Inkelas, S. and D. Zec 1995. "Syntax-phonology interface." In J. A. Goldsmith, ed., *The Handbook of Phonological Theory*. Oxford: Blackwell, pp. 535–49.
Jackendoff, R. 1977. "X'-Syntax: a study of phrase structure." Cambridge, MA: MIT Press.
Kayne, R. 1994. *The Antisymmetry of Syntax*. Cambridge, MA: MIT Press.
Koopman, H. 1994. "Licensing heads." In D. Lightfoot and N. Hornstein, eds., *Verb Movement*. Cambridge: Cambridge University Press, pp. 261–96.
Koopman, H. and D. Sportiche 1991. "The position of subjects." *Lingua* 85: 211–58.
Odijk, J. 1997. "C-Selection and s-selection." *Linguistics Inquiry* 28: 365–71.
Pesetsky, D. 1982. "Paths and categories." Doctoral dissertation, MIT, Cambridge, MA.
Rizzi, L. 1990. *Relativized Minimality*. Cambridge, MA: MIT Press.
Rothstein, S. D. 1991a. "Syntactic licensing and subcategorization." In *Syntax and Semantics*, Vol. 25. San Diego: Academic Press, pp. 139–57.
Rothstein, S. D. 1991b. "Heads, projections and category determination." In K. Leffel and D. Bouchard, eds., *Views on Phrase Structure*, Dordrecht: Kluwer Academic Publishers, pp. 97–112.
Rubin, E. J. 1996. "The transparent syntax and semantics of modifiers." West Coast Conference on Formal Linguistics, 15.
Sag, I. and C. Pollard 1989. "Subcategorization and head-driven phrase structure." In M. Baltin and A. Kroch, eds., *Alternative Conceptions of Phrase Structure*. Chicago: University of Chicago Press, pp. 139–81.
Seely, T. D. 2000. "On projection-free syntax." MS, Eastern Michigan University.
Speas, M. 1990. *Phrase Structure in Natural Language*. Dordrecht: Kluwer.
Svenonius, P. 1994a. C-Selection as feature checking. *Studia Linguistica* 48.2: 133–55.
Svenonius, P. 1994b. "Dependent nexus." PhD dissertation, University of California, Santa Cruz.
Truckenbrodt, H. 1999. "On the relation between syntactic phrases and phonological phrases." *Linguistic Inquiry* 30: 219–55.
Ura, H. 1995. "Towards a theory of 'strictly derivational' economy conditions." *MIT Working Papers in Linguistics* 27: 243–67.
Ura, H. 1996. "Multiple feature-checking: a theory of grammatical function splitting." Doctoral dissertation, MIT, Cambridge, MA.
Uriagereka, J. 1999. "Multiple Spell Out." In S. D. Epstein and N. Hornstein, eds., *Working Minimalism*. Cambridge, MA: MIT Press, pp. 251–82.
van Riemsdijk, H. 1998. "Categorial feature magnetism: the endocentricity and distribution of projection." *Journal of Comparative Germanic Linguistics* 2: 1–48.
Watanabe, A. 1996. *Case Absorption and Wh-Agreement*. Dordrecht: Kluwer.
Yang, C. 1997. "Minimal computation: derivation of syntactic structures." MS, MIT, Cambridge, MA.
Zec, D. and S. Inkelas 1990. "Prosodically constrained syntax." In S. Inkelas and D. Zec, eds., *The Phonology–Syntax Connection*. Stanford, CA: Center for the Study of Language and Information.

Chapter three

Rule Applications as Cycles in a Level-free Syntax

Samuel David Epstein and
T. Daniel Seely

1 Introduction: the role of Minimalist method

Minimalism can be characterized as the specification of a method of inquiry under which it is hoped that explanatory adequacy will be advanced – explanatory adequacy in both the general scientific sense and the linguistic-specific sense (of explaining how humans develop knowledge of language).[1] A well-known assumption of Minimalist syntactic inquiry is that sound and meaning are ineliminable, and correspondingly the two interface levels of PF and LF, and only these, are postulated. Certain legibility conditions for these interface levels are hypothesized, including the arguably natural conditions that every element appearing in a PF representation must have a phonetic interpretation and every element in an LF representation must have a semantic interpretation.

Lexical items, each composed of a bundle of features, also seem ineliminable. Some features of lexical items are illegitimate at one or the other interface. For instance, certain features have no LF interpretation but do have a PF interpretation; and the structural Case feature borne by N is an important instance of this. The pronoun *him* seems synonymous with *he*, although their PF interpretations are distinct, as evidenced by their different pronunciations. Thus, given the legibility condition that only LF-interpretable features can appear in an LF representation, it follows that there must be some mechanism by which the LF-uninterpretable Case feature of an N (DP) is removed before an LF representation is generated. The mechanism is assumed to be syntactic (*he* vs. *him* being distributionally determined); in other words, in certain "licensing configurations" the Case feature can be removed, yielding an LF legitimate Case-free DP.

This is an oversimplified, but (we hope) illustrative review of how a milestone in the development of an explanatory theory of syntax, namely Chomsky's eliminative explanation of the Case Filter and its reduction to independently motivated apparatus, is facilitated by the Minimalist method. Lying at the heart of this method is the identification of the legibility conditions, and determination

of what might constitute an optimal "solution" to them. Proceeding in this manner allows one to identify what represents a departure from optimal assumptions, thereby identifying topics for further investigation. As Chomsky (2001a: 2, "Derivation by phase," henceforth DBP) writes:

> the Strong Minimalist Thesis sets an appropriate standard for true explanation; anything that falls short is to that extent descriptive, introducing mechanisms that would not be found in a "more perfect" system satisfying only legibility conditions.

Thus, Minimalist method first seeks to determine "in the abstract" what would constitute an optimal solution. With this in mind, the data is then analyzed and we seek to construct an analysis evincing as little departure from these assumptions as is possible, the ideal being no departure from these assumptions at all.

This is precisely how DBP proceeds. On the basis of just the legibility conditions, feature interpretability at the interfaces, and feature valuation, DBP argues that Spell Out must be cyclic. Not wanting to *stipulate* the cycle in this derivational system, DBP seeks to *explain* why there is cyclic Spell Out (i.e. more than one application of Spell Out in a derivation), arguing that this central departure from the Y-model architecture "follows naturally from examination of feature interpretability" (DBP, p. 5). The explanation of cyclicity (what we call Chomsky's "general" argument) hypothesizes that such cyclicity follows from the generative on-line derivational process of feature valuation and legibility. Importantly, this argument, guided by Minimalist method, provides in effect the "target" for the empirical analysis. The analyses proposed in DBP are designed to conform to the general argument that Spell Out must be cyclic. The general argument then raises a question that centrally concerns DBP: "If it follows naturally that Spell Out must indeed be cyclic, which particular categories undergo cyclic Spell Out?" And DBP argues that the categories that undergo cyclic Spell Out, i.e. the "phases," are vP and CP. Thus, on the one hand, there is the general argument in DBP that there must be cyclic Spell Out; and, on the other hand, there is the specific argument(s) that cyclic Spell Out proceeds via vP and CP phases, and hence that there is derivation by phase.

In what follows, we will suggest that the general, methodologically guided argument seeking to explain the existence of cyclic Spell Out in DBP is problematic; a detailed presentation of what we take to be the problem is given in section 2. The basic empirical problem is that Spell Out does not have the required information to operate properly before or after feature valuation; i.e. neither the prevaluation nor postvaluation representation is sufficient; hence it can't operate at all. As a result the DBP general argument that iterative feature valuation naturally induces iterative (i.e. cyclic) Spell Out cannot be maintained. We will then suggest possible approaches to the problem. In our view, the most natural approach overcomes the problem confronting the timing of cyclic Spell Out in DBP by assuming that Spell Out is derivational and applies to the output of every transformational rule application (as in the level-eliminating system of Epstein et al. 1998, and Epstein and Seely 1999, forthcoming). Applying Spell Out in this manner seems conceptually optimal in that it requires no account of why some categories get spelled out cyclically, i.e. constitute phases (a central concern of DBP), while others do not. In short, spelling out at each transformational rule application not only readily overcomes the precise problem in DBP we reveal, but is also "natural" in requiring no new mechanisms beyond those already postulated in a derivational system. This model incorporates ordered transformational rule application but eliminates levels of representation in the standard sense, instead placing them or their effects "inside" transformational rule application. Moreover, the architecturally suspect, empirically problematic, and computationally inefficient mechanisms of lookahead, as well as lookback (each discussed explicitly below), are rendered unnecessary precisely because the interfaces are reached by each transformational rule application. We suggest that putting Spell Out "inside" transformational rule application is not only the null hypothesis, but may represent the only way to maintain the DBP *explanation* of cyclicity as following naturally from feature valuation and interpretability.

Overall, we seek to eliminate all levels, and thus overcome a problem confronting the application of Spell Out in DBP, which requires appeal to an unspecified lookback algorithm, which is needed to find the previously unvalued form of a now valued feature. We seek to solve the ordering problems inducing lookback Spell Out (and also lookahead) by "bringing the interfaces to each rule application," so that lookback and lookahead are not necessary. This, in turn, simplifies the model to include *no* levels, and seeks to deepen explanation by moving one step closer to relocating within the independently motivated, universalized rules of Merge, Move/Attract, and Agree all GB principles, filters, and stipulated definitions of relations on representation.

2 DBP's general argument for cyclic Spell Out

This section presents important background material regarding the feature system and the nature of feature valuation assumed in DBP (subsection 2.1). Subsection 2.2 presents the general DBP argument seeking to explain the existence of cyclic Spell Out, and then reveals a possible problem with it.[2]

2.1 *Background: The DBP feature system*

Fundamental to the Minimalist Program is the assumption that there are lexical items, each consisting of three types of features: semantic, phonological, and formal. The formal features, which play a central role in the analyses of DBP, include Case, EPP, the phi-features of the functional categories T and v, and the phi-features of the lexical categories N and V. Some of these formal features are LF-interpretable – for example, the phi-features of N; while others are not – for example, the Case and EPP features. In earlier stages of the development of "feature checking" in Minimalism, interpretability is assumed

to be an inherent property of features. Thus, in Chomsky (1995, chapter 4) "Categories and Transformations" (CT) and in Chomsky (2000) "Minimalist Inquiries" (MI), Case is not, but the phi-features of N are, LF-interpretable. In DBP, on the other hand, the notions interpretable and uninterpretable play no direct role in the statement of syntactic operations; they are replaced, in a manner that we make clear below (section 3), with "valued" and "unvalued."

The following generalizations regarding interpretability and feature value, which figure crucially in the general argument explaining cyclic Spell Out, hold:

(1) a. It is (naturally) assumed that unvalued lexical features are illegible (and hence uninterpretable) at both PF and LF, and thus any expression containing such features will fail to converge.

b. Valuation, however, is a necessary but not sufficient condition for LF convergence. The Case feature of a DP/N, for example, may be valued by the operation Agree, but a valued Case feature is by hypothesis still not interpretable at LF, rather it is interpretable only at PF – this is the nature of structural case. In fact, all unvalued features that can be valued by a syntactic operation (i.e. the Case feature, the phi-features of T and *v*, and the EPP feature) are ultimately unuseable at LF; either they are by hypothesis LF-uninterpretable (as is true for the Case feature and the EPP feature) or else they are LF-interpretable and so can appear in a convergent LF representation but yield "gibberish." Such is arguably true with the valued phi-features of T; the phi-features themselves (e.g., the feature [singular-number]) are LF-interpretable, but the complex consisting of T and its valued phi-features is LF-anomalous, hence "convergent gibberish." Note, however, that some valued features are PF-interpretable; for example, Case and the phi-features of T.[3]

According to the DBP analysis, some formal features enter a derivation with a value (e.g. the phi-features of N), and all such features are interpretable at LF. Some formal features enter a derivation without a value (e.g. Case, EPP, and the phi-features of T and *v*).[4] The syntactic operations Agree and Move (may) assign values (under the appropriate circumstances) to unvalued features. The now-valued formal features are never useable to LF (they yield either LF nonconvergence or gibberish), but they may be perfectly interpretable to PF (depending on the feature).

Why does DBP seem to have both the valued vs. unvalued distinction and the interpretable vs. uninterpretable distinction? We answer the question in greater detail in section 3, but the answer relevant to immediate concerns is this: Spell Out can have access to valued and unvalued without having to know anything about whether the valued or unvalued feature is LF-interpretable. This is a desirable result since Spell Out (arguably) has no direct access to the LF interface and so in fact cannot know if a feature is LF-interpretable. Importantly, lookahead to the interface levels in DPB is, through the notion

valuation, thereby avoided. Spell Out can make what turn out to be crucial derivation-internal decisions, namely to Spell Out all and only features that *will* or *would* be uninterpretable[5] at the later LF level, but it can do so just by seeing whether or not the feature *now*, derivation-internally, has a value.

2.2 *The general DBP argument that Spell Out must be strongly cyclic*

Notice, first, that from (1a) above, it follows that unvalued features must be valued (if an expression initially containing these unvalued features is ever to converge). Indeed, unvalued features drive the syntactic operations Move and Agree, both of which result in valuation (thus, the valuation operations conform to "Last Resort"). An unvalued feature of X renders X "active," and only active elements may participate in syntactic operations. Move and Agree are thus purposeful; they apply for a reason, and the reason is to value unvalued formal features; syntactic operations have the "goal" of generating objects that are legible to the interfaces.

From (1b), it follows that those formal features that are valued by a syntactic operation must be "stripped away" from the narrow syntax. These syntactically valued features must not make their way to LF; if they do, then the expression containing them will either fail to converge (for Case and/or EPP features) or else will converge as gibberish (as is arguably true for an LF representation containing valued phi-features of T). Since Spell Out is the operation that removes features from the narrow syntax, it thus follows from DBP that *at least one* application of Spell Out is necessary. If there were no Spell Out, then virtually all expressions would fail to converge (or else would converge as gibberish).

DBP makes the further argument that Spell Out is cyclic (i.e. it occurs more than once in the course of a single derivation), and that this need not be stipulated, but rather, by the general argument it "follows naturally from examination of feature interpretability . . ." DBP thus has the important goal of not stipulating but rather of explaining the existence of cyclic Spell Out. With the important background and detail on feature valuation and interpretability in place, let us now examine this argument, turning thereafter (section 3) to a possible problem with it.

First of all, according to DBP, Spell Out removes (what amount to) LF-uninterpretable features from a syntactic object K. Thus, after Agree applies between T and N in (2), valuing the Case feature of N and the phi-features of T,

(2) T be a man outside

Spell Out removes, among other things, the LF-uninterpretable Case feature of *man*, and the phi-features of T, sending the residue to the narrow syntax *en route* to LF. Spell Out, then, must "know" what to Spell Out. The question is: how does it know? How does Spell Out know which features are

LF-(un)interpretable given that Spell Out is a non-"semantic" operation applying derivation-internally before the LF interface is reached?

DBP answers:

(3) The natural principle is that the uninterpretable features, and only these, *enter the derivation without values, and are distinguished from interpretable features by virtue of this property.* (DBP, p. 5; editors' emphasis)

Note, then, that DBP effectively replaces the notions interpretable/uninterpretable with valued/unvalued, questions concerning which we will consider in some detail below. Notice also that the distinction between interpretable and uninterpretable made in (3) rests on feature status upon entering the derivation, but Spell Out applies after (potentially long after) entry, as we will discuss shortly.

The question that now arises is: "When does Spell Out apply in the course of a derivation?" Crucially, to actually spell out before valuation is problematic. As DBP notes, this is "too early" since unvalued features are in fact PF-uninterpretable (as well as LF-uninterpretable) and thus spelling out an expression containing such unvalued features will fail to converge.[6] Thus, Spell Out cannot apply before valuation.

The other option is for Spell Out to apply after valuation. However, DBP claims that this too is problematic. Spelling out after valuation is "too late" since the distinction between LF-interpretable and LF-uninterpretable is "lost" immediately after valuation:

(4) The operation Spell Out removes LF-uninterpretable material from the syntactic object K and transfers K to the phonological component. It must therefore be able to determine which syntactic features are uninterpretable, hence to be removed. Prior to application of Agree, these are distinguished from interpretable features by lack of specification of value. *After application of Agree, the distinction is lost.* (DBP, p. 5; editors' emphasis)

The basic point here is clear enough. DBP assumes that Spell Out cannot know in a direct way what is and is not semantically interpretable. Spell Out is not itself "semantic" and has no access to the LF interface. It cannot look at a derivation-internal representation, consult the LF bare output conditions, determine what is interpretable at LF, and then go back inside the derivation to Spell Out the LF-uninterpretable features present in the current representation. But the right result can be obtained in an indirect way, and in conformity with Inclusiveness, by assuming featural underspecification. Spell Out can see the valued vs. unvalued distinction, and since valued/unvalued is linked to LF-interpretable/uninterpretable in the right way, Spell Out can "know" what to strip away to the phonology. Spell Out *does* have access to valued and unvalued. However, once a feature is valued, then, for Spell Out, the distinction between LF-interpretable and LF-uninterpretable is lost.

To sum up, DBP argues that Spell Out before valuation is too early, but after valuation is too late. But this is precisely the crux of DBP's general argument:

that it follows from basic assumptions that Spell Out must apply cyclically. Recall that this is argued to "follow naturally;" and hence cyclic Spell Out, the central architectural innovation in the MI/DBP abandonment of the Y-model, is not stipulated. The crucial passage is given below in its entirety:

(5) The operation Spell Out removes LF-uninterpretable material from the syntactic object K and transfers K to the phonological component. It must therefore be able to determine which syntactic features are uninterpretable, hence to be removed. Prior to application of Agree, these are distinguished from interpretable features by lack of specification of value. *AFTER application of Agree, the distinction is lost.* To operate without reconstructing the derivation, *Spell Out must therefore apply SHORTLY AFTER the uninterpretable features have been assigned values* (if they have not been assigned values at this point, the derivation will crash, with uninterpretable features at the interface). Spell Out must be strongly cyclic (as assumed in MI). The conclusion, which follows naturally from examination of feature interpretability, has other desirable consequences. (DBP, p. 5; editors' emphasis)

The idea seems to be that features are being valued iteratively (since the derivation is assumed to proceed "bottom-up," Chomsky 1993), and once they are valued, Spell Out cannot operate in the requisite fashion. Thus, Spell Out must also operate iteratively.

3 The problem with the general argument that Spell Out must be strongly cyclic

The argument, as stated in (5), that cyclic Spell Out "follows naturally from examination of feature interpretability" seems to us to contain a logical contradiction. Simply put: if it is true that "after application of Agree [i.e. value-assignment, SDE/TDS] the distinction is lost," then it does not follow that "Spell Out must therefore apply *shortly* after the uninterpretable features have been assigned values." If after valuation is too late, then shortly after is too late as well. Spelling out shortly after value-assignment provides no solution to the central problem at hand, since as soon as a feature is valued, Spell Out will not have the information required to operate. Yet it is this very on-line derivational valuation process that constitutes the explanatory argument that Spell Out must apply cyclically, since the distinction upon which Spell Out operates is lost quickly, as the derivation proceeds. To illustrate the problem, consider again (2), repeated here:

(2) T be a man outside.

Suppose that Agree has applied, valuing (among other things) the phi-features of T. Suppose that Spell Out applies immediately after Agree. Looking just at

the object (i.e. the single representation) in (2), what Spell Out now sees are valued features. And after valuation (shortly after or otherwise), the phi-features of T have the same status as the phi-features of N; they are valued features, and, therefore, Spell Out will not be able to distinguish them; i.e. it will not know that N's phi-features should not be spelled out, but that T's should be. Indeed, after valuation (even shortly after), it is not obvious that Spell Out could operate at all since, "without reconstructing the derivation" (see (5)); i.e. by looking just at the object in (2), Spell Out has no means of knowing what to actually Spell Out. More specifically, without derivational lookback, it has no means of knowing which of the valued features that it now sees were, at an earlier derivational point, unvalued. And without derivational lookahead to the interfaces, Spell Out cannot know what is or isn't interpretable at the not-yet-reached interfaces LF and PF.

Notice that the logical problem raised above could potentially be resolved if there *were* some form of derivational lookback.[7] Spell Out could be designed to Spell Out only those features that were valued in the course of the derivation; and to determine this, Spell Out would need access to the derivation itself (and not just the current syntactic representation). In section 5, we consider this potential resolution to the problem in some detail. First, however, we will examine more closely DBP's introduction of the notion "valuation" and further clarify the problematic nature of the general argument for cyclic Spell Out, and we will then, in section 4, present a possible solution to this problem, one which appeals to fundamental properties of the derivation itself.

Cyclic Spell Out, in the general sense, means that the transference of features to PF occurs not just once but more than once during the derivation; and furthermore that it occurs bottom-up in the course of the derivation. Thus, rather than the standard Y-model as in (6)

(6) Lexicon → syntactic operations → Spell Out → PF
(6) Lexicon → syntactic operations → LF

we have (something like) the model in (7)

(7) Lexicon → syntactic operations → Spell Out → PF, Spell Out → PF, . . . etc.
(7) Lexicon → syntactic operations ----------> LF

DBP abandons the long-standing Y-model in favor of a model more like that in Epstein et al. (1998), Epstein and Seely (1999, forthcoming) and Uriagereka (1999), to the extent that cyclic Spell Out is hypothesized.

Recall now the key step in the DBP argument seeking to explain such cyclic Spell Out:

> (8) Prior to application of Agree, these [i.e. LF-uninterpretable features] are distinguished from interpretable features by lack of specification of value. After application of Agree, the distinction is lost. To operate without reconstructing the derivation, Spell Out must therefore apply shortly after the uninterpretable features have been assigned values. . . .

One question that we raised earlier, and will examine now in more detail, is this: Why is a distinction made between an interpretable/uninterpretable feature vs. a valued/unvalued feature? Why does DBP introduce the concept "valued/unvalued;" why not maintain the analyses from CT and MI postulating just and only "interpretable/uninterpretable"?

The answer, we believe, is twofold. First, the introduction of the notion "valued/unvalued" can be seen perhaps as a response to a problem raised by Epstein et al. (1998), concerning the entire Minimalist theory of transformational rule application. Specifically, the problem is how a syntactic operation can be triggered by the uninterpretability of a feature at the LF interface *before* the LF interface is reached. Adapting the Epstein et al. observation to Spell Out in DBP, we can ask: How can Spell Out know that a feature is not interpretable to the LF interface when Spell Out is not at the LF interface and furthermore has no direct access to the LF interface? The valued/unvalued distinction, on the surface at least, solves this problem. Spell Out in fact does not know which features are LF-(un)interpretable. All it sees (focusing on formal features for present concerns) are features that have or do not have a value, and that is sufficient if, as we saw above, "all unvalued features are LF-uninterpretable/unuseable."

And there is an additional possible reason for DBP's introduction of "valued/unvalued." If we have just interpretable and uninterpretable, as in CT and MI, then we need feature deletion followed, in some cases, by feature erasure, and, as DBP itself notes, this is undesirable, creating an architectural paradox.[8] In the CT and MI frameworks, there are cases where a feature is deleted but still accessible to the syntax. In some cases, this deleted feature is then erased, which means that it is eliminated entirely and hence absent at the LF interface. The valued/unvalued distinction of DBP, along with not applying Spell Out immediately,[9] gets the effect of the CT/MI deletion-erasure mechanisms in, perhaps, a more elegant way. In DBP, certain formal features enter the derivation without a value. A syntactic operation, like Agree, results in the valuation of the feature. The feature is not yet spelled out, and hence the valued feature can, and crucially does, participate in later syntactic operations. This is just what happens in DBP in example (9), for instance:

(9) [C [$_{TP}$ T seem [EXPL to have been [caught several fish]]]]
(= DBP example (18a), p. 17)

) There is no single representation upon which Spell Out can operate in an empirically adequate manner.[11]

us, Spell Out must examine multiple representations, i.e.:

) Spell Out must itself be a derivation-sensitive operation, not a purely representational one.

nimally, it examines two, and no more than two, representations. Which ɔ? The null hypothesis is that it can examine only the input to a given nsformational rule application (in which a feature is unvalued) and the put of that same transformational rule application (in which the feature is *v* valued). Crucially, this requires no new mechanisms (such as phases and ıse level lookback, to be discussed in section 5). Universal grammar already orporates independently motivated mechanisms each with a "before" and "after," namely transformational rules consisting precisely of a structural cription (a "before"), specifying the input to the transformational rule ɔlication, and a structural change (the "after"), specifying the output of the nsformational rule application. Thus, we propose that Spell Out operates on and only those formal features that appear without a value in the input to ıle being applied, but appear in the output of that rule application with "previously" unvalued feature now valued. If no single representation is ficient, the minimal "phase" is optimally a single transformational rule ɔlication[12] – requiring no stipulated privileged phasal categories such as *v*P CP, and requiring no new mechanisms at all.

Consider, for example, (2), repeated below, focusing for illustration on just phi-features of T:

T be a man outside.

ler the Probe–Goal analysis, the structural description of the operation ee applying to T and *man* includes the unvalued phi-features of T. The ıctural change of this operation includes the now valued phi-features of f Spell Out can see both the input to and output of Agree, it can see the ɔcess" whereby an unvalued feature became valued, and then can spell just these features, as desired. Notice again that the input to and output single transformational rule application seems to be the minimal set of cts that Spell Out can inspect. Thus, our central proposal is that Spell Out rates "inside" transformational rule application. This comports with the ɔry proposed in Epstein (1994, 1999), Epstein et al. (1998), and Epstein and y (1999, forthcoming), which seeks to explain syntactic relations, syntactic rs, and, more generally, macrostructure tree properties, as expressed by Principles, by appeal to independently motivated properties of local trans-

rule application in terms of optimal solutions to the bare output conditions imposed by the interfaces. That is to say, every rule application must value features, which, if unvalued, induce crash at the interface. What is the most efficient way to determine that features have been valued by each transformational rule application? Suppose every transformational output is evaluated by PF (and LF), whose very legibility conditions each transformational rule application seeks to satisfy as best as is (locally) possible. Notice that, as a result, we eliminate not just D-structure and S-structure, as in Chomsky (1993); but PF and LF too are eliminated as levels of representation in the standard sense (as in Epstein et al. 1998, and Chomsky 2001b, as will be discussed below). Applying Spell Out in this manner seems conceptually optimal in that it requires no account of why some categories get spelled out cyclically, i.e. constitute phases (a central concern of DBP) while others do not. In short, spelling out each transformational output not only readily overcomes the precise DBP "before and after" problem we reveal, but is also "natural" in requiring no new mechanisms beyond those already postulated in a derivational system. This model incorporates ordered transformational rule application but eliminates levels of representation in the standard sense, instead placing them or their effects "inside" transformational rule application. Moreover, lookahead is rendered unnecessary, precisely because the interfaces are reached by each transformational rule application.

4.1 *A single representation?*

Recall that section 4 began with the assertion that there is no single representation upon which Spell Out can operate in an empirically adequate manner. However, it might be counter-argued that there is a single representation, i.e. a single syntactic object, upon which Spell Out could operate correctly. Consider (12):

(12) John was arrested John

Suppose the occurrence of *John* in Spec-TP contains a valued case feature, whereas the occurrence in direct object position bears unvalued case. Then, as long as Spell Out knows that this is the "same feature," Spell Out could see in this single representation both the valued and unvalued Case feature, and hence know to spell it out. No derivational lookback would seem to be required under this view. We have two points to make regarding this roughly "chain-based" approach.

First, it has been argued that there are no traces/copies, from which it follows that a representation like (12) would simply not exist. It has been noted, for instance, that copies/traces are encodings of previous derivational stages. And, as Epstein et al. (1998) argue, such encodings are necessitated in the rule-free, derivation-free GB theory, precisely because we need to know where a category has been in the course of the derivation but are not allowed to refer to the derivation in this rule-free approach. Within the Minimalist framework, in which iterative transformational rule application (i.e. the derivation) is reified, the possibility of the elimination of traces/copies arises; indeed, the elimination of traces/copies is just what is suggested in Epstein et al. (1998).[13] On different grounds, Epstein and Seely (1999, forthcoming) argue against A-traces in examining successive cyclic A-movement in English; and Lasnik (1999) argues against A-traces based on nonreconstruction phenomena with A-movement. The central point is that if these arguments are on the right track, and there are no traces/copies, then a representation like (12) is not possible.

Second, suppose instead that there is a copy/trace, as in (12). The question that now emerges is: can the copy/trace have an unvalued Case feature, after the Case feature of the mover has been valued? We believe the answer is "no." Recall that under copy theory as identity there is in (12) only one DP *John*. There are, however, two occurrences of this *single* DP in the representation. Since there is but one DP in the representation and since that one DP is a bundle of features, each feature having one and only one value, it cannot be that one occurrence of a DP has a valued feature while the identical DP (in another position) has that same feature unvalued.[14] So it follows, even with traces/copies, that a representation like (12) is not possible, i.e. where the Case feature of the mover *John* is valued but the Case feature of the copy/trace is unvalued. This, in turns, supports our conclusion that Spell Out can't operate on a single representation.

5 Phasal Spell Out

We have just proposed that:

(13) a. No single representation will suffice; and

b. appeal to the derivation is therefore necessary; and, optimally, the only aspect of the derivation that needs to be inspected is inside single transformational rule applications, minimizing lookback, and eliminating lookahead.

This is tantamount to the claim that each transformational rule application constitutes a "phase," which we believe to be the null hypothesis. Like Epstein et al. (1998), Epstein and Seely (1999, forthcoming), and Uriagereka (1999), Chomsky (2000, 2001a, and 2001b) abandons the Y-model's single split to PF and LF in which interface interpretation applies only after all transformational rules have applied. For example, Chomsky (2001b): 4, "Beyond explanatory adequacy," henceforth BEA most recently characterizes his cyclic Spell Out approach as level-free, writing that

(14) in this conception there is no LF: rather, the computation maps LA to <PHON, SEM> piece-by-piece cyclically. There are, therefore, no LF

properties and no interpretation of LF, strictly speaking, though sigma and phi interpret units that are part of something like LF in a non-cyclic conception.

Call the relevant units "phases." It remains to determine what the phrases are, and exactly how the operations work.

The difference between our proposal, which we believe to be the null hypothesis, and Chomsky's, concerns the size of the phase. For Chomsky, the phases are the specific categories *v*P and CP. In what follows we will discuss three ways in which this diverges from our null hypothesis, engendering what we take to be nonoptimal solutions to the output conditions. Furthermore, we note possible empirical problems with *v*P and CP as phases, in addition to problems confronting the quest to explain why CP and *v*P in particular are the only phases.

5.1 *Why* v*P and CP?*

The specification of particular phase level categories runs the risk of stipulation. As discussed at length in Epstein and Seely (1999, forthcoming), the specification of *v*P and CP as phases is potentially problematic. Chomsky (MI, DBP, BEA) attempts to motivate *v*P and CP as phases on the grounds that they are relatively independent at the interface. But, leaving aside potential unclarities concerning the notion "relatively independent," an important additional question is: How can we know that they are relatively independent *at the interface* if Spell Out applies *before* the interface is reached, and without access to interface properties? It is a potential architectural paradox to hypothesize that *v*P and CP are spelled out *cyclically, internal to the narrow syntax* by virture of them having the property of being, "later," relatively independent *at the interface*.

The other notion invoked to explain why *v*P and CP are phases is the concept "propositional." However, it seems to us that there are *v*Ps that are non-propositional; for instance, *who bought what* and *everyone bought something*. Prior to movement these *v*Ps exhibit vacuous quantification; subsequent to movement they are open "sentences" which, internal to the *v*P, contain free variables. The same is true for the embedded CP in, for example, *who do you think that John likes?* Further, in the other direction, there are propositions that are arguably neither *v*Ps nor CPs; for example, the small clause in *I consider [John smart]*. Another question is: Why should a propositional element, where "propositional" is a semantic notion, be spelled out to PF; why should PF care about the propositional content of what is spelled out?

A third specification of phase is a category within which all theta-roles borne by the head are discharged. But this seems to allow raising TPs, passive, and unaccusative *v*Ps to be phases since all relevant theta-roles are in fact discharged, but this does not seem to be the desired result. Thus, the concern here is that *v*P and CP are stipulated as phases (moreover, see Epstein and Seely 1999, forthcoming) for extensive discussion of empirical problems with this stipulation regarding phases, involving Merge-over-Move and short passives).

Yet another possible problem is that these phases constitute special, privileged points in the derivation, which seem to us somewhat like the reintroduction of levels which MI, CT, BEA, Epstein et al. (1998), and Uriagereka (1999) all seek to eliminate.

5.2 *Global lookback?*

As we pointed out in section 4, one way to resolve the logical problem with Spell Out that we revealed is to assume derivational lookback. DBP does assume such lookback, but attempts to limit it to the phase level. Thus, DBP (p. 12), states that:

(15) The valued uninterpretable features can be detected with only *limited inspection of the derivation* if earlier stages of the cycle can be "forgotten" – in phase terms, if earlier phases need not be inspected. (editors' emphasis)

The important point here is that "inspection of the derivation" *is* required. Two questions emerge at once: why is it required? And, what exactly is the algorithm for inspection of the derivation?

Consider, for illustration, (16):

(16) I wonder what John will buy.

First-merge yields:

(17) buy what

where *what* has an unvalued Case feature. The following further Merges then occur:

(18) a. *v* + [buy what]
b. John + [*v* [buy what]]

Next, *what* shifts to the outer spec of *v*P. At this point in the derivation (i.e. in this syntactic object), the Case feature of *what* is valued by phi complete *v*:

(19) what + [John [*v* [buy what-copy]]]

Under copy theory, it is assumed that the Case feature on the *identical* copy is also valued, otherwise movement would never yield convergence. Valuing the trace/copy's feature is necessitated under the assumption that the copy is featurally identical with the mover (although these occurrences are positionally distinct; in other words, if, in the representation (19), *what* and *what*-copy are the same thing, then there is only one Case feature to value; and if *it* is valued, it is valued (for extensive discussion of this issue, see Epstein and Seely 1999, forthcoming; see also Nunes 1999 for important related discussion). Indeed, if

the trace/copy's Case feature (somehow) remained unvalued, then movement for Case checking would fail to result in satisfaction of the bare output conditions, undermining the entire theory of movement/transformations. So, it is a general feature of the theory of movement, and its relation to the satisfaction of bare output conditions, that the trace/copy must be valued.

Next, *will* (T) is merged in; and thereafter *John* is attracted to spec of T:

(20) John will what John-copy *v* [buy what-copy]

C now merges in, and C attracts the closest occurrence of *what* (occupying the outer spec of *v*P):

(21) what John will what-copy John-copy *v* [buy what-copy]

Crucially, this is the position in which *what* is to be spelled out. However, as we saw in section 4, the current representation (i.e. (21)) provides insufficient information for Spell Out to operate correctly: the features of *what* and all its copies are valued and once valuation applies, it is too late for Spell Out to operate (without reconstructing the derivation). Again, as we noted in section 4, it is not clear why Spell Out would even try to operate at all here, since there is nothing offensive to spell out (or problematic) in the current representation. The issue of why Spell Out operates at all is empirically crucial, and pervasive. But suppose, for the sake of argument, that Spell Out has a "problem" of some sort, and knows that it must spell out something now, but all the features are valued so it can't. What is Spell Out to do? This is, we believe, the DBP argument for derivational lookback (i.e. "limited inspection of the derivation," p. 12). Spell Out detects the valued features on *what* and its occurrences and (for some reason) begins to engage in derivational lookback, seeking to find an earlier point in the derivation in which the Case feature of *what* was unvalued. If that unvalued feature can be found (in representations (17), (18), or (19)), Spell Out can "see" that the feature changed from unvalued to valued, and that information suffices to instruct Spell Out to spell out the Case feature of *what*. So, this seems to be why DBP assumes that derivational lookback is required.

The remaining question is: how does derivational lookback operate exactly? What is the algorithm? We're not sure. But the important point is, in the above derivation, when we constructed (21) (i.e. the final stage), Spell Out has to find a representation (i.e. a derivational stage) at which *what* still has an unvalued case feature; i.e. it has to find either (17) or (18), in each of which *what*'s case feature *is* unvalued. Moreover, Spell Out must have some means of determining that the valued case feature on *what* in the representation (21) is "the same" Case feature as the unvalued one in the syntactic object that was generated at an earlier stage in the derivation. Why the search is initiated, and how this search and determination of feature "identity" is to be formalized, and the extent to which it can be made computationally efficient, is beyond the scope of this chapter; but these constitute crucial questions for the DBP analysis, we think. However, since wh-movement is itself unbounded, e.g.:

(22) What did Fred say that Bill said that Fred said that Bill said . . . that John will buy?

Notice that at best, it would seem that lookback would have to be indefinitely far. So, it is not clear that "inspection of the derivation" is indeed "limited" under the derivation by phase approach.

In fact, the situation might be worse. Consider just one iteration of Spec-CP to Spec-CP movement:

(23) What did Bill say that John will buy?

Again, for *what* to be spelled out at this point, the current representation (i.e. (23)), is insufficient (since the Case feature on each occurrence of *what* is valued). Thus, for some reason, Spell Out is compelled to look back to find a previous stage of the derivation that contains the corresponding Case feature on *what* unvalued. The previous derivational stages (syntactic objects) containing this unvalued form of the Case feature of *what*, however, are all internal to the most deeply embedded *v*P since, recall, it is in the outer Spec of this *v*P (early in the derivation) that the Case feature of *what* becomes valued. Thus, for derivational lookback to find a syntactic object with an unvalued case feature on *what*, it must look back to the derivational stage (*v*P; or points generated before that). Notice, however, that this is in fact impossible under DBP's (see also MI) Phase Impenetrability Condition (henceforth PIC):

(24) The domain of H is not accessible to operations outside HP [HP a strong phase]; only H and its *edge* are accessible to such operations.

It is assumed in DBP that "optimally all operations are subject to the same conditions," and thus that Spell Out, which is an operation, is subject to the PIC. Thus, when *what* reaches the matrix Spec-CP and is to be spelled out, the most deeply embedded *v*P has already been spelled out, and thus is, under the PIC, the principle at the heart of the derivation by phase approach, inaccessible.[15] In a nutshell, before valuation is too early, after is too late, so you have to see "both," but, under the PIC, you can't. Overall, this represents a potentially serious problem with interphasal valuation.

Intraphasal valuation may also be problematic in a different sense. Consider, for instance, the standard analysis of unbounded successive cyclic A-movement:

(25) [$_A$ John seems to be certain to seem to be certain . . . to be arrested]

In this case, Spell Out would, at the point marked A, be operating within a single phase (none of the intermediate TPs are phases). Now, suppose that each movement of *John* (assuming that *John* has moved successive cyclically) is valuing an EPP feature on the intermediate Ts, as has been proposed (in, for example, CT; see also BEA). In the single CP, all the Ts, including the most deeply embedded T, are valued. This single representation is thus insufficient

for Spell Out to operate on. Again, for some empirically crucial reason, Spell Out starts to look back. It is going to have to find a representation at which the most deeply embedded T was unvalued. But, its valuation happened "a long time ago;" in fact, given that A-movement is unbounded it can happen indefinitely "far back." The point we are getting at is that depending on the actual algorithm for looking back, and the order in which previous stages are to be inspected, limiting lookback to within-phase-lookback, under which earlier stages can be "forgotten," isn't necessarily computationally efficient. The lookback needed, even though limited to a single phase, could still be unbounded since there is recursion within phases.[16]

Pursuing the matter further, notice that for the Case feature of *John* in (25), the most efficient way to locate the unvalued Case feature quickly is to

(26) Inspect the immediately preceding stage of the derivation.

By looking back just this far, Spell Out can see that *John*'s Case feature was unvalued, and hence is now to be spelled out. However, if this is the general algorithm, then to find the unvalued EPP feature on the most deeply embedded T, can, in certain cases, require an indefinite number of steps.

If EPP is not feature checking (see, for example, Chomsky 2000 and Lasnik 2001) then the illustration above is invalid. However, if, as Lasnik (and others) argue, EPP is in fact "configurational," then it seems to us to undermine the entire Minimalist theory of movement based on feature interpretability at the interfaces. More generally, "configurational" requirements represent a retreat to the stipulation of molecular tree properties and the failure to deduce molecular tree properties from atomic structure of lexical features, and their mode of attraction and repulsion. It amounts to the reincorporation of phrase structure rules and/or principles of GB, precisely the type of principle that gave rise to the quest for Minimalist explanation, which prohibits such postulates. Stating that "there must be a category here" if true, constitutes nothing more than a description.

5.3 *Simultaneity?*

It is perhaps in response to the "before" and "after" valuation problem discussed here[17] that BEA seeks to eliminate "time" within the phase. The basic idea seems to be that if before is too early and after is too late, then eliminate "time" by having all feature valuation transformations apply at once within the phase. As stated in BEA (p. 22):

(27) Again, it is "as if" all operations are applying simultaneously at the phase level . . .

We should begin by noting that it is not entirely clear to us what precisely is meant by simultaneous rule application here, nor why it is "as if" operations apply simultaneously (as opposed to actually applying simultaneously). Thus,

the discussion below is tentative. But, for purposes of discussion, we (naturally) assume that

(28) If the operations OP1, OP2 are applied simultaneously to a given syntactic object X, then X must meet the structural description of both OP1 and OP2. Furthermore, OP1 and OP2 apply to X without any ordering (or "time") between their application; and thus with no intermediate representation ever generated by the application of one OP, but not yet the other.

We now have a number of points regarding simultaneous rule application.

First, simultaneity does not seem compatible with Earliness, yet Earliness is advocated in DBP and BEA. In fact, DBP (p. 15), states that:

(29) As discussed in MI and sources cited, there is mounting evidence that the design of FL reduces computational complexity . . . One indication that it may be true is that principles that introduce computational complexity have repeatedly been shown to be empirically false. One such principle, Procrastinate, is not even formulable if the overt/covert distinction collapses: purely "covert" Agree is just part of the single narrow-syntactic cycle. With the motivation for Procrastinate gone, considerations of efficient computation would lead us to expect something like the opposite: perform computations as quickly as possible, the Earliness Principle of Pesetsky (1989).

We adopt the position in DBP that delay is computationally inefficient. But simultaneous rule application represents a type of delay. Rather than applying an operation O1 at point P1 of a derivation, the syntax "waits" until a phase level is reached, at which point a set of operations (including O1), whose structural descriptions are all met, apply simultaneously.

A related point is that simultaneous rule application seems nonderivational. For example, a theory seeking to explain syntactic relations (as in Epstein 1994, 1999; Epstein et al. 1998) in terms of the step-by-step derivation cannot be maintained since the iterative steps of the derivation are necessarily eliminated under simultaneity. Thus, the syntactic relations that Chomsky characterizes as "free" run the risk of becoming costly. More generally, if simultaneous rule application is allowed, the question arises as to when it is allowed, and why. Unrestricted simultaneous rule application amounts to the elimination of ("Minimalist") derivation altogether, and a return to a GB-type representational theory.

Second, assuming that Merge (for some reason) is not subject to simultaneity, (rather, only feature valuation operations are), we would seem to have two cycles: Merge on the one hand and valuation on the other. Our goal is to have just a single cycle.[18]

Another problem concerning simultaneous rule application within the phase arises if, as in MI and DBP, there exists empirically motivated explicit rule ordering within a single phase. Such within-phase ordering in fact plays a

crucial role in DBP and MI, in properly deriving the following case, which occupies a central role under the elimination of the Case Filter at S-structure:

(30) There seems there-copy to be a man outside.

In the bottom-up step-by-step derivation of DBP and MI, the following point is reached:

(31) [to be a man outside]

Under the Merge-over-Move analysis, since *there* is present in the Numeration, and visible to Merge, it must be merged *at this point*, yielding:

(32) [there to be a man outside]

Subsequent step-by-step rule applications apply, still all within the same single phase, since there is no *v*P nor CP yet generated, producing:

(33) T seems [there to be a man outside]

Now crucially, the idea within MI and DBP is that the matrix phi-complete T searches a Matching Goal. The *previously inserted "there"* is now the closest matching Goal, and therefore *there* is attracted to matrix spec TP, but *a* man is not, yielding the correct predictions:

(34) a. [CP [TP there seems [there-copy to be a man outside]
b. *[CP [TP A man seems [there to be a man-copy outside]

Recall, this all occurs within a single phase, the matrix CP. The success of the MI and DBP analyses thus rests on the explicit order of application of rules within a single phase.

(35) Within the matrix CP phase (the only phase present), *there* had to be introduced before matrix T undertook Attraction, so that when matrix T attracted, *there* was already in place, and closer to *there* than is *a man*.

The prevention of such superraising thus rests on rule ordering within a single phase. Under simultaneous rule application within the phase, this central account is lost. We thus tentatively reject simultaneity as the solution to the "before and after" problem confronting Spell Out.

6 Summary

DBP (and also BEA) seeks to *explain*, not stipulate, why there is multiple, cyclic Spell Out, arguing that it "follows naturally from examination of feature

interpretability" (DBP, p. 5). Since features are valued iteratively in the course of the derivation, and since, once valued, Spell Out cannot properly operate, it is said to follow that Spell Out must "shadow" the iterative valuation process and thus must itself apply iteratively (i.e. "shortly after" each feature valuation). Feature valuation itself is invoked, arguably, because the derivation-internal, nonsemantic operation Spell Out has no direct access to the LF interface and so cannot know if a feature must be spelled out to PF *now* because it *will* be uninterpretable at LF later. Crucially, lookahead to the interfaces is thereby avoided.

DBP also seems to seek to eliminate "lookback:"

(36) *After* application of Agree, the distinction is lost. *To operate without reconstructing the derivation*, Spell Out must therefore apply *shortly after* the uninterpretable features have been assigned values.

So, suppose there is no lookahead and no lookback. Suppose further that after application of Agree is too late, but before is too early. Then, we have argued, Spelling-out *shortly after* value assignment provides no solution to the entirely empirical problem, since as soon as a feature is valued, the distinction Spell Out operates upon is lost. But recall that it is this very on-line iterative derivational valuation process that constitutes the basis of the DBP (and BEA) attempt to explain why Spell Out must apply cyclically, this cyclic application constituting the crucial departure from the Y-model.

We thus conclude that this attempt to explain cyclicity is problematic, and, importantly, it leaves unclear when Spell Out can possibly operate at all.

Our remaining arguments can then be summarized as follows:

1. We have sought to maintain Chomsky's explanation for cyclicity by applying Spell Out neither before nor after valuation, since each is empirically problematic, but instead by applying Spell Out "inside" the very valuation (transformational) operation itself. This is tantamount to construing each single rule application as a "phase."
2. This is consonant with the derivational approach seeking to deduce principles, filters, and syntactic relations from the properties of simple, local, independently motivated transformational rule application. If correct, the study at hand is interesting since Spell Out, being neither a principle, nor a filter, nor a relation, is similarly "driven inside the rule."
3. Simultaneous rule application (as proposed in BEA) is rejected as the solution to the "before and after" problem. We note that simultaneous rule application is nonderivational and engenders empirical problems while raising issues of which operations are and are not simultaneous, and why.

With Spell Out located "within" the rule:

4. Levels are eliminated. Instead, interpretation by the interfaces applies to each syntactic object generated by the computational system.

5. Reaching the interfaces at each stage comports best with "last resort;" if derivation-internal operations must each increase legitimacy at the interface (the entire theory of movement), this is best enforced (without lookahead or lookback) by having the interfaces immediately evaluate the operation itself.
6. Under our analysis, there is no *lookahead*. Spell Out does not need to consult the interfaces in order to determine what is to be spelled out.
7. Under our analysis, there is no *lookback* (beyond the input–output of a single rule application) to find an earlier stage when the now valued feature was unvalued. We noted empirical problems concerning lookback (to what can't be seen given the PIC) as well as cases of unbounded lookback. More centrally, it is not clear what ever induces lookback to initiate a search for unvalued features in the first place, since valued features in the current representation are inoffensive to Spell Out. Nor is the precise lookback procedure specified. Specifically, it is not clear in what order previous stages of derivation are examined, nor why. Perhaps the most important point about lookback is that if the theory allows it, why not maintain the Y-model, lacking cyclic Spell Out, perform all transformations, then lookback and Spell Out features accordingly? For myriad reasons, this seems to be the wrong approach.
8. There is no need for an independent account of which categories are phases (e.g. CP and *v*P) and which are not, and why. All syntactic objects are phases, and as in 4 above, each syntactic object undergoes interface (PF/LF) interpretation.

Finally, we have argued along the way that:

9. Formulating the EPP as a "configurational principle" is barred under Minimalist assumptions (see Epstein and Seely 1999, forthcoming) since it is a representational macro-tree description, demanding explanation in terms of lexical features and their mode of combination.

Our proposals constitute what we believe to be the specification of the null hypothesis regarding the organization of a "multiple splits" derivation-based model of UG lacking levels altogether. Our analyses are far from conclusive, but, we hope, contribute to the ongoing and (we think) exciting attempts to explicitly identify and seek explanatory maximization in formulating the theory of the biologically determined aspects of human knowledge of language, as begun and continued in Chomsky's routinely pioneering work.

Acknowledgments

For detailed and insightful written comments, we would like to thank Justin Fitzpatrick, Sam Gutmann, Mark Hale, and Hisa Kitahara. Thanks also to Joshua Epstein, Zeljko Boscovic, Beverley Goodman, and Esther Torrego for valuable discussion. Finally, we thank the audience at the 2001 Landelijke Onderzoekschool Taalwetenschap (LOT, Netherlands Graduate School of Linguistics) Summer School at the University of Utrecht.

Notes

1 This is a substantially revised version of a paper originally written in June 2000 (entitled "Cyclic spell out in 'derivation by phase'"). We are indebted to Noam Chomsky for extensive discussion of this earlier version of the paper, which was written and distributed before the appearance of Chomsky (2001b), "Beyond explanatory adequacy," aspects of which we address below.

2 In section 5, we will consider a similar argument, from Chomsky (2001b), for cyclic Spell Out.

3 Note further that some features, like the purported EPP feature, are apparently not PF-interpretable, and hence not interpreted at all. This raises the issue of the unsatisfying "uniqueness" of the EPP feature; unlike all other syntactically valued formal features, the EPP feature is never interpreted by LF nor by PF, which may suggest that it does not "exist," as is claimed in Epstein and Seely (1999). We will address the EPP in greater detail in sections 4 and 5 below.

4 Note that it has been argued recently (see, for example, Lasnik 2001) that EPP may not be a feature but rather a "configurational requirement." See section 5 below for discussion of, and skepticism regarding, this view.

5 By "uninterpretable" here we mean features that cause either LF crash or yield LF gibberish. Technically, it is not that Spell Out spells out all and only LF uninterpretables in the strict sense; rather it spells out all and only those features that are "unuseable" to LF in the sense specified in (1b) above.

6 For instance, if Spell Out transfers the unvalued Case feature of *man* in (2) to the phonology, the result is fine for LF (the Case feature is not present in the object that is seen at LF and thus cannot cause LF crash) but since that Case feature has not yet been valued, and unvalued features cause crash (see (1a) above), then spelling out to PF before valuation invariably induces PF crash.

7 We consider the issue of derivational lookback in DBP in detail in section 5. Derivational lookback, to the phase level, is (crucially) assumed in DBP. But the matter is not entirely straightforward; for example, DBP (p. 5) states that "To operate *without reconstructing the derivation*, Spell Out must therefore apply shorter after the uninterpretable features have been assigned values . . ." "Without reconstructing the derivation . . ." might suggest that there is no derivational lookback. We will argue that this quote is perhaps best interpreted as "Without reconstructing *the entire* derivation . . ." Thus, there is lookback, but it is limited (to the phase). See section 5 below for further discussion.

8 Thus, footnote 7 of DBP: "and without recourse to some such device as the distinction between erasure and deletion, invoked in Chomsky (1995) to deal with a paradox of EST-based systems with a single position for Spell Out: the 'overt' part of the narrow-syntactic computation eliminates uninterpretable features, but they have to remain until the stage of Spell Out of the full syntactic object, because of their phonetic reflexes."

9 As we will note in more detail in section 5, phasal Spell Out in DBP entails that Spell Out is in fact delayed until the phase (i.e. *v*P/CP) is reached. At this point, our focus is still on the general argument for the existence of cyclic Spell Out,

rather than the more specific argument that the actual units to be spelled out are *v*P and CP.

10 More precisely, they are "unuseable" (see note 5 above).

11 (10) holds under the (standard) view that if a feature F of a mover is valued, then the feature F of the occurrences of the mover are also valued, i.e. under copy theory as identity. We consider this matter in some detail in section 4.1 below.

12 Note that Chomsky (2001b: 14) argues that "It follows that phases should be as small as possible, to minimize memory for S-O [i.e. Spell Out]. . . ." See section 5 below for further discussion.

13 See particularly, Epstein et al. (1998: 153, 175).

14 See Nunes (1999) for important discussion of this, and related, issues; see also Epstein and Seely (1999, forthcoming) for extensive discussion of copy theory as identity.

15 It is also perhaps noteworthy that under the PIC, the residue of Spell Out, i.e. what remains after Spell Out, is itself not a constituent; e.g. [*v*P Spec [v' v . . .]] is not a syntactic category, i.e. a head and its Spec(s) do not form a constituent.

16 We would like to thank Sam Gutmann for very helpful discussion of this issue.

17 And in Epstein and Seely (2000).

18 Note that BEA (p. 4) argues that "the best case is that there is a single cycle only."

References

Chomsky, N. 1993. "A Minimalist Program for linguistic theory," in K. Hale and S. J. Keyser, eds., *The View from Building 20: Essays in Linguistics in Honor of Sylvain Bromberger*. Cambridge, MA: MIT Press. (Also appears as chapter 3 of Chomsky 1995.)

Chomsky, N. 1995. *The Minimalist Program*. Cambridge, MA: MIT Press.

Chomsky, N. 2000. "Minimalist inquiries: the framework," in R. Martin, D. Michaels, and J. Uriagereka, eds., *Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik*. Cambridge, MA: MIT Press.

Chomsky, N. 2001a. "Derivation by phase," in Michael Kenstowicz, ed., *Ken Hale: a Life in Language*. Cambridge, MA: MIT Press.

Chomsky, N. 2001b. "Beyond explanatory adequacy," MS, MIT, Cambridge, MA. A revised version to appear in A. Belletti, ed., *Structures and Beyond: Current Issues in the Theory of Language*, submitted to Oxford University Press.

Epstein, S. D. 1994. "The derivation of syntactic relations," MS, Harvard University.

Epstein, S. D. 1999. "Un-principled syntax and the derivation of syntactic relations," in S. D. Epstein and N. Hornstein, eds., *Working Minimalism*. Cambridge, MA: MIT Press, 1999.

Epstein, S. D., E. Groat, R. Kawashima, and H. Kitahara. 1998. *A Derivational Approach to Syntactic Relations*. Oxford: Oxford University Press.

Epstein, S. D. and N. Hornstein, eds., 1999. *Working Minimalism*. Cambridge, MA: MIT Press.

Epstein, S. D. and T. D. Seely. 1999. "SPEC-ifying the GB 'subject:' eliminating A-chains and the EPP within a derivational model," MS, University of Michigan and Eastern Michigan University; paper also presented at the LSA Summer Institute (1999).

Epstein, S. D. and T. D. Seely. 2000. "Cyclic Spell Out in 'derivation by phase,'" MS, University of Michigan and Eastern Michigan University.

Epstein, S. D. and T. D. Seely (forthcoming) *Transformations and Derivations*. Cambridge: Cambridge University Press.

Lasnik, H. 1999. "Chains of arguments," in S. D. Epstein and N. Hornstein, eds., *Working Minimalism*. Cambridge, MA: MIT Press, 1999.

Lasnik, H. 2001. "A Note on the EPP," *Linguistic Inquiry* 32, 2: 356–61.

Nunes, J. 1999. "Linearization of chains and phonetic realization of chain links," in S. D. Epstein and N. Hornstein, eds., *Working Minimalism*. Cambridge, MA: MIT Press, 1999.

Pesetsky, D. 1989. "Language particular processes and the Earliness Principle," paper presented at the GLOW colloquium, Utrecht, The Netherlands.

Uriagereka, J. 1999. "Multiple Spell Out," in S. D. Epstein and N. Hornstein, eds., *Working Minimalism*. Cambridge, MA: MIT Press, 1999.