Comments on Hornstein

KLAUS ABELS

1. Introduction

Hornstein represents minimalism in linguistics as a project that takes certain generalizations about the human language faculty, or universal grammar, to be (roughly) empirically correct and whose aim it is to deduce these properties from a more fundamental theory of the human mind/brain.

According to him, the stock of empirically correct claims about the language faculty comes from the work in the principles and parameters framework, generalized phrase structure grammar, relational grammar, and lexical-functional grammar—all of which Hornstein views as ‘much the same theories in slightly different notation.’ The properties of the language faculty described by these theories are the targets of deduction.

To make progress on this project, the properties of the language faculty have to be stated in terms as simple and parsimonious as possible. Further, some of the properties of the language faculty might follow from what the literature calls ‘interface conditions’—conditions placed on the linguistic system by other cognitive modules that use the products of the linguistic system. Third, and this is crucial to the project described by Hornstein, some of the properties of the language faculty might follow from the way in which linguistic computations proceed. In this connection Hornstein introduces the following consideration: He assumes that the language faculty evolved in the human species rapidly, mentioning 50–100,000 years, which, if true, suggests that parochial, domain specific structure and computation within the language faculty must be quite limited; the time necessary for evolutionary tinkering to take place is simply not available. This, in turn, suggests that general computational principles (such as computational efficiency) or computational methods known to be involved in other cognitive domains can and must be invoked in an explanation of the properties of the language faculty.

After laying out this general program, Hornstein discusses the canonical binding theory, which was developed within government and binding theory, with the aim of illustrating how its properties can be thus deduced.

Address for correspondence: Department of Linguistics, UCL, Room 115B, Chandler House, 2 Wakefield Street, London WC1N 1PF, UK.
Email: k.abels@ucl.ac.uk

In this brief note, I will try to put Hornstein's project into a more general context. In particular, I will show what place this project has in the construction of an overall theory of language, when see as an information processing device. The general framework for this is provided by Marr, 1982. The project described by Hornstein seems both coherent and worthwhile to me. However, the examples that Hornstein provides to illustrate how this project is carried out, and indeed the vast majority of work in minimalism, are examples of quite a different project with no relation to the one advertised by Hornstein in the first part of his article. I refrain from discussing the merits and problems of the concrete linguistic proposal concerning the binding theory sketched by Hornstein in the second half of his article.

2. The Project

When generative linguists study the language faculty, they approach it as an information processing system (see also Neeleman, this volume). In his seminal book on vision, Marr (1982) points out that a complete understanding of any information processing system has to include theories at three distinct levels, which he calls the computational, the algorithmic, and the hardware levels.

A theory at the most abstract level, the computational level, must characterize what the information processing system does and why. The characterization of what the system does is, in essence, a description of the function the system computes, that is, a description of the input–output relations that characterize it. A complete theory at this level must also specify why this function is appropriate for the goal of the computation. At the next lower level, the level of algorithms and representations, the specific algorithm used by the system to compute the function characterized at the higher level is described and the internal and external representations are determined. Finally, at the lowest level, the hardware that implements the algorithm and how it implements the algorithm has to be described.

Marr illustrates the three levels using a very schematic model of a cash register. At the computational level, a cash register can be characterized as an adding machine. This is what the cash register does. Addition is the appropriate function for a cash register to compute, because intuitive notions about the behavior of the correspondence between the accumulation of goods bought and the price are correctly reflected by addition. At the next lower level, we can ask how the numbers are represented in the cash register (e.g. in a decimal or a dual representation) and by which of a number of different algorithms addition is performed. Finally, there is the question of how the algorithm is actually realized in the hardware of the cash register. The descriptions at the three levels constrain each other in broad terms but are otherwise independent of each other.

Figure 1 below reproduces Marr's own summary of the content of the description at each of these levels.

Computational Theory	Representation and algorithm	Hardware implementation
What is the goal of the computation, why is it appropriate, and what is the logic of the strategy by which it can be carried out?	How can this computational theory be implemented? In particular, what is the representation for the input and output, and what is the algorithm for the transformation?	How can the representation and algorithm be realized physically?

Figure 1 *The three levels at which any machine carrying out an information-processing task must be understood (from Marr, 1982, p. 25).*

Transformational grammar of the 60s, as Marr (1982, p. 28) himself points out 'is a true computational theory in the sense defined earlier. It is concerned solely with specifying what the syntactic decomposition of an English sentence should be, and not at all with how that decomposition should be achieved.' This characterization certainly remains true also for work done in generative theories of the 80s and early 90s. Theorizing at the most abstract, the computational level has remained the mainstay of work in theoretical linguistics.

Within this setup, Hornstein's premise that the results and generalizations emerging from government and binding theory of the 80s are roughly empirically correct, corresponds to the claim that those results and generalizations adequately characterize the function computed by the language faculty.

Following Marr, we should ask why this is the appropriate function to compute. Some of the considerations relevant to answering this question will revolve around issues of the purpose of linguistic computations. Views on this differ to a certain extent. Do linguistic computations structure thought? Do they only make independently existing and structured thoughts accessible to sensorimotor systems that can externalize and internalize them? Do they do both? While the first of these questions is not settled, linguists assume that linguistic computation certainly serves to interface between meaning representations and sensorimotor representations. Certain properties of the computational theory of the language faculty flow quite naturally from such very bare considerations. For example, the form-meaning pairing that the language faculty implements is locally compositional. Local compositionality is effectively forced in a system that is unboundedly expressive, a system in which the form-meaning correspondence at the level of the atomic sign is not predictable, and a system which is rapidly learnable.

While such considerations go a certain way towards motivating properties of the language faculty, they underdetermine the language faculty massively.[1] There

[1] The problem is made worse by the fact that too little is known about the truly language independent properties of meaning in the mind, which means that an important anchoring point is missing.

are very many functions that are compatible with the constraints arising from such general considerations. Natural language grammars represent a tiny, systematically structured subset of the range of logical possibilities (see Adger, this volume). So the question arises where the additional structure comes from.

An obvious place to look for answers to some of these questions is the algorithmic level.

I pointed out above that the theories of an information processing device at the computational and the algorithmic level depend on each other to a certain extent. An illustration might be useful here. Results from formal language theory in the 70s and 80s indicated that natural languages cannot be characterized by context free grammars (Bresnan *et al.*, 1982; Culy, 1985; Huybregts, 1976, 1984; Shieber,1985).[2] Clearly, considerations of the complexity of the grammar when viewed as a function, that is, when described at the computational level, puts a lower bound on the complexity of the algorithms that can compute it.

It is entirely conceivable that properties at the algorithmic level might determine what is going on at the computational level. In particular, if the constraints imposed on a computation are so weak that many different functions would satisfy them, then that function might be chosen which can be computed with a particularly simple or efficient algorithm, or maybe with routines that the overall system already possesses. While some literature on minimalism in linguistics aims to show that language is a perfect (maximally simple) solution to the boundary conditions, Hornstein in the present paper leans in a different direction. He remarks 'that focusing on computational efficiency can be misleading in an unhelpful way. The problem is not so much whether linguistic computations are efficient but whether they are cognitively provincial.'[3] In this context, the alleged fact that language arose on an evolutionarily short timescale of 50–100,000 years is intended to lend plausibility to the idea that linguistic computations are not 'cognitively provincial'.

The project, in other words, is to explain properties of the computational theory of the language faculty in terms of its algorithmic theory. An explanation of this sort commits the person proposing such an explanation to a number of sub-projects. First, he would need to describe the information processing system at the algorithmic level. As we saw this involves developing an algorithm that correctly describes the input–output profile characterized at the computational level. It would require proposing and defending a particular representational format for the data manipulated by the system. Further, to count as an explanation of

[2] Shieber, 1985 is usually credited with the formal argument that languages cannot be characterized by context free grammars even when viewed as string sets (weak generative capacity). As argued by Kracht, 2007, the type of data discussed in Huybregts, 1976 already settles the case once local compositionality, that is, strong generative capacity, is taken into account.

[3] In the context of Marr's three levels, the term 'computational efficiency' itself is somewhat misleading, since what is meant is not a property of the computational but of the algorithmic theory.

the higher level properties, one would need to show that alternative algorithms are less simple, less efficient, or, given again the evolutionary considerations, not independently available in the system. Similarly for the representational format, which would either have to be shown to be particularly suited to be manipulated by the particular algorithms or to preexist in the system.

The following quote from Hornstein and Pietroski, 2009 should make it clear that what I have presented here is not a far-fetched, straw man interpretation of what Hornstein takes to be minimalist project.

> We view the task of getting beyond mere description of innate linguistic constraints, and moving towards an explanation of how and why [the human faculty of language] has just these constraints, as akin to the task described by Marr (1982) in the context of studying vision. Given a description of the input-output profile for some posited cognitive system, one can and should ask how that system computes the relevant function, bearing in mind that nature has somehow realized the actual algorithm. And at least initially, it can be useful to abstract from various details of the input–output profile, in order to identify some fundamental operations—perhaps corresponding to an important subsystem—that are especially good candidates for realization.

The 'description of innate linguistic constraints' in the quote above is of course the body of facts and generalizations coming from the work of government and binding theory and its cousins. These are analogized with theories at the computational level. Recall that in the present paper Hornstein characterizes government and binding theory, generalized phrase structure grammar, lexical-functional grammar, and relational grammar as 'much the same theories in slightly different notation'. Hornstein's dismissal of notation as an important differentiating factor indicates that he locates the above-mentioned theories at the computational level. According to the view expressed in the above quote, to get 'beyond mere description of innate linguistic constraints' one has to consider the algorithmic theory of language.

In the preceding paragraphs I have tried to articulate as clearly as I could what I take Hornstein's characterization of the minimalist project to be. I hope to have managed to do this without undue distortion. As far as I can see, there is nothing principally unsound or categorically undoable about this project. Roughly the second half of Hornstein's article is intended as an illustration of how the project can progress and of the kind of fruit it can bear. However, in my view Hornstein makes no progress in this project, because he never embarks on it.

3. A Failure to Engage the Questions

Recall that the project by Hornstein—that of explaining properties of the language faculty at the computational level in terms of properties of the algorithmic theory—requires constructing and defending suitable algorithms and representations.

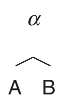
I find no trace of this in Hornstein's paper. After setting aside the notational differences between various theoretical schools in linguistics as irrelevant at the computational level, Hornstein adopts without discussion the standard minimalist representational system for phrase structure and the particular proposal for constructing phrase structure representations going back to Chomsky, 1995: phrase markers are built up bottom to top via the binary operation Merge.

While this move may be innocuous at the computational level, it is far from innocent at the algorithmic level. Here both the representation format and the method for building up complex representations need to be defended. Bottom up merger, as far as I can tell, is not a contender for a serious theory at the algorithmic level and was never meant to be. With this as background, consider the following passage from Hornstein's article:

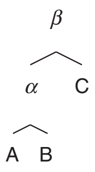
> What other properties might a 'nice' cognitive computation system have? Monotonocity/No Tampering (if α and β are inputs to an operation then their identities are preserved in the output of that operation, i.e. once a constituent, always a constituent; a nice feature for computational objects to have if constituents are required for further reasons e.g. interpretation, parsing etc.), . . .

The idea of the no tampering condition in minimalist syntax is the following: Say the operation Merge has created constituent α by combining A with B, (1-a). The gist of the no tampering condition is now that further applications of Merge are not allowed to alter the immediate dominance relations previously established. This reasoning allows (1-b); it rules out (1-c) as the output of any subsequent operation. The reason for this is that the immediate dominance relation established between α and B in (1-a) is no longer present in (1-c). This reasoning motivates the slogan: 'once a constituent, always a constituent'.

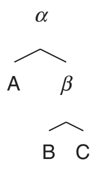
(1) a. Merge(A, B) =



b. Merge(Merge(A, B), C) =



c.



While the preservation of information, monotonicity might well be a 'nice' computational property, a change in representational format and algorithm would allow the transition from (1-a) to (1-c) without any violation of preservation of information. Suppose for the sake of illustration that the representation of phrase markers involved lists of precedence and dominance relations.[4] In such a system (1-a) would be characterized by (2-a), (1-b) by (2-b), and (1-c) by (2-c).

(2) a. dominates = {<α,A>, <α,B>}
precedes = {<A,B>}
b. dominates = {<α,A>, <α,B>, <β, C>, <β, α >, <β, A>, <β, B>}
precedes = {<A,B>, <A,C>, <B, C>}
c. dominates = {<α,A>, <α,B>, <α,β >, <α,C>, β,B>, <β,C>}
precedes = {<A,B>, <A,C>, <B,C>}

Given the representation format chosen in (2), the transition from (2-a) to (2-c) all of a sudden appears to be information preserving (monotonic, in accordance with the spirit of non-tampering). The first conclusion is that what appears to be a nice property in one representational system might not be a nice property in a closely related one. Therefore, if important results in syntax rely on the non-tampering condition and if, furthermore, an explanation of the non-tampering condition is to be derived from the algorithmic theory of the language faculty, then a detailed defense of the correctness of the representational format in (1) and the procedure generating complex structures is needed.

Alas, the prospects for this don't seem very bright. No work on the real-time production or processing of language uses bottom-up tree generation, while work that takes the contingencies of real-time language processing seriously and is therefore a much better contender for an algorithmic theory is likely to use representations of the type seen in (2) (see for example Phillips, 1996, 2003). Indeed, Phillips offers a theory of coordination and a resolution of Pesetsky paradoxes (Pesetsky, 1995) that crucially relies on information preservation being computed as in (2) rather than in (1).[5]

For the project that I extracted above from Hornstein's article, such issues are crucial. However, he fails to mention them, let alone engage them. Instead, the

[4] I am assuming here an encoding without a direct representation of immediate dominance and immediate precedence. Both can simply be derived: For all x, y, x immediately dominates/preceds y iff x dominates/precedes y and there is no z, such that x dominates/precedes z and z dominates/precedes y.

[5] Phillips's solution to Pesetsky paradoxes has been dealt a lethal blow in Lechner, 2003. For more recent discussion see also Janke and Neeleman, 2012. Nevertheless, the type of representation presented in (2) remains useful when characterizing parsing processes that have to recover phrase markers in real time and incrementally (for discussion see Abels and Neeleman, 2012).

abbreviated argument quoted above and the much more lengthy one about binding theory takes the representation and the generative methods for granted. No argument is given that these representations and methods are adequate characterizations of the algorithmic level. Indeed, it is more than dubious that they are. The account proposed of binding theory in the second half of the article therefore remains firmly at the computational level where arguments of computational complexity must necessarily take a different shape from the ones presented, since they must be independent of choice of methods and of the representation.

The above remarks are in no way intended to devalue Hornstein's approach to binding theory as a theory at the computational level. I have not discussed the question whether it is empirically adequate and passes criteria of simplicity and parsimony here. This would require a very different paper.

I have also not claimed that either the project Hornstein describes or the vastly different project that he actually embarks on are wrong-headed or incoherent. My point here is rather that the very interesting project sketched by Hornstein is not the project he or most practicing minimalists pursue; Hornstein's account does not deduce properties of the language faculty from considerations at the algorithmic level. Contrary to what he proposes as a research program, he doesn't even engage the relevant questions. To find out what minimalism is about, pay attention to what minimalists do, not to what they say they do.

Department of Linguistics
UCL

References

Abels, K. and Neeleman, A. 2012: e. Ms., UCL.

Adger, D. 1982: Construction and grammatical explanation: Comments on Goldberg. *Mind & Language*, 28, 466–78.

Bresnan, J., Kaplan, R., Peters, S. and Zaenen, A. 1982: Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13, 613-35.

Chomsky, N. 1995: Bare phrase structure. In G. Webelhuth (ed.), *Government and Binding Theory and the Minimalist Program*. Oxford: Basil Blackwell, 383-439.

Culy, C. 1985: The complexity of the vocabulary of Bambara. *Linguistics and Philosophy*, 8, 345-51.

Hornstein, N. and Pietroski, P. 2009: Basic operations: minimal syntax-semantics. *Catalan Journal of Linguistics*, 8, 113-39.

Huybregts, R. 1976: Overlapping dependencies in Dutch. *Utrecht Working Papers in Linguistics*, 1, 24-65.

Huybregts, R. 1984: The weak inadequacy of context-free phrase structure grammars. In G. de Haan, M. Trommelen, and W. Zonneveld (eds), *Van Periferie naar Kern*. Dordrecht: Foris, 81-99.

Janke, V. and Neeleman, A. 2012: Ascending and descending VPs in English. *Linguistic Inquiry*, 43, 151-90.

Kracht, M. 2007: The emergence of syntactic structure. *Linguistics and Philosophy*, 30, 47-95.

Lechner, W. 2003: Phrase structure paradoxes, movement and ellipsis. In K. Schwabe and S. Winkler (eds), *The Interfaces: Deriving and Interpreting Omitted Structures*. Amsterdam: John Benjamins, 177-205.

Marr, D. 1982: *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. New York: Freeman.

Neeleman, A. 2013: Comments on Pullum 2012. *Mind & Language*, 28, 522–31.

Pesetsky, D. 1995: *Zero Syntax*. Cambridge, MA: MIT Press.

Phillips, C. 1996: Order and structure. Doctoral dissertation, MIT.

Phillips, C. 2003: Linear order and constituency. *Linguistic Inquiry*, 34, 37-90.

Shieber, S. 1985: Evidence against the context-freeness of natural languages. *Linguistics and Philosophy*, 8, pp. 333-43.