The Strong Minimalist Thesis and Bounded Optimality*

DRAFT-IN-PROGRESS; SEND COMMENTS TO RICKL@UMICH.EDU

Richard L. Lewis
Department of Psychology
University of Michigan

27 March 2010

1 Purpose of this note

This note is an attempt to understand a fundamental question about language knowledge (and the issues concerning its formulation) raised in the Minimalist approach, as expressed, for example, in Chomsky (2005) and Chomsky (1998):

> *To what extent does language approximate an optimal solution to conditions that it must satisfy to be useable at all, given extralinguistic structural architecture?* (Chomsky, 2005).

Or, as Chomsky (1998) characterizes it:

> *How "perfect" is language. . . How good a solution is language to the general conditions imposed by the architecture of the mind/brain?*

Specifically, what follows is an initial attempt to create a (semi-)formal framework for formulating the issue as a problem of *bounded optimality*. Such formalization may of course be premature, and at this stage it is very abstract. But the hope is that doing so may yield the following benefits:

- The framework may help us to see how, exactly, current Minimalist theory—with its concerns for finding derivational procedures that are efficient, economical and so on—relates to the leading question identified above.

*This note originally developed from discussions in a graduate seminar on "competence and performance" that I co-taught in Fall 2009 at the University of Michigan with Sam Epstein, with support from the Marshall Weinberg Fund for Philosophy and the Cognitive Sciences. Please don't blame Sam or Marshall Weinberg for any nonsense contained here.

- The framework may afford the formulation of potentially new and interesting hypotheses concerning the nature of the "bounds" that define the problem for which language is a solution. (E.g, it may provide a new way of thinking about old questions concerning UG and its parameterization.)
- The framework may lead to the formulation of sharper questions concerning the explanatory nature of the posited bounds and "goodness metrics"—a better understanding of exactly what kinds of explanations are on offer via theories that appeal to such bounds or metrics (which comprise the "third factor" discussed in Chomsky (2005)).
- The framework may lead to computational modeling explorations that attempt to *derive* (through computational search) boundedly optimal solutions to highly simplified "micro-Minimalist" problems defined in terms of "interface" and other conditions.

2 Formalizing the "Strong Minimalist Thesis"

2.1 Syntax as a link between sound and meaning

We take as our starting point the traditional characterization of language as a system that links sound and meaning (Chomsky, 1998, 2005).

Let Π be the infinite set of possible (human) phonetic representations ("PF" representations)[1]. Without loss of generality (I think), assume Π to be countably infinite. Let Λ be the (countably) infinite set of possible (human) semantic representations ("LF" representations).

I depart now momentarily from Chomsky (1998) by first defining a language to be an infinite set of pairs of expressions, rather than a generator of such pairs, to be more explicit about the distinction between the set and the generator. The move is not crucial to the framework but I believe helps to simplify presentation—in any event we will arrive in short order to generators as abstract representations of cognitive state.

A *language* L is a set of pairs of expressions (π, λ) drawn from $\Pi \times \Lambda$. That is, $L \subset \Pi \times \Lambda$, or L is a *relation* between Π and Λ. (Note that thus far we haven't even claimed that L must be a function, etc.)

Let $\mathcal{L}$ be the set of all possible languages (i.e., the power set $\mathcal{P}(\Pi \times \Lambda)$). We are interested in identifying the subset $\mathcal{H}_{\mathcal{L}} \subset \mathcal{L}$ of possible *human* languages.

Any L is an infinite set of expression pairs (π, λ). Attributing "knowledge of L" to a mind requires representing L *intensionally* via some finite representation.

One class of finite representation of (possibly infinite) sets of expressions are *computable functions*. Without loss of generality (I think), let us restrict ourselves to the $\mathcal{L}_{comp} \subset \mathcal{L}$ that

[1] Not the most general move, given the facts about the modality-independence of language, but along with Chomsky (2005) I believe we can safely set this aside for now and return to it when we better understand the appropriate abstraction of the sensory-motor interface.

are computable. So our first conjecture is

$$\mathcal{H}_{\mathcal{L}} \subset \mathcal{L}_{comp} \tag{1}$$

Note that by moving to the class of computable functions, we are immediately committed to *computational/derivational/process* representations of knowledge state.

Now, $\mathcal{L}_{comp}$ includes all kinds of degenerate cases, such as L_{silent}: map every $\lambda \in \Lambda$ to the null phonological string. And L_{banana}: the (infinite set) of ordered pairs of the form ("banana", λ), for all $\lambda \in \Lambda$.

Because $\mathcal{L}_{comp}$ consists of computable sets, we can choose some formal language of computation to express the functions that generate those sets. We can then enumerate all the functions $f_1, f_2, \ldots$ in the set of functions $\mathcal{F}_{comp}$ that compute $L_1, L_2, \ldots \in \mathcal{L}_{comp}$. Note that $L_1, L_2, \ldots$ are infinite sets (of ordered pairs), but $f_1, f_2, \ldots$ are finite objects (representations of computable functions). Indeed, because the L's are infinite sets, we have no choice but to represent them formally via some finite (generative) notation.

Let's be very explicit about what the f's are (and are *not*) computing. Following Chomsky (1998), we take them to be generators of pairs. Thus, some particular computable function, let's say $f_{79} \in \mathcal{F}_{comp}$, will generate some ordered, countably infinite sequence of expression pairs:

$$\begin{array}{c} (\pi_1, \lambda_1) \\ (\pi_2, \lambda_2) \\ (\pi_3, \lambda_3) \\ \vdots \end{array} \tag{2}$$

Note that this function takes no input; in particular, it is *not* a mapping from sound to meaning or meaning to sound. Taken as a specification of a relation between Π and Λ, there is no constraint that the relation be a function in the sense of mapping one element of a set to one and only one element of the other set. The function simply enumerates paired expressions.

2.2 Bounded optimality

We now turn to formulating the bounded optimality problem. This requires specifying three parts: the space of candidate functions, the bounds, and a goodness function that is to be maximized. We have already (abstractly) identified the space of candidate functions, $\mathcal{F}_{comp}$ above.

Bounds. As I understand it, the key Minimalist bound is the following:

For language to be useable at all, it must satisfy the interface conditions ("legibility" requirements).

Note that we have already *implicitly* adopted this constraint above in the definition of $\mathcal{F}_{comp}$: we have restricted ourselves to mappings *between sound and meaning representations.* That is what the interface conditions demand. It is a natural move, though the resulting implicit nature of the interface conditions suggests that it might be useful to adopt a larger space of possible mental representations and possible languages as mappings between those representations—such that they need *not* satisfy the interface conditions by definition. Then one could formulate the interface conditions as a separate explicit bound, or one could move them into the goodness function (such that candidate functions are evaluated in part by "how well" they satisfy the interface conditions). This move could be easily made but I won't pursue that here.

But let's consider further possible constraints on "useability." Consider the mappings L_{silent} and L_{banana} above—"easy" mappings to represent and compute. But they somehow do not satisfy our notions of what a mapping between Λ and Π must be in order to be minimally useable, perhaps, for example, preserving information content in some coding sense (perhaps related to what has been called "Inclusiveness" conditions). Suppose we can make explicit such additional constraints on "useability", so that we can identify the subset of useable languages $\mathcal{L}_{use}$. We expect the useable languages to also be computable, so that

$$\mathcal{L}_{use} \subset \mathcal{L}_{comp} \tag{3}$$

Let $\mathcal{F}_{use}$ denote the functions that generate $\mathcal{L}_{use}$.

For now let's take useability to be just those constraints that are *independent* of computational or other considerations—acknowledging that such considerations may come into play in deriving the class of boundedly optimal useable languages (below). There are important choices here to be made between identifying some constraint as a hard bound on the optimality solution, or as a part of the goodness function, where it can thereby be treated as a soft constraint trading off against other demands.

Goodness function. We now define a *goodness function* G over $\mathcal{F}_{use}$ that says how "good" each function is— what it means for one function to be better than another. Suppose that the goodness function just maps an f to a real (absolute numbers don't matter of course, all that matters is the partial order imposed over $\mathcal{F}$):

$$G : \mathcal{F} \rightarrow \mathbb{R} \tag{4}$$

Perhaps we care about *description length*—the complexity of the representation of the function. Then we might take G to be a function (in part) of the inverse of the number of symbols in the representation of each $f \in \mathcal{F}_{use}$. Or perhaps we care about *computational com-*

plexity/efficiency, in the standard sense in computer science[2]. Perhaps we care about *cognitive demands* implied by the functions, so lower cognitive demands mean higher goodness values. The cognitive considerations may express themselves via the choice of computational architecture for representing the functions. Thus, in principle, one could adopt Turing machines, Post production systems, recursive function theory, etc.—or perhaps more fruitfully some other (Turing-equivalent) formalism that abstractly captures the cognitive considerations. Then, the computational complexity questions (how time- and space-efficient are the functions?) are addressed relative to the choice of that architecture.

Note, crucially, that whatever the consideration, we are *taking seriously the intensional representation (as a computational function) of the language as a claim about an object in the natural world, namely internal cognitive state of species members*. The goodness function here is predicated over (finite) function representations, not infinite sets of ordered pairs of (λ, π). And, as Chomsky (1998) points out, if computational complexity is part of the goodness function, then we are adopting the following assumption:

(iii) *Considerations of computational complexity matter for a cognitive system (a "competence system", in the technical sense of this term).* (Chomsky, 1998, p. 126).

The point here—and it is both a subtle and extremely fundamental point—is that (a) we are using computational functions to make (abstract) assertions about cognitive state; (b) we are *not* taking these functions as descriptions of the real-time, asymmetric processes used in speech production or comprehension; but (c) we nevertheless are *selecting out good functions* on the basis of computational concerns. A bit more on this move below.

The boundedly optimal solution. Once we have defined goodness G, then there is an *optimal* function in our space of possible "useable" functions:

$$f_{optimal} = \arg \max_{x \in \mathcal{F}_{use}} G(x) \tag{5}$$

[2]It is worth noting in passing here a possibly important divergence between typical concerns of computer scientists analysing the efficiency of algorithms, and the kinds of computational efficiency considerations that are appealed to in current syntactic theory. The kinds of distinctions that motivate most algorithm analysis in computer science concern very broad complexity classes—e.g. is the algorithm polynomial or exponential?. More specifically, such analyses are very rarely concerned with the values of *constants* in the complexity equations (does this component of the algorithm take 1 step or 7?). A conjecture is that most of the computational considerations in syntactic theory are about optimizing *very small constants* (e..g can we get the potential domain of targets for this relation down to *one* from a small number?). The point here is that *both approaches* can clearly be understood formally as instances of characterizing computational complexity, but the domains (bounded biological organism vs. bounded artificial computer system) may lead to very different goals. Thus, the computer scientist-as-algorithm-designer is happy to get to log or linear solutions and typically doesn't worry much about constant times because the effects of such constants will change in six months when new computer hardware comes out. I suspect that being very clear about (a) the potentially shared formal analysis technique and (b) the vastly different application domains will help to avoid much confusion in the future when further attempts are made to ground syntactic theory computationally.

Actually, there is an *equivalence class* of such optimal functions; let's denote that class by $\mathcal{F}_{optimal}$. Now let $\mathcal{L}_{optimal}$ be the sets of ordered pairs computed by $\mathcal{F}_{optimal}$. Then one form of the strong Minimalist thesis is:

$$\mathcal{H}_{\mathcal{L}} = \mathcal{L}_{optimal} \tag{6}$$

An important potential outcome: There are no "suboptimal" solutions. One possible potential outcome is that, once we properly understand the constraints/bounds, we will discover that the bounded optimality formulation is in fact a *degenerate* one, in the sense that there is only a *single* solution to the constrained problem, or an *equivalence* class of solutions that are equally good. Using our notation, this discovery would be:

$$\mathcal{H}_{\mathcal{L}} = \mathcal{L}_{use} \tag{7}$$

This is perhaps the strongest (in the sense of most deeply explanatory) possible result that this approach affords.

Avoiding a potential misconception. Note that *nothing* in the framework above guarantees that the solution to the bounded optimality problem will be perfectly useable in the sense that "all possible expressions can be parsed by the comprehension system and produced by the production system".

3 How derivational procedures fit into the framework

The discussion above is completely abstracted away from the machinery for describing the candidate functions in $\mathcal{F}_{use}$. As I understand it, most current Minimalist theory is about exploring such candidate functions, but I wish to make the link here very explicit.

Derivations start with some *numeration* N—some set of lexical items chosen from a finite set W. Let us assume that we have adopted some formal device that allows N to consist of *tokens* of items from W, so that a given lexical item may appear multiple times. However we represent tokens, there is some countably infinite set of possible numerations $\mathcal{N}_W$.

It seems to me that a common question and point of confusion that arises here is "where does the numeration come from?" The implicit expectation is that the numeration somehow serves as a kind of "input" for a language generation function. My understanding is that the numeration is simply a device that allows one to create computational functions that generate (infinite) pairs of expressions (λ, π). I wish to make this role explicit here, which renders the question of where the numeration comes from somewhat meaningless.

Suppose that we have a well defined computational function D that will take a numeration N and generate a finite set of pairs $(\lambda, \pi) \in \Lambda \times \Pi$:

$$D(N) = \{(\lambda_1, \pi_1), (\lambda_2, \pi_2), \ldots (\lambda_k, \pi_k)\} \tag{8}$$

The k presumably depends on D and the cardinality of N. D is some *derivational procedure*—internal to D may be a sequence of *merges*, for example.

D is a specific hypothesis about a candidate $f \in \mathcal{F}_{use}$. Recall that the f's take no input: they simply generate a sequence of expression pairs. We can now construct the f_D that corresponds to the derivational procedure D. The little procedure in Table 1 does this; it just depends on choosing the lexicon W. It simply sequences through the set of all possible numerations, applying D to each numeration (producing some k expression pairs for each one):

Table 1: The generating function f_D corresponding to derivational procedure D and lexicon W.

choose W
for each $N \in \mathcal{N}_W$ **do**
  output $D(N)$
end for

Now, to the extent that the goodness function G is concerned about efficiency of either computation of f_D or economy of representation of f_D, those evaluations will be determined by D—in fact we can substitute D for f_D and not change the solution to the bounded optimality problem. I'll refrain from formally stating this assertion and move on to more interesting issues.

4 The required elements of the framework and the nature of the explanations it provides

To summarize: the framework above suggests that a formulation of the bounded optimality problem must contain the following elements:

1. A characterization of Λ.
2. A characterization of Π.
3. A space of possible functions $\mathcal{F}$ that generate pairs of representations (potentially restricted to $\Lambda \times \Pi$).
4. Further bounds on that space (if any).
5. A goodness function G to compare f's in the space.

Then, to the extent that we discover that the human language faculty is an approximation to the $f_{optimal}$ that is the solution to our problem so defined, we have a potential answer to the leading question of the Minimalist approach, and a potentially deep explanation of the nature of the language faculty—where the nature of that explanation rests on the degree to which the specific elements of the problem formulation are minimal and independently motivated (the "third factor" bounds). We now turn to some questions raised by the framework, including how to theoretically interpret candidate elements of the bounded optimality formulation.

5 Questions raised by the framework

1. What is the minimal formulation of the *useability* conditions/bounds above, beyond the interface conditions? Something seems necessary to avoid the degenerate cases.

2. Why might G care about computational efficiency of candidate functions, if in fact these functions are *not* executed by the biological system in language use (the question implicit in (iii) above from Chomsky (1998))?

   > Here is one speculative answer: The computational efficiency of the performance system is *not* in fact unrelated to the computational efficiency of the generator functions f_D or derivational procedures D. It may be possible to formally prove something about this relationship.

3. Why might G care about the computational *simplicity* of candidate functions, if in fact these functions are *not* executed by the biological system in language use (the question implicit in (iii) from Chomsky (1998))?

   > Here is one speculative answer: The computational simplicity of the derivational procedures is somehow related to the likelihood that the biological computational mechanisms required for implementing the performance system would be "discovered" in evolution (a position related perhaps to Hauser, Chomsky, and Fitch (2002))—a position that does *not* presume, for example, that the mechanism must have been selected for language under adaptive pressures for communication.

4. What are the proper formulations of Λ and Π?

   > Here are some speculative answers: One primary constraint on Λ is that the representations take the form of *graphs* (a somewhat different move from taking *sets* as the key construct). One primary constraint on Π is that it consists of sequences. So *sequences* as an abstraction may be the important link between modalities (sign language, sound).

5. What are relevant *bounds* on the problem besides the interface conditions above?

Here is a speculative answer, hinted at above: Perhaps one relevant bound is to be found in the choice of computational architecture for expressing candidate functions (this might be a way to take into account the "extralinguistic structural architecture" of Chomsky (2005))—perhaps one that abstractly represents cognitive constraints, such as operating with a content-addressed memory subject to similarity-based interference.

Another speculative answer is that there are *no other bounds*.

6. What is the *minimal abstract formulation* of a boundedly optimal language micro-problem such that we can explore the solution to the micro-problem computationally in an interesting way?

   This would require fixing Λ, Π, the choice of formal language/architecture for representing functions, the goodness function G, useability constraints, and so on. It seems in principle possible, though the search space may be enormous even for the simplest formulation. The challenge is finding something that is both tractable with current hardware and worth doing in the sense that the exercise will potentially advance our understanding.

7. Where—if anywhere—is the locus of *learnability* constraints?

   Two possibilities emerge. One very interesting and potentially novel possible outcome is that UG is simply a parametric form of the set of functions $\mathcal{F}_{optimal}$ as defined above, with no further constraints required for learnability. This goes well beyond the horizons of current research, but it is perhaps the strongest possible result: a *derivation* of UG as the parametric form of $\mathcal{F}_{optimal}$, coupled with the discovery that this parametric form is sufficiently well-constrained (or essentially has no parameters at all) that it is clearly learnable.

   Another possibility is that *learnability* constraints themselves should be explicitly taken into account. The problem with this move is that may complicate the framework above by requiring goodness functions predicated over sets of functions (so sets of possible languages are evaluated in terms of learnability).

References

Chomsky, N. (1998). Some observations on economy in generative grammar. In P. Barbosa, D. Fox, P. Hagstrom, M. McGinnish, & D. Pesetsky (Eds.), *Is the best good enough? optimality and competition in syntax* (pp. 115–127). Cambridge, MA: MIT Press.

Chomsky, N. (2005). Three factors in language design. *Linguistic Inquiry*, *36*(1), 1–22.

Hauser, M. D., Chomsky, N., & Fitch, W. T. (2002). The faculty of language: What is it, who has it, and how did it evolve? *Science*, *298*, 156–579.