intraderivational constraint that influences grammaticality, i.e. it belongs into class (v) of the model in (20) above. At the same time, *match* is a stepwise constraint, i.e. one that checks every application of Attract/Move. How can we implement such a twitter principle? Either we invent a 'flag' [±mismatch] which is added to each derivation. Mismatch during a derivation sets the value of this flag to [+], and after checking economy conditions all [+mismatch] derivations are filtered out as

---

[33] *Cancellation* therefore is a misleading term, since on the contrary the derivation is continued. It must be continued in order to see if it finally converges, hence if it is going to be in the Competitor Set.

ungrammatical. Alternatively we implement *match* via universal quantification over derivational stages (or universal quantification over checking configurations, using the copy theory of movement), in the way demonstrated in (21) above. In any event *match* seems to be rather inelegant in its actual implementation, violating an otherwise desirable principle such as (22).

> The example that leads Chomsky to this particular version doesn't seem to be relevant, though, namely a derivation in which SpecV is occupied by an [accusative] DP α while the object position is occupied by a [nominative] DP η (MP:309f). Assuming the usual derivation that involves crossing paths, α winds up in SpecAgrS while η winds up in SpecAgrOP, i.e. double mismatch. If mismatch precluded convergence, we could, according to Chomsky, construe an alternative derivation with nested paths, which would yield matching case features (η in SpecAgrS, α in SpecAgrO, i.e. *she kissed him* meaning 'he kissed her'). To block this option, the mismatching but economical version must block the matching uneconomical one. Therefore mismatching derivations must converge (although it is not quite clear in what sense the mismatching derivation would be more economical).
>
> However, nested paths as in the matching derivation are precluded by the MLC, and the MLC is not a transderivational, 'soft', constraint, but an intraderivational, 'hard' one. In particular we know that the MLC does not care for convergence. It thus blocks the unwanted nested paths derivation, regardless of whether there is a more economical competitor or not.
>
> So it seems that *match* could as well be implemented as a garden variety stepwise constraint that leads to non-convergence.

To define the notion of mismatch, Chomsky supplements the notion of *checking configuration* (=a configuration in which some feature is within the checking domain of the checker) with that of *checking relation*:

(56) Feature F′ is in a *checking relation* with *f* iff F′ is in checking configuration with *f*, and f and F′ match in feature values (cf. MP:310)

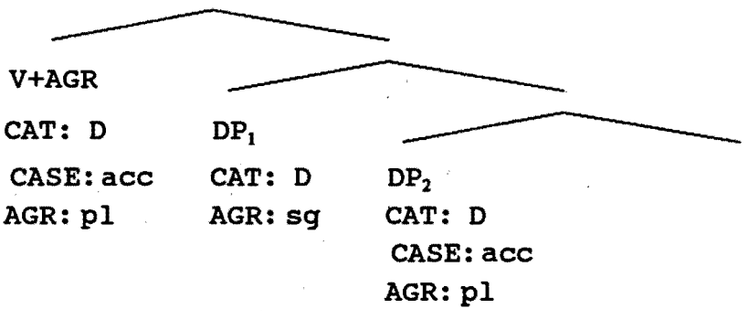
The following thus holds: Let F be a sublabel of M, then F

attracts α, if α is the closest feature whose value matches that of F. All other features of (the element bearing) α are carried along by movement, entering into a checking configuration with all sublabels of M. This rather blind application of Attract can lead to ungrammatical sentences in at least two cases: i) if not all [-interpretable] features can subsequently be deleted (the derivation crashes, e.g. (41.b) above); or ii) if α carries along some features that do not match the corresponding sublabels of M (the derivation is cancelled, though possibly convergent).

Consider now the case where one single element has to check several features, e.g. [V AgrO] with CASE:[assign acc], CAT:[D] and AGR:[...]. If any of these feature is strong, that feature will get to attract first, possibly dragging in other features as free riders. If these match with [V AgrO]'s, checking will take place immediately.

Assume instead that all features of [V AgrO] are weak. Can *any* feature of [V AgrO] attract the closest element matching it (after Spell-Out)? Or can only that element be attracted which is the closest that matches *some* feature of [V Agr]? To give an example, take the following configuration:

(57)



Obviously, DP_1 is the closest category that can be attracted by some feature of [V AGR], namely the D-feature. And just as obviously this would result in mismatch wrt. the AGR feature. On the other hand, one might argue, DP_2 is the closest element that can be attracted by V+AGR's CASE and AGR feature (suppose, for the sake of the argument, that DP_1 has had its case feature checked before). So V+AGR - or its sublabels - could attract DP_2, checking the D-

feature as a free rider and thereby avoiding the mismatch. Is this latter option available?

According to MP:310 (bottom) it is not. For an element to be attracted it has to be the closest among those that could check *any* feature of the target. This is also expressed in the above formulation of Attract ((37): '...*some* sublabel of K').

It remains an open question, though, whether other, more remote features can subsequently be attracted. Consider again (55), repeated here.

(58) AGRS seems $[_{IP}$ that it was told John $[_{CP}$ that IP]]

AgrS's only strong feature is [D], hence *it* is overtly attracted. *It*, however, has already checked and deleted its [nominative] feature with the embedded AgrS. So can't the [assign nominative] feature of matrix AgrS covertly attract the DP *John*? *John* is the next element bearing a [nominative] feature, in fact the only element that can check *any* of AgrS's features. As said before we want the answer to be negative, since *it seems that it was told John that...* is unacceptable, but it is unclear why. The only candidate for blocking LF attraction of *John* would be the trace/copy of *it*. But that trace has only one feature left, [D], and [D] has already been checked and deleted at matrix AgrS. So AgrS can*not* attract the trace of *it*, hence cannot block attraction of *John*.[34] It thus seems that (58), or rather **it seems that was told John that...* is not properly excluded by the MLC and *match*.

3.4. Conclusion

---

[34] This is extremely important, for otherwise associate raising with expletives would always be blocked, cf. (i) (see the discussion in MP:301f, esp. ex.(91)).

(i) there seems *t* to be some book on the table

For matrix AgrS to be able to attract the case and agreement features of the associate *some book*, the trace of *there* must not count as an intervener. This is ensured if the only feature of AgrS that attracts *there*, [D], is checked and deleted after raising *there*. Since *it* in (58) has checked and deleted its case feature in the embedded clause, (58) equals (i) in all relevant aspects.

As it looks, there is only one transderivational principle left at the end of C&T, Procrastinate. And even Procrastinate is no longer implemented in the 'classic' way (i.e. as a principle that globally compares converging derivations), but locally, constraining the process of generation itself.

Most constraints have been incorporated into the definition of Attract F, first and foremost Last Resort (movement presupposes attraction by some [-interpretable] feature) and the Minimal Link Condition (only closest features can be attracted). It also remains true that movement is motivated strictly locally: *It is not convergence that the movers desire, but merely feature checking*. This point is worth emphasizing: Last Resort is not a transderivational constraint, but intraderivational (and stepwise, evidently so because it is part of Attract).

Gone is *Greed*: It is irrelevant whether the moved category has to get rid of some feature or not, in particular (i) [+interpretable] features can be attracted, and (ii) one feature on, say, DP can redeem several heads from their features (cf. e.g. the discussion of *wh*-islands around examples (38)ff above).

As said above, locality (or Relativized Minimality) is built into Attract, too, crucially using the notion of equidistance which remains de facto the same but is formulated the other way around.

Attract F as formulated also resolves an ambiguity around Last Resort that arose earlier: Is it sufficient if movement of α will *ultimately* result in feature checking, or must feature checking take place *with this very step* (cf. e.g. the discussion in MP:257ff where three interpretations of Last Resort, there ex. (20), are discussed)? The answer now must be *with this very step*! For note that movement is solely triggered by the target position, i.e. the requirement to check a feature. There can be no movement without immediate feature checking.

Within the MP the only restriction on the *locality* of movement discussed is the MLC. Hence, as noted above, intermediate traces, and hence movement without feature checking, are ruled out in principle, because they are

useless. If the MLC is in fact the only locality condition within core grammar (as minimalist assumptions would no doubt lead us to expect), intermediate traces are obsolete, and so would be any condition on the length of chain links such as 'Minimize Chain Link' or 'Shortest Move' which seem to have ceased to exist.[35]

The principle Fewest Steps, too, seems to have disappeared without a trace. On a large scale it is subsumed by the definition of Attract: Only [-interpretable] features can trigger movement, hence there are no superfluous moves (e.g. moves for convergence, or moves for checking [+interpretable] features in the moved element, or movement to create intermediate traces). It seems that the total number of movements is largely determined by the number of strong and [-interpretable] features in the initial numeration plus the 'minimal' clause of the MLC which leaves little choice as to which element to attract. Although not all cases are easy to conceive of it seems that Fewest Steps would in fact be redundant.

> Fewest Steps reappears, however, in MP:357, where the following scenario is discussed: There are two DPs, and three features on functional heads that need to be checked, subject case, object case, and the EPP feature, the latter two being strong. The question is: Why will the subject have to move overtly in this situation, i.e. why can't the object check object case and subsequently the EPP in SpecT (with the subject raising to T covertly)? And the answer is: Because such a derivation would involve three movements (object to AgrO/outer Specv; object to SpecT; subject to T), and there is a shorter derivation, in which the subject moves overtly to SpecT to check subject case and EPP, while the object moves only once to check its case.
>
> Fewest Steps as envisaged here is again locally implemented: 'after raising the object, we chose the operation that will lead to the shortest convergent

[35] As mentioned above, intermediate landings can be implemented by successive checking of intermediate features. But these are not intermediate landings in the technical sense, but feature driven movements.

derivation' (MP:357).[36] Note also that Fewest Steps has to resolve a tie wrt. Procrastinate here. Since subject and object are equidistant from the checking position for object case, both can be raised. In that sense, movement of neither is unavoidable, though *some* movement needs to be done. The situation is in fact similar to the mutual blocking of I-to-C and XP-to-SpecC movement discussed in section 2 (ex.(30)f) but crucially more dilemmatic, since the two competing derivations or movements are in no sense 'the same'. The exact interplay between Procrastinate and Fewest Steps would thus have to be extremely complex, if implementable at all.

As J. Sabel (p.c.) points out, however, Fewest Steps might be obsolete in this case as well, allowing us to eliminate it completely (along with the problems it would raise, one is inclined to add). This presupposes that we adopt an idea proposed (though only tentatively assumed) shortly afterwards, namely that the object's case position is *below* the subject's base position and equidistance can altogether be dispensed with, i.e. movement can *in no case* cross an element of the same type.[37] In this case, raising the object to SpecT to check the EPP is not an option since it would have to illicitly cross the subject.

In sum, it seems not entirely clear whether Fewest Step is assumed to be operative at the end of C&T, but it seems likely that it is superfluous.

Summing up, the final version of the MP economy system can be depicted as in (59), if Procrastinate is implemented globally...

---

[36] It is less clear that the local and the global implementation of Fewest Steps differ in empirical consequences since both concern the overall number of steps. As Joachim Sabel (p.c.) pointed out one can conceive of a different formulation, to the effect that Fewest Steps locally prohibits movement of α if movement of α can altogether be avoided in some competing derivation. This would be in full parallelism to the local implementation of Procrastinate. In fact, the empirical effect of such a formulation seems to be the same as that of 'traditional' Fewest Steps and local Procrastinate, since leaving something in situ altogether is just a special case of procrastinating movement (in the technical sense).

[37] As announced in FN 16 at the very beginning of this section we have ignored this option - which presupposes abandoning agreement categories - so far (and will continue to do so). This is not because I find the idea itself revolting but merely because the resulting system is not even roughly characterized in C&T nor is it appointed an 'official' part of the MP.

(59)

(i) numeration

|

(ii) generation

Move

Merge

(respecting:

- the constraints built into Attract/Move (MLC, LR)
- possibly other intraderivational constraints (extension requirement?)

|

(iii) interface conditions

(=intraderivational constraints defining *convergence*)

Full Interpretation (LF & PF)

θ-marking and possibly -discharging

|

(iv) economy conditions

(=transderivational constraints defining *economy*)

Procrastinate

((Fewest Steps))

|

(v) other constraints

(=intraderivational constraints defining *grammaticality*)

Gibberish

Mismatch

(ECP, according to Collins 1994)

...or as in (60), if it is implemented locally.

(60)

(i) numeration

|

(ii) generation

Move

Merge

(respecting:

- the constraints built into Attract/Move (MLC, LR)
- possibly other intraderivational constraints (extension requirement?)
- stepwise but transderivational constraints:

Procrastinate (& Fewest Steps?)

| \

| generation

| +interface conditions

|

(iii) interface conditions

(=intraderivational constraints defining *convergence*)

Full Interpretation (LF & PF)

θ-marking and possibly -discharging

|

(iv) other constraints

(=intraderivational constraints defining *grammaticality*)

Gibberish

Mismatch

(ECP, according to Collins 1994)

The distinction between (iii) and (iv) can in principle be given up. Note, however, that the requirements grouped in (iii) form a significant class, namely the class of constraints that are also operative in generating the Competitor Set, whereas those in (iv) are not. We will return to these intricacies in 6 below.

This concludes the presentation of the final MP economy system. In the next section we will discuss some principles that have become obsolete. This is done in order to facilitate an understanding of the earlier discussions in MP chapters 1 through 3 as well as of earlier or non-

orthodox work done by others within the minimalist setting. We also want to keep an eye on whether the cases discussed in connection with these principles earlier within the MP are still covered by the leaner final version.

4. Obsolete Principles

4.1. Greed / Last Resort

As said above, within the MP movement only takes place to check features. Let us now come back to the somewhat sophistic question mentioned at the end of section 1 above: Does, say, DP move in order to check and delete *its* features, or to check and delete those of AGR? The case is undecidable if both elements have to get rid of their features, since movement would equally serve both purposes.

The crucial case are thus features that need not be checked, i.e. [+interpretable] features, features that are not deleted. These include the categorial features (N and V) and, as assumed in C&T, features like person and number of nouns. Assume a DP has only features of this type left, will it nevertheless move if AGR calls its demands? To use a common term: Can there be *altruistic* movement?

In MPLT the answer to this question is definitely negative: α moves if α needs to check *its* features. That is called the principle of *Greed*. Greed is basically a radicalized version of Last Resort. LR requires that movement is legitimate only if inevitable to save convergence ('...a step in a derivation is legitimate only if it is necessary for convergence...,' MP:200). It thus leaves room for altruistic movement.

Greed is 'self-serving last resort' (MP:201: 'raising of α...is blocked by the fact that *its own requirements* are satisfied without raising...', italics there). It thus subsumes LR. Greed can block an operation - even if only that very operation would eventually save convergence - provided this operation is not driven by local, selfish reasons ('the self-serving property of Last Resort cannot

be overridden even to ensure convergence,' MP:201).[38]

As said before, all features other than categorial features are assumed to require checking and deletion within MPLT. As a consequence, DPs almost always move, even given Greed. Suppose, for example, that AgrS has a strong N feature (EPP). In that case, no DP would move to AgrS in order to check that feature. The strong feature of AGR could thus only be checked if there is another feature on AGR, say, case that DP itself needs to check.

If - as assumed in C&T - case is the only [-interpretable] feature of DPs, Greed would block all movements that do not take place in order to check case. All other feature checking - e.g. AGR eliminating its strong and even weak features - would have to occur as a by-product, contingent on case driven movement. These features couldn't ever trigger movement by themselves. Presumably this is a far too severe restriction, but as said before, Greed vanishes with the introduction of [+interpretable] features, and public interest wins out.

4.2. Greed, Procrastinate and Enlightened Self-Interest

At the times of MPLT it was sometimes argued that Greed and Procrastinate threaten to conspire so as to block all overt movement (Lasnik 1993, Cavar & Wilder 1994, Kim, Kolb, Müller & Sternefeld 1995). The argument goes like this: Suppose you are a subject DP needing to check your case and agreement features. In order to do so you'll have to eventually move to SpecAgrS. The [T AGR] head has a strong D-feature. If no DP shows up in SpecAgrS by Spell-Out the derivation is bound to crash. But you, the subject DP, would only move out of self-interest (Greed), and by Procrastinate you would do so as late as possible, in any event not before Spell-Out. Due to your being greedy the derivation is inevitably going to crash.

This, it should be noted, is the standard situation since - by assumption - only functional categories have

---

[38] Chomsky seems to use *Greed* and *Last Resort* basically as synonyms. *Greed* is defined as 'self-serving Last Resort,' but about *Last Resort* it is said that 'Last Resort, then, is always "self-serving"' (both quotes from MPLT, MP:201).

strong features. The moving element usually (or generally) doesn't. Over all, then, overt movement will never take place, unless the *moved* category bears a strong feature that can be checked that way. As this state of affairs is obviously undesired, it has been proposed to relax Greed, e.g. as *enlightened self-interest* (cf. e.g. Kim, Kolb, Müller & Sternefeld:61):

(61) *Enlightened Self-Interest*: An element α can only be moved if
   a. some feature of α cannot otherwise be checked/deleted, or
   b. the derivation would crash otherwise

Clause (61.b) is the enlightened part about this condition. To use Kim, Kolb, Müller & Sternefeld's apt words: 'What good are checked features if they cannot be enjoyed in convergence...?'

This, however, is obviously not what Chomsky intends. Consider his discussion of (62): Why can't this derivation converge?

(62) seems to [$_{\alpha}$ a strange man] [that it is raining outside]

In (62) matrix T has a strong D-feature. The DP *a strange man* could move and check this feature. That wouldn't be greedy, of course, (after all, DP doesn't have to check its D-feature) but it would be enlightened, because only then would the derivation converge. According to (61.b) movement could thus take place and we'd - wrongly - predict (63) to be grammatical.

(63) * A strange man seems to *t* that it is raining outside.

But '[t]he self-serving property of Last Resort cannot be overridden even to ensure convergence' (MP:201). (62) has

to crash, (63) is predicted to be ungrammatical.[39]

Although there are certainly ways to maintain Enlightened Self Interest as defined in (61) and nevertheless block (63) (see e.g. FN 39), such a modification seems unnecessary, basically because the alleged shortcoming of the principle of (shortsighted) Greed, when combined with Procrastinate, are only apparent.

To see this, recall that Greed is an intraderivational constraint: Move α cannot take place if it doesn't result in a feature of α being checked. Procrastinate, as opposed to that, is a transderivational constraint: Out of two (or more) competing and converging derivations only the one with the fewest number of overt steps survives.[40] As a consequence of that, a DP, say a subject, can move only if it has a feature that needs to be deleted. If there is no such feature, movement is impossible, regardless of convergence. That is why (63) is out.

However, *at which point of the derivation* (pre or post Spell-Out) the subject moves is arbitrary during the derivation, i.e. both options are generatable. In case both derivations - one with overt subject raising the other with covert subject raising - converge, Procrastinate gets to choose between the two of them, picking the latter. That's what precludes e.g. overt V-to-AGR and overt object raising in English.

But if only the derivation involving overt movement converges, Procrastinate is irrelevant. This is the case with overt subject raising: If the DP moves to AgrS only after Spell-Out the derivation crashes (due to the strong feature in T). Accordingly, this derivation is not in the

---

[39] One could argue that it is T's case feature that is eventually going to provoke the crash, regardless of Greed: Notice that *a strange man* is not likely to be able to check this feature, either because its case feature is already deleted or because of case mismatch (DP's case is either objective or inherent). As before we ignore such considerations, trying to extract the spirit of Chomsky's argument.

[40] This formulation leaves open the question whether Procrastinate and Fewest Steps are ranked. Are more steps better, if they are all covert? Or does Procrastinate only operate in equally long derivations (cf. FN 12 above and the excursus on page 51)? Since empirical data are hard to bring to bear on this issue and the question doesn't seem to arise in the final version we will not dwell on this point.

Competitor Set to the one with overt subject raising (in fact it is in no Competitor Set whatsoever), hence questions of procrastinating do not even emerge.

This illustrates that it is important to keep in mind the differences between and the interactions of intra- and transderivational constraints. The problem that motivated Enlightened Self Interest seems to be nonexistent (that Procrastinate is in fact such a transderivational constraint - with the properties relevant here - can again be told e.g. from the discussion summarized in subsection 3.2. of this paper).

4.3. Fewest Steps

We have seen before that Fewest Steps is (presumably) redundant. It is largely replaced by the form of Last Resort as incorporated in Attract: Only unchecked [-interpretable] features can trigger movement. Put differently, the number of movements is determined by the number of [-interpretable] features within the numeration.

Let us briefly reconsider some of the cases that have been handled using Fewest Steps in the literature. Collins (1994) discusses what he calls *Chain Interleaving* in examples like (64).

(64) * Who was a friend of *t* arrested?

There are at least three ways of generating this string: (i) subject to SpecAgrS, followed by *who* to SpecC. This derivation is blocked because the subject is an island after raising.[41] (ii) *who* to SpecC, followed by subject to SpecAgrS. This option is prohibited by the extension requirement (or the strict cycle): The latter movement targets a landing site lower than the former's. So far so good.

The third possibility considered by Collins is Chain

[41] Although it remains mysterious what actually accounts for this island status within the MP.

Interleaving (cf. also Kim, Kolb, Müller & Sternefeld: 75ff):

(65) a. was [$_{VP}$ arrested a friend of who]
b. was [$_{VP}$ who$_1$ [$_{VP}$ arrested a friend of t_1]
c. [$_{DP}$ a friend of t_1]$_2$ was [$_{VP}$ who$_1$ [$_{VP}$ arrested t_2]]
d. who$_1$ was [$_{AgrSP}$ [$_{DP}$ a friend of t_1]
[$_{VP}$ t'_1 [$_{VP}$ arrested t_2]]]

The trick is in (65.b): *Who* is extracted from the object position (hence no subject island) and adjoined to VP. The subject is subsequently moved to SpecAgrS (respecting the strict cycle since the VP adjoined position is lower than that). Eventually, *who* is moved further to the Spec of CP (again strictly cyclic). Probably the trace t'_1 is deleted and we're all set, having derived the unacceptable (64).

According to Collins, derivation (65) is blocked by derivation (i) (and possibly (ii) as well) from above, courtesy of Fewest Steps: (i) and (ii) involve two movement operations whereas (65) requires three. Accordingly (65) is uneconomical and out.[42]

How could we block (65) with Fewest Steps gone? We would simply insist that intermediate adjunction to VP such as in (65.b) is illegitimate. Within the *Barriers* framework (Chomsky 1986, Lasnik & Saito 1992) VP adjunction was quite common, but mainly for theory internal reasons (cf. also the remarks in C&T, MP:324). Within the MP, intermediate adjunction is neither necessary nor in fact possible, see the discussion in section 3.3.1. above (see Sabel 1995 for

[42] NB: The ungrammatical derivation blocks the grammatical one. If cyclicity (or rather: the extension requirement) is a 'wired-in' stepwise principle (ii) cannot be a competitor to (65), for it doesn't converge. (i), however, converges since the ECP (or whatever is responsible for subject islands) is not an interface condition (see diagrams above), though it is ultimately ungrammatical (this must be so since otherwise (i) couldn't block (65) either). Hence it blocks the longer but grammatical derivation.

Within the MP we do not find much about barriers/ECP/CED effects other than those that follow from relativized minimality. Following Collins' assumptions (i) would be called *Gibberish* in the MP terminology.

Note that if the argument to be presented goes through we can dispense with the assumption that derivations violating the ECP converge in the first place.

empirical arguments against intermediate adjunction), unless we assume that V has a (presumably) strong D- or Q-feature that attracts *who*. Since this seems an odd assumption we do not need Fewest Steps to block derivation (65).

Other cases discussed by Collins include lowering (followed by LF raising), 'sideways movement,' i.e. in general movements that violate the Proper Binding Condition. It should be evident that such derivations are trivially blocked by the Extension Requirement a.k.a. Cyclicity.

Let us close this section by looking at some cases taken from Epstein 1992 (cf. once more Kim, Kolb, Müller & Sternefeld:67ff). Epstein observes that *wh*-phrases in English are either in SpecC_{+w} or in their base position (as in multiple questions), but that they cannot be partially moved, i.e. scrambled, topicalized or moved to SpecC. Basically the same holds for German (all the examples are taken from Kim, Kolb, Müller & Sternefeld; the (b') examples show where the relevant movements are possible with non-*wh* phrases):

(66) a. Warum hat er den Studentinnen *was* gegeben?
why has he the students what given
b. * Warum hat er *was* den Studentinnen *t* gegeben?
'Why did he give what to the students?'
b! Warum hat er *den Schlüssel* den Studentinnen *t* gegeben?
why has he the key the students given
'Why did he give the key to the students?'

(67) a. Who said [$_{CP}$ that John likes *whom*]?
b. * Who said [$_{CP}$ that [$_{IP}$ *whom* [$_{IP}$ John likes *t*]]]
b! Who said [$_{CP}$ that [$_{IP}$ *Mary* [$_{IP}$ John likes *t*]]]

(68) a. Who thinks [$_{CP}$ John saw *what*]?
b. * Who thinks [$_{CP}$ *what* John saw *t*]?

In (66.b) the *wh*-word *was* ('what') has been scrambled from its base position (indicated by the trace *t*), yielding

ungrammaticality. As opposed to that, scrambling the full DP *den Schlüssel* ('the key') in the very same way, as in (66.b'). is perfectly possible.

Epstein reasons as follows: In the unacceptable (b)-sentences overt movement of the lower *wh*-words takes place, unlike in the (a)-sentences.[43] Since both variants are in the same Competitor Set, the 'shorter' (a)-derivations block the 'longer' (b)-derivations.

Basically the same reasoning, however, can be applied without appeal to Fewest Steps. The point is that there is no reason for moving the lower *wh*-words in the first place. Neither the embedded COMP nor I or V attract any feature of them. Lacking attraction, movement cannot take place.

> On closer inspection it seems that the discussion is besides the point anyway. The question is not why the lower *wh*-words can *not* move, but rather why non-*wh*-elements *can* undergo such optional movements. The only coherent interpretation for phenomena such as (66.b') and (67.b') within the MP seems to be some (optional) feature [scramble] or [topic] which triggers the pertinent movement if present. But then the (b) and (a) examples wouldn't be in the same Competitor Set to begin with, so no economy condition can rule out one in favor of the other.
>
> The substantial question thus seems to be why *wh*-phrases such as those in the (b) examples can *not* bear such features, unlike their counterparts in (b'). It appears that [scramble] or [topic] are incompatible with the *wh*-feature - but this is an observation rather than an explanation.
>
> If this reasoning is correct, the data in (66) through (68) have nothing to do with economy but only with the question which feature combinations are allowed within a numeration and which aren't - a question yet untackled.
>
> That something along this line is on the right track is also suggested by the examples in (69).

---

[43] In case there is LF *wh*-movement - contrary to what is assumed within the MP - the (a) examples would still be shorter, provided that (i) LF movement takes place without an intermediate landing, or (ii) creating an intermediate chain does not count as a movement on its own right (this is the very idea behind the *Form Chain* algorithm discussed in MPLT (MP:182).

If neither (i) nor (ii) were to hold, both derivations would have the same number of steps. Yet the LF version would be preferred because only one out of three steps is overt (as opposed to two out of three with the competitor). Hencé Procrastinate would take care of the issue.

(69) a. * Wer behauptet, *wen* habe sie *t* getroffen?
who claims whom have she met
b. Wer behauptet, sie habe wen getroffen?
who claims she has me whom
'Who claims to have met whom?'
b! Wer behauptet, *den Fritz* habe sie *t* getroffen?
who claims the Fritz has she met
'Who claims that Fritz she met?'

At first glance (69.a) resembles the examples in (68): An in situ *wh*-phrase is illicitly moved to the embedded SpecC. Notice though that (69.a) and (69.b) contain the very same number of steps, since (69.b') is an embedded V2 clause with SpecC overtly filled. The relevant movements are: *wer* ('who') to matrix SpecC and some argument to the embedded SpecC position. Moreover we would rather expect (69.a) to block (69.b) since in the former *wen* ('whom') has a shorter way to go to matrix C at LF than in the latter.

We conclude that economy is irrelevant in these cases. It seems that the pattern boils down to stipulations about the cooccurence of *wh*-features with features such as [topic] or [scramble].

5. How Strong is the Case for Economy?

Let us finally address the question how crucial economy principles, in particular transderivational constraints, are within the MP.

As for the *Minimal Link Condition*, built into Attract/Move, the question is easily answered: It is essential for the very workings of movement theory. This is not really a surprise, though, since the MLC is virtually a variant of Subjacency and Relativized Minimality, both of which are known to be of utmost empirical significance.

What is completely new is the *Last Resort* part of Attract/Move. Its empirical consequences, too, are remarkable and mostly deal with a question that was often ignored in earlier works, namely: What prohibits certain movements in one language, but permits them in another? I should hasten to add, though, that the MP faces another question instead, namely: What prohibits certain feature configurations in one language, but permits them in the

other? To illustrate this, note that if some type of movement, say scrambling or object raising, is attested in some language L, the likely conclusion is that there is a (strong) feature F triggering it. If that very movement is systematically absent in some other language L', this cannot simply be attributed to Last Resort. Rather we have to answer the question what prohibits the occurrence of feature F in language L'. In that sense Last Resort simply reverses the mode of proceeding: Instead of stipulatively prohibiting substitution into and adjunction to all those positions which are not attested as landing sites in L' we now stipulatively assume (possibly optional) strong features in all those positions attested as landing sites in L.

Let us now turn to the transderivational constraints. We have already seen that evidence in favor of *Fewest Steps* is rare, if existent at all, leading us to suspect that it is not operative in the theory sketched in C&T.

What about *Procrastinate*? To answer this question we should distinguish two main effects of Procrastinate. On the one hand Procrastinate serves to prohibit overt movement, where covert movement is possible. This function is of course crucial in the treatment of word order variation among languages. As argued at the beginning of subsection 3.2. above, this function does not require a transderivational constraint. If Attract would be modified by something like 'Weak features are invisible/ignored before Spell-Out', it would completely account for this function (unless there are cases in which a weak feature triggers overt movement in order to ensure convergence, a case not discussed within the MP).

On the other hand, Procrastinate forces - as a by-product if you like - insertion instead of early overt movement, if possible, as we have discussed in section 3.2 above. This effect can hardly be captured by an intraderivational constraint, because Procrastinate has to 'look ahead' here, keeping an eye on convergence.

It thus seems that (43) above is *the* crucial argument for Procrastinate and for transderivational constraints quite in general (but remember that (48) appears to be a

counterargument to that version of Procrastinate!). To the extend that they can or should be treated otherwise, a grammar without transderivational constraints seems possible (see also the discussion in Sternefeld 1995, where such a grammar is argued for).

That is not to say that such a grammar would not adhere to 'economy principles.' As said above I know of no precise and coherent definition of the concept economy, but it seems reasonable to call a grammar that heavily relies on the Minimal Link Condition and Last Resort economy based. It might thus be the case that Chomsky's conclusion that the Language Faculty is crucially based on economy principles is correct without there necessarily being transderivational mechanisms in it. Someone who shares Chomsky's general skepticism towards transderivational constraints will consider this Good News. One might even suspect that the much-cited minimalist spirit would prefer a grammar that does not require transderivational constraints (this is at least suggested by the remarks in C&T). If that should be correct, the appendix is obsolete.

6. Appendix: Some Notes on Local Economy, its Complexity, and its Implementation

We have at several occasions mentioned the local implementation of transderivational constraints - in particular Procrastinate. As already mentioned, Chomsky's main concern seems to be not with the empirical differences but with the computational complexity required by global transderivational constraints.

The problem - according to Chomsky - is this: Comparing all derivations based on a given numeration is a complex enterprise on two counts. First, all those derivations have to be generated, and second they have to be evaluated wrt. the pertinent constraints, say number and location of steps. Presumably the set in question is very large, infinite in the worst case. The question then is: Is a system that requires complex operations on gigantic sets a plausible candidate for a cognitive system?

In this appendix I deal with three and a half questions around this issue. The first question is whether the local implementation is possible in the first place. The answer is going to be 'yes,' but at the price of some rather unminimalistic modifications. This is demonstrated in 6.1. The second question then is whether the local implementation is computationally less complex than the global one. The answer is again 'yes' though the difference might not be crucial (section 6.2.). The third question is whether there is a well defined question about computational complexity with the two implementations, and the answer is going to be 'no'. Comparing local and global implementation is like comparing apples and pears (6.3). Which straightforwardly brings us to the last half issue, namely whether the problem is a problem in the first place. At the end of this section some thoughts are presented to the effect that complexity or even infinity is nothing to be afraid of. These remarks, however, are highly tentative, intended basically to provoke discussion.

6.1. How can we locally implement Procrastinate?

We have already hinted at some difficulties that arise with implementing Procrastinate locally (all these remarks carry - *mutatis mutandis* - over to locally implemented transderivational constraints in general). Put briefly, the two difficulties are:

D1 We have to make precise the notion of 'identity of movement'

D2 We have to make sure that Procrastinate - albeit operating in the middle of generating the structure - can 'look ahead' so as to guarantee that the chosen continuation eventually converges

Let us elaborate on these in turn.

6.1.1. Procrastinate and Identity of Movement

In subsection 2.2. above I suggested that the only way to tackle D1 is by identifying 'movement' with a pair of phrase markers that can be related by the operation Move.

Identical movements are then defined via identity of the pairs representing them. It was also pointed out that this treatment necessitates a total ordering on the order of rule application within a derivation, presumably X^0 adjunction before substitution before XP adjunction. I will continue to assume this here.

Then Procrastinate could be formulated along the following lines (definitions to be revised are preceded by a star):

(70) Let a. D_N be the set of *numerations* and
b. D_{PM} the set of *phrase markers*

*(71) Let D_δ be the set of *derivational stages* δ, where $\delta = \langle N,P\rangle$, with $N \in D_N$ and $P \in D_{PM}$.

(72) Let D_Δ be the set of *derivations*. Δ is a derivation iff Δ is an *n*-membered sequence[44] of derivational stages, such that
a. the initial stage δ_0 equals <N,->
b. the final stage δ_n equals <-,P> and there is no $\delta^* \in D_\delta$ such that δ^* can be derived from δ_n
c. for all δ_i, δ_{i+1}, δ_{i+1} is derived from δ_i by application of Move or Merge

(73) Competitor Set: For all derivational stages $\delta \in D_\delta$, the *Competitor Set* $\kappa(\delta)$ is the set of all $\Delta' \in D_\Delta$ such that
a. $\delta \in \Delta'$
b. Δ' converges

*(74) Procrastinate: Move can only apply to δ, forming δ', if for all derivations $\Delta \in \kappa(\delta)$, $\langle\delta,\delta'\rangle$ is a subsequence of Δ and δ' is before Spell-Out

The treatment as sketched, however, is based on a substantial simplification in (71), namely that a

---

[44] Strictly speaking Δ is not a *sequence* if we consider the Spell-Out to PF computation as part of the derivation. I will ignore this henceforth.

derivational stage can be equated with a phrase marker (plus the numeration). Recall, however, that derivational stages within the MP consists of a *set of phrase markers* plus the numeration. At least before Spell-Out, parts of the phrase marker are under construction independently from each other, before they are finally merged into one big tree. Clearly, Procrastinate has to be operative in constructing these subtrees.

It should be obvious that the ordering of operations in different subtrees is in no natural way determined. In other words, if the derivation of *if Peter comes he is happy* has reached the stage at which - among others - the phrase markers *AGR* $[_{VP}$ *Peter comes]* and *AGR** $[_{VP}$ *he is happy]* are already constructed, it is completely open (and in fact irrelevant) whether subject raising will take place first in the *if* clause or in the main clause. We therefore have to revise the definition of derivational stage:

(75) Let D_δ be the set of *derivational stages* δ, where $\delta = \langle N,P\rangle$, $N \in D_N$ a numeration and $P \in \wp D_{PM}$ a set of phrase markers.

Move operates on phrase markers, not derivational stages. We therefore introduce the function MOVE, which operates on derivational stages (we only do this for tree extending movement here)(*Move* in (72.c) must then of course be replaced by MOVE):

(76) MOVE is a relation in D_δ, such that for all N, $N' \in D_N$ and all S, $S' \in D_{PM}$, MOVE($\langle N,S\rangle,\langle N',S'\rangle$) holds iff there is a phrase marker $B \in S$, and
  a. the root R of B has some sublabel that attracts feature α in B
  b. B* is formed by copying α (pace pied piping) to R
  c. $N' = N$ and $S = (S-\{B\}) \cup \{B^*\}$
  d. Procrastinate: for all derivations $\Delta \in \kappa(\langle N,S\rangle)$ there is a $\delta = \langle N'',S''\rangle \in \Delta$, such that $B^* \in S''$ and δ is overt

This solves D1, provided that we accept the extraneous

ordering on movements. (76.d) merely requires that every competing derivation must overtly contain the result of Move α as well (note that it has to contain the *input* to Move α by the definition of Competitor Set).

6.1.2. Move and C-Move

As already hinted at in section 3.2.1. the local implementation of Procrastinate requires yet another complication of the formalism, one that has to do with D2 above. As said there, Procrastinate must do 'advance calculation'. It has to generate the Competitor Set at the point where movement is at issue. In sections 2 through 5 we took the Competitor Set to be the set of convergent derivations based on the same numeration. This was straightforward since under the global implementation economy considerations apply so to speak 'after' LF and PF. Understanding Procrastinate to work in the course of the generation itself changes the picture. Now a derivation cannot be generated if it locally violates Procrastinate. That means that Procrastinate codetermines the set of convergent derivations. Accordingly, the Competitor Set used by Procrastinate cannot be the set of eventually converging derivations, but only the set of derivations *that would converge had it not been for Procrastinate*. In other words, to generate the Competitor Set we use the usual operations Move and Merge as well as the usual interface conditions such as FI but abstract away from all economy conditions. In effect, we thus reintroduce the dichotomy between *convergent* and *economical*.

This in turn necessitates a number of complications in the formalism. In (76), Procrastinate is part of the definition of MOVE, and MOVE in turn is part of the definition of Δ, the set of derivations. Let us assume the following definition for *convergence*:

(77) Δ *converges* iff
  a. LF(Δ) meets FI
  b. PF(Δ) meets FI
  (where LF(Δ) and PF(Δ) abbreviate the respective derivational stages $\delta \in \Delta$)

The Competitor Set consists of non-procrastinating convergent derivations. Since by definition (72) all elements of D_Δ obey Procrastinate, the Competitor Set cannot be a subset of D_Δ. We thus define another domain $D_{\Delta c}$:

(78) Let $D_{\Delta c}$ be the set of *c-derivations*. Δ is a c-derivation iff Δ is a sequence of n derivational stages $\delta \in D_\delta$ such that
   a. the initial stage δ_0 equals <N,{}>
   b. the final stage δ_n equals <-,P> and there is no $\delta^* \in D_\delta$ such that δ^* can be derived from δ_n
   c. for all δ_i, δ_{i+1}, δ_{i+1} is derived from δ_i by application of c-Move or Merge

c-Move in (78.c) is of course just MOVE without Procrastinate:

(79) C-Move is a relation in D_δ, such that for all N, $N' \in D_N$ and all S, $S' \in D_{PM}$, C-Move(<N,S>,<N',S'>) holds iff there is a phrase marker $B \in S$, and
   a. the root R of B has some sublabel that attracts feature α in B
   b. B* is formed by copying α (pace pied piping) to R
   c. $N' = N$ and $S = (S-\{B\}) \cup \{B^*\}$

(We can now define MOVE using c-Move in the obvious way.) The Competitor Set can now be defined as a subset of $D_{\Delta c}$:

(80) Competitor Set: For all derivational stages $\delta \in D_\delta$, the *Competitor Set* $\kappa(\delta)$ is the set of all $\Delta' \in D_{\Delta c}$ such that
   a. $\delta \in \Delta'$
   b. Δ' converges

(81) Economy: Δ is *economical* iff $\Delta \in D_\Delta$ and Δ converges

The system as sketched here allows for a local implementation of Procrastinate. It requires in turn (i) imposing an extraneous ordering on movements within a phrase marker, and (ii) incorporating Procrastinate into

MOVE, thereby necessitating two different concepts of (converging) derivation.

Further issues arise if we consider more than one transderivational constraint. Does the generation of the Competitor Set for Procrastinate obey Fewest Steps? And vice versa? Or is the Competitor Set universally generated regardless of *any* transderivational principles? I will not dwell on these questions. They are the notational variants of the question discussed in section 2 i.e. whether global transderivational constraints apply conjunctively or subsequently.

6.2. How Much Simpler is Local Economy?

As repeatedly noted, local economy conditions are nevertheless transderivational. They require generation and comparison between the members of a set of derivations. As such they are complex operations. This is very much unlike the version of e.g. the Minimal Link Condition as presented above, which doesn't make reference to alternative derivations at all. No doubt a grammar without transderivational constraints would be simpler and presumably computationally less complex than one that uses them (see the remarks in section 5). But that is not the issue here.

The issue is to compare two systems, both including transderivational constraints. Let me first start with an strong allegation:

Strong Allegation: Global Economy requires to compute the Competitor Set once, while local Economy needs to compute it every time movement is at stake. Therefore local economy is more complex, not the other way around.

The reason why I think this allegation is besides the point is that there presumably is a trivial way of calculating the Competitor Set at stage δ_i if it has already been computed for some earlier stage δ_{i+k} of the same derivation. If $\kappa(\delta_{i+k})$ is the earlier set, then $\kappa(\delta_i)$ is the subset containing those $\Delta \in \kappa(\delta_{i+k})$ that contain δ_i as a derivational

stage. If something along these lines is correct, repeated calculation of the Competitor Set does not pose a relevant computational burden on the theory. Following the principle of *in dubio pro reo*, let us assume so and take the Strong Allegation to be refuted.

Extending this reasoning it seems that given the local implementation, too, the Competitor Set really has to be calculated only once, namely the very first time movement is at stake. Let us call this stage δ_{fm} (for 'first movement'). On all later occasions the above procedure is applied. This then leads us to the second, weaker allegation:

Weaker Allegation: The computational complexity involved in calculating the Competitor Set at stage δ_{fm} in the local implementation is in fact just as big as that for calculating the global Competitor Set. Therefore both version are equally complex and none should be preferred on these grounds.

Though the general line of this argument is valid, it is not really correct. The reason is that calculating $\kappa(\delta_{fm})$, though presumably quite early in the derivation, is still less complex than calculating the global Competitor Set. To see this, note that calculating the global Competitor Set is nothing other than calculating the local Competitor Set for the initial stage δ_0 of the derivation, i.e. the stage where there is nothing but the numeration (P, in the sense of (72) and (75) above, is the empty set). The question is thus: Is, for any derivation Δ, calculating $\kappa(\delta_0)$ more complex than calculating $\kappa(\delta_{fm})$? Let us make the assumption that the computational complexity involved in calculating a Competitor Set is proportional to the cardinality of that set.[45] Now note that computing the Competitor Set involves

[45] This is a slight simplification, at least if we stick to the definitions given above. According to these, the Competitor Set consist of convergent derivations only, i.e. derivations that reach the interfaces and conform to the principles operative there. However, in calculating the Competitor Set the grammar will try out numerous 'dead ends', i.e. derivations that can't be carried any further (i.e. never grow into a single tree) or simply don't converge. Calculating these just the same requires computational power, hence enhances complexity. Definition (78) above should therefore be changed so that partial

not only Move but also Merge. At the point δ_{fm}, Merge (though of course not Move) can (and presumably will) have taken place several times. In fact, the more structure Merge has built up before any movement is considered (e.g. before the first [-interpretable] feature is encountered), the less possibilities there are to continue the derivation. Accordingly, $\kappa(\delta_{fm})$ will in fact be smaller in cardinality than $\kappa(\delta_0)$, and so will the computational complexity required. So there *is* a difference in computational complexity, though its magnitude is hard to estimate. Partly it depends on pure accident. If a lot of mergings are done before considering movement, $\kappa(\delta_{fm})$ will be considerably smaller than $\kappa(\delta_0)$, but if fate has it that an attracting feature is involved very early, the difference melts down to practically nothing.

The next question is of course whether the difference, say in the optimal case (i.e. lots of Merge before Move), is *significant* wrt. the overall complexity of computations necessary even in that case. For example, if it should turn out that the complexity involved in calculating $\kappa(\delta_0)$ is about 1.1 times bigger than that involved in calculating $\kappa(\delta_{fm})$ we might consider this an irrelevant reduction of computational complexity. It is beyond the scope of this paper to prove or disprove that the situation is like this. I am inclined to think, though, that computational improvements are in fact minimal in relation to the overall complexity involved. To see why, the reader is invited for a short excursus regarding Merge.

6.2.1. Short Excursus Regarding Merge

In its local implementation MOVE checks with every step whether the resulting derivation stage can ultimately yield a converging derivation. We can say that MOVE (though not c-Move, which is required, too, see 6.1. above) doesn't enter dead ends, i.e. continuations that do not yield a single phrase marker or simply don't converge (this is trivial because it shifts the burden of stupidly trying out

---

derivations, too, are part of $D_{\Delta c}$, by changing (72.b, 78.b) to (i)
(i) there is no $\delta^* \in D_s$ such that δ^* can be derived from δ_n
The computational complexity for calculating $\kappa(\delta_i)$ can then reasonably be equated with the cardinality of the set $\{\Delta:\Delta\in D_{\Delta c}\ \&\ \delta_i \in \Delta\}$.

every continuation to c-Move, the operation that computes the Competitor Set). Nevertheless, generating an actual derivation is not deterministic. The reason is that there are no comparable constraints on Merge.

Merge, it seems, is quite unrestricted as to what it does. In the worst case it could senselessly combine arbitrarily chosen elements: Stick Vs on top AgrPs, give intransitive verbs an object, embed Cs under adjectives and so forth. It seems likely that the number of 'mistakes' Merge can do both in calculating the Competitor Set and in trying to complete the actual derivation exceeds the number of errors c-Move makes. The reason is that Move (or rather Attract) has severe intraderivational preconditions on its application, such as Last Resort, MLC etc.

Assume we depart from minimalist assumptions in making Merge more intelligent. We could make it sensitive to selection, subcategorization and possibly even the phrase structure status of the objects it deals with (e.g. 'Is this would-be complement saturated?' etc.). Yet Merge would make mistakes, see e.g. the discussion in section 3.2. where an otherwise perfect Merge leads to a non-convergent derivation.

As a consequence, generating a derivation is *not* deterministic, even with such an enlightened Merge. In fact there isn't a simple algorithm to generate a converging derivation from a given numeration, even if there is one (see next subsection).

Of course we could give Merge some look-ahead as well. Suppose that just as with MOVE we check before merging whether the resulting structure can ultimately yield a convergent derivation. Such a local but transderivationally constrained Merge could be defined as in (82).[46]

---

[46] Here, as before, we pretend - for ease of exposition - that the numeration is simply a set of lexical items, rather than a set of pairs of lexical items and natural numbers, as assumed in the MP (e.g. p.225f). The necessary changes to the 'real' theory are straightforward.

(82) TD-Merge is a relation in D_δ, such that for all <N,S>, <N′,S′> $\in D_\delta$, TD-MERGE(<N,S>,<N′,S′>) iff

a. there are two elements (lexical entries or phrase markers) B, B′ $\in S \cup N$ such that B* results from joining B and B′ and

b. N′=N-{B,B′} and S′ = (S-{B,B′})$\cup${B*} and

c. κ(<N′,S′>) is non-empty

The crucial clause is (82.c) (without it we have simple Merge, which we haven't defined before): TD-Merge - just as MOVE - calculates the local Competitor Set and settles on some merger only if the result is a candidate for a converging derivation. Move and TD-Merge in tandem do in fact generate a convergent and economical derivation, provided there is one. Otherwise it would stop after the first attempt at applying Merge or Move.

But miracles don't happen and so it shouldn't come as a surprise that the computational complexity required with such a grammar is just as big as that involved in calculating global Competitor Sets. The proof is easy: The very first step in the derivation (i.e. the transition from δ_0 to δ_1) - necessarily Merge, though that is irrelevant here - requires calculating the local Competitor Set, $\kappa(\delta_1)$, which obviously is almost as complex as $\kappa(\delta_0)$.

In sum, generating a convergent derivation using MOVE and local Procrastinate might be slightly less complex, but still very complex, due to the unrestrictedness of Merge. Making Merge transderivational avoids such 'mistakes' in building the actual derivation but shifts the burden to the calculation of the local Competitor Set, which is just as complex as that of the global Competitor Set.

In general it is unclear why Move should be restricted in the way envisaged by Chomsky while other sources of complexity remain untouched. Besides Merge, note that e.g. the selection of a numeration is completely unrestricted. Most logically possible numerations will in fact never yield a convergent derivation, and since there are no restrictions on building numerations, so will the grammar

(thanks to Susann Siebert for pointing this out).[47] Restricting all of these will in all likelihood result in a system that is just as complex as the global one.

6.3. What exactly is the Question of Computational Complexity?

Now that we have sharpened our understanding of what determines computational complexity in which ways we are in a position to step back and ask the more general question heading this section. We have seen above that the local implementation is likely to involve less complex computations. According to Chomsky's premise (which we will challenge at the end of this appendix) low computational complexity correlates with higher cognitive plausibility. The question, however, is: Computational complexity for *what*?

Let us consider the global model first: Here we start by calculating all possible derivations from a given numeration. Among those which made it to the interfaces (i.e. excluding those that failed to join in one phrase marker, thereby stopping at Spell-Out) we select the ones that converge and - among them - crown the most economical one(s). The grammar as a whole then succeeds in determining the set of grammatical derivations from a given numeration. Note that this set is possibly empty, namely in all those cases where the numeration is already 'ill-formed', i.e. allows for no convergent derivation at all (see above). And since there are no restrictions on numerations this is likely to happen most of the time.

With the local model we generate *one* possible derivation from a given numeration. Of course, if the numeration is ill-formed in the sense alluded to above, no derivation will actually converge. But even if it is not, i.e. if there *is* a possible convergent derivation based on the pertinent numeration, the local model - unlike the

---

[47] This might not be entirely correct since Chomsky tentatively assumes a principle on numeration formation, according to which an element may only enter the numeration if it has an effect on output (C&T (76),MP:294). Needless to say that such a condition multiplies computational complexity far beyond the point considered so far.

global one - is not guaranteed to find it. The reason, as pointed out above, is Merge. If Merge does wrong at some point the derivation is not going to converge.[48]

For the sake of the argument, what about the model with local Move and transderivational Merge, as sketched in subsubsection 6.2.1. above? That model is guaranteed to generate a convergent derivation (if there is one). It differs from the global implementation only in that it finds *one*, not *all*. But recall that here, computational complexity is just as big as with the global model.[49]

In sum, the three models considered differ wrt. their computational complexity as well as wrt. their 'abilities' or coverage, as the following chart illustrates:[50]

the	*is*	*and - given a numeration - generates*
Global Model	most complex	all convergent derivations
Local Model with TD-Merge	slightly less complex	one convergent derivation
Local Model	considerably less complex	one convergent derivation or nothing at all

Obviously the difference in computational complexity

---

[48] It might be tempting to think of some sort of 'backtracking' algorithm that re-starts the derivation from the point of the last Merge. However, such a model has just the same complexity/ability ratio as the other ones.

[49] As a matter of fact it is slightly smaller since, as mentioned above, the local model has to calculate $\kappa(\delta_1)$ while the global model has to calculate $\kappa(\delta_0)$. This difference in computational complexity corresponds to the difference in coverage: The global model finds *all* convergent derivations, the local only *one*.

[50] One might supplement this chart by a model that builds numerations with 'look ahead' (see FN 47 above). Such a model would range in top position due to its incredible computational complexity, but would also top the others in terms of abilities because it would be the only one that is guaranteed to generate a derivation that corresponds to an acceptable sentence of the language (*pace* gibberish of course). Note that all the models discussed in the chart are helplessly dependent on a decent numeration.

correlates with the difference in ability. The point of this is that comparing the models in the way done in 6.2. is just not possible since they do completely different things. Moreover it seems a good bet that augmenting one of the simpler models with the means to reach the abilities of the global one would bring with it a corresponding increase in computational complexity.

A further point is whether any of the tasks the models are capable of carrying out should be subjected to considerations of computability. In effect Chomsky's worry is whether the generative grammar can compute some rudiments of a derivation, a convergent derivation or all convergent derivation given a numeration in finite and - ideally - short time. But why should that matter? Note that these questions are crucially different from (83.a), let alone (83.b).

(83) a. Can the grammar produce some acceptable sentence in finite/short time?
b. Can the grammar decide in finite/short time whether a given sentence belongs to the language?

It seems to me that these questions, if any, are relevant in terms of cognitive plausibility (see also the remarks in MP:161 and 201, which suggest that Chomsky is actually interested in the question of parsability, which - if anything - is related to (83.b)), but there are not touched by considerations of global vs. local implementation.

> Note that the only way to solve task (83.b) with any generative grammar as stated in theoretical linguistics is to generate the complete set of expressions that constitute the language and then see whether sentence S is in that set. But since languages - even in the formal sense - are infinite, this cannot be achieved, regardless of the type of grammar in question.
>
> This is different from saying that it is impossible to construct a parser that incorporates the principles found in, say, PPT work and is able to assign a structure to any given sentence of the language (and none if the sentence is ungrammatical) in finite time. But then we are dealing with the

> computational properties of the parsing algorithm rather than those of the model of grammar underlying it. Note, for example, that an LGB-type grammar provides no means to generate a grammatical sentence except for trial-and-error (neither do grammars of the PSG type). Yet there are parsers based on these theories that can fulfil tasks like these.

These considerations bring us to the last, most general question: Is it reasonable to regard computational properties of a model of grammar as criteria for its cognitive plausibility? Apparently Chomsky would give a positive answer to that question, as evidenced by his concern with computability.[51] Another position would be to say that this is just too narrow an interpretation of the computer metaphor. One might as well ask: (i) Are the very computational steps carried out be the grammar carried out by the mind/brain implementing that grammar, too? And if yes, (ii) Do we have to assume that the implementation of that grammar in the mind/brain has the same properties wrt. computational complexity as the formalism used by the grammar?

Question (ii) for example is answered negatively by the proponents of Optimality Theory. Optimality Theory standardly operates with infinite Competitor Sets (cf. section 2. above). A serial processor would thus take quite a long time to carry out the task, if it ever could. It is shown, however, that systems that use parallel processing algorithms, i.e. artificial neural networks, can carry out equivalent computations in finite time. Suppose this much. Given the fact that the mind/brain appears to have a parallel rather than a serial architecture, the problem of infinite sets might in fact turn out to be one of the *emulation* with serial algorithms rather than one of the theory *per se*. We shouldn't therefore disregard the possibility that fear of infinity (or computational complexity) is misguided in this context.

---

[51] 'Computational complexity may or may not be an empirical defect; it is a question of whether the cases are correctly characterized (e.g., with complexity properly relating to parsing difficulty...)' (MPLT, MP:201).

7. References

Baker, Mark C. (1988) *Incorporation*. Chicago: University of Chicago Press.

Cavar, Damir & Chris Wilder (1994) Word Order Variation, Verb Movement, and Economy Principles. *Studia Linguistica* 48. 46-86.

Collins, Chris (1994) Economy of Derivation and the Generalized Proper Biding Condition. *Linguistic Inquiry* 25. 45-61.

Chomsky, Noam (1981) *Lectures on Government and Binding*. Dordrecht: Foris.

Chomsky, Noam (1986) *Barriers*. Cambridge MA: MIT Press.

Chomsky, Noam (1991) Some Notes on Economy of Derivation and Representation. In: Freidin, Robert (ed.) *Principles and Parameters in Comparative Grammar*. Cambridge, MA: MIT Press.

Chomsky, Noam (1993) A Minimalist Program for Linguistic Theory. In: Hale, Ken & Jay Keyser (eds.) *A View from Building Twenty*. Cambridge, MA: MIT Press. 1-52.

Chomsky, Noam (1995) *The Minimalist Program*. Cambridge, MA: MIT-Press.

Epstein, Samuel David (1992) Derivational Constraints on $\bar{A}$-chain formation. *Linguistic Inquiry* 23. 235-259.

Fox, Danny (1995) Economy and Scope. *Natural Language Semantics* 3. 283-341.

Grimshaw, Jane (1993) *Minimal Projections, Heads, and Optimality*. Ms. Rutgers University.

Kim, Shin-Sook & Hans-Peter Kolb & Gereon Müller & Wolfgang Sternefeld (1995) Minimalismus in der Syntax. *SfS Reprt* 03-95. Tübingen.

Kitahara, Hisatsugu (1993) Deducing Superiority Effects from the Shortest Chain Requirement. *Harvard Working Papers in Linguistics* 3. 109-119.

Lasnik, Howard (1993) Lectures in Minimalist Syntax. *UConn Occasional Papers in Linguistics* 1. Cambridge, MA: MITWPL.

Lasnik, Howard & Mamoru Saito (1992) *Move α*. Cambridge, MA: MIT-Press.

Marantz, Alec (1995) The Minimalist Program. In: Webelhuth G. (ed) *Government and Binding Theory and the*

Minimalist Program. London: Blackwell. 349-382.

May, Robert (1985) *Logical Form*. Cambridge, MA: MIT-Press.

Müller, Gereon (1995) *Partielle W-Bewegung und Optimalität*. Talk given at GGS 1995, Jena.

Prince, Alan & Paul Smolensky (1993) Optimality: A Theory of Constraint Interaction. *RuCCS Technical Reports* 2. Rutgers University.

Reinhart, Tanya (1994) WH-in-Situ im the Framework of the Minimalist Program. *OTS Working Papers*. Utrecht.

Rizzi, Luigi (1990) *Relativized Minimality*. Cambridge, MA: MIT-Press.

Sabel, Joachim (1995) *Intermediate Traces, Adjunct Movement, and Minimize Chain Links*. Ms. Frankfurt University.

Sternefeld, Wolfgang (1996) *Competing Reference Sets*. Ms. Tübingen University.

Index