Syllable structure in Optimality Theory

1. Syllable structure: The view from OT

When we analyzed syllable structure before, there were two main steps in our procedure:

Step 1.	Determine **what syllable structure is like** in the language we are analyzing.

In Step 1, we used phonological evidence from the data to determine the language's choice for as many of the "syllabification options" as possible. (Allowable nuclei/onsets/codas/clusters?)

Step 2.	Make our **model of the mental grammar** build the syllable structure that we need for the language we are analyzing.

We experimented briefly with taking care of Step 2 by means of three **syllable-building rules**: the Nucleus Rule, the Onset Rule, and the Coda Rule. But these raised problems and questions!

Now that we are working in OT, rather than a rule-based approach to phonology, this gives us a new way to handle Step 2. (Note, however, that Step 1 remains the same! We still need to use evidence to determine what the syllables of a given language are like.)

Our new model of the mental grammar **contains no rules at all**. This means that we won't be referring to any "Nucleus Rule" or "Onset Rule" when we talk about syllable structure. As always in an OT analysis, our goal is to make the mental grammar use well-motivated constraints to **choose the actual output** over other plausible, hypothetical outputs that the language might have chosen (which are often outputs that some other language *does* choose).

2. A set of important syllable-structure constraints

Here is a set of basic syllable-structure constraints that are widely used in OT work.

Onset	Assign one * for every syllable that does not begin with an onset
NoCoda	Assign one * for every syllable that ends with a coda
NoOnsetCluster	Assign one * for every syllable that has more than one segment in the onset (this constraint is also known as *ComplexOnset)
NoCodaCluster	Assign one * for every syllable that has more than one segment in the coda (this constraint is also known as *ComplexCoda)

3. Other relevant constraints

These two constraints are also important to consider in most syllable-structure analyses.

NoDeletion	Assign one * for every segment in the input that is not in the output (this constraint is often called Max; its ancestor was called Parse)
NoEpenthesis	Assign one * for every segment in the output that is not in the input (this constraint is often called Dep; its ancestor was called Fill)