Syntax-Prosody in Optimality Theory

Jenny Bellik & Nick Kalivoda

Much recent work on the syntax-prosody interface (Truckenbrodt 1999; Selkirk 2011; Elfner 2012; Myrberg 2013; Ito & Mester 2013; among many others) has fruitfully employed violable constraints within the framework of Optimality Theory (OT; Prince & Smolensky 1993).

An OT system consists of two functions, Gen and Con. In practice, most research has focused on Con—developing a constraint set and ranking that successfully model the phenomenon of interest. Gen has typically received less attention, with only a small number of candidates that could reasonably be expected to be optima under some ranking being explicitly considered in the analysis. This is probably due to the fact that manually generating all the possible candidates is prohibitively time-consuming for input syntactic structures with even a few terminals. In addition, since both inputs and outputs in syntax prosody work take the form of trees, it is impossible to generate and evaluate them automatically using regular expressions (as in OTWorkplace), which are too low in the Chomskyan hierarchy of language types to represent trees of arbitrary size.

To address these problems, we have developed SPOT (Bellik, Bellik & Kalivoda 2017; accessible online here), a JavaScript application that automatically generates prosodic candidate sets, and automatically evaluates each constraint for each candidate. In this talk, we illustrate the exponential increase in the number of relevant prosodic output candidates as the number of terminals in the syntactic string increases, and demonstrate how to use SPOT's HTML interface to automatically generate these candidates. We also show how SPOT is able to quickly and automatically assign violations for these candidates.

We also show how SPOT can be used to refine Con. Once the full set of candidates is considered, the precise formulation of each constraint can have unintended consequences for the computation of optimality. Small variations in the algorithmic definitions of familiar constraints like EqualSisters, Binarity, and NonRecursivity will be shown to make significantly different predictions.