On the locus of the linearizing algorithm

Noureddine Elouazizi
LUCL, Leiden University
n.elouazizi@let.leidenuniv.nl

One of the fundamental assumptions of the Minimalist Program is that the linear order of the syntactic objects emerges from a computational process of flattening their hierarchical grouping. This flattening process is argued to be induced by legibility conditions at the phonological interface (Chomsky 1995c), motivated by the physics of speech (Higginbotham 1983b) and executed through some version of the Linear Correspondence Axiom (LCA-Kayne (1994), Chomsky (1995c) and Uriagereka (2002)). Drawing on the study of two seemingly unrelated syntactic constructions from Tarifyt Berber, viz. constructions with clause bound multiple copy pronunciation of preverbal particles and constructions with object clitics copying in interpolation contexts, this paper argues against a modular view of the linearizing algorithm and defends an approach wherein linearization of the syntactic objects is an integral part of the narrow syntax (computational system- MERGE). It is the order of the application of MERGE, the input data-structure that MERGE applies to and the Spell Out points of the phases that yield the linear orders of the syntactic elements. The approach defended in this paper is shown to hold analytic and empirical consequences, including: (i) a syntax that correctly derives the linear order of the syntactic objects above and below the word level in a unified way and (ii) it collapses the interpolation effects observed with the object clitics constructions in Tarifyt Berber and other languages which attest it (e.g. European Portuguese) to the interpolation effects observed with the WH-constructions in some Slavic languages (e.g. Serbian, Bulgarian and Macedonian) and derives the similarities and differences from a single grammar mechanism, viz., MERGE. Conceptually speaking, the theory of linearization that this paper defends is fundamentally compatible with a dynamic derivational view of phases and syntax, precisely MERGE, is the optimal way to meet interface requirements without being reducible into these same requirements.