

OT fundamentals: Constraints and constraint tableaux

1. From goals to constraints

- (1) “Goals” are formalized in OT as **constraints**
- (a) When we propose a constraint, we need to give it a **formal definition** that states the conditions under which that constraint assigns a **violation** (*’)
- Constraint definitions refer to the entities in our model of phonological representations, such as features, word boundaries, syllable structure, etc.
 - It is useful to give the constraint a convenient name, and provide a plain-language paraphrase of what goal it represents, but the **definition is key**
- (b) Ideally, each constraint formalizes one **simple** goal
- Complicated patterns should come from the **interaction** of simple constraints, not from constraints that are themselves complex
- (2) Let’s take the “goals” we identified for Cairene Arabic and English, and **formalize** them as constraints
- (a) “avoid codas” can be formalized as this constraint:
 NoCODA Assign one * for every syllable that has a coda
- (b) “avoid onset clusters” can be formalized as this constraint:
 NoONSETCLUSTER Assign one * for every syllable that has multiple segments in the onset
- (3) Constraint **ranking** is represented by the symbol “ » ” or “ >> ”
- (a) The relative priority of different constraints in a given language depends on their ranking in that language’s **constraint hierarchy** — a higher-ranked constraint will be satisfied at the expense of a lower-ranked constraint
- (b) The phonological patterns of languages differ because their constraint rankings differ (the constraints are universal!)
- **A » B** means “A **dominates (outranks, is ranked higher than) B**”
 - For Cairene Arabic, we proposed that NoONSETCLUSTER » NoCODA

2. Constraint tableaux: How the mental grammar maps a UR to an SR

- (4) In OT, the grammar *doesn’t* take a UR and *gradually derive* a SR by applying rules
- (a) The grammar takes a UR,
- (b) considers **all SRs** that languages *might* choose for this UR (more on this later),
- (c) and determines which SR is chosen as the winner by the constraint hierarchy

- (5) We use a **constraint tableau** to show input, outputs, constraints, and violation marks
- *Tableau* is a French loanword, so the plural is spelled *tableaus* or (French) *tableaux*

| /faslu/ 'his term' | NoONSETCLUSTER | NoCODA |
|--------------------|----------------|--------|
| → a. fas.lu | | * |
| b. fa.slu | *! | |

- (a) The **input** is *usually* the same as a UR (more on this later)
- The input is placed in the top left corner of the tableau
 - The competing output candidates are generated from the input
- (b) The **output candidates** are the potential surface forms that compete to be chosen
- The output candidates are arranged in the leftmost column, in whatever order seems most useful for the discussion
- (c) The **optimal output** (also called the **winning candidate** or **winner**) is indicated with the “pointing finger”, or with an arrow or another convenient symbol
- The other candidates are sometimes called ‘losing candidates’ or ‘losers’
- (d) The **constraints** are listed from left to right, with the highest-ranked first
- Two constraints separated by a **solid line**: the left one **dominates** the right one
 - Two constraints separated by a **dashed line**: **no ranking relationship** has been established between the two
- (e) Each time a candidate violates a constraint, a **violation mark** ‘*’ is added to the tableau in the appropriate cell
- A **fatal violation**, which is a violation that makes a candidate lose, is sometimes marked with ‘!’, although this notation can become confusing in a complex tableau and is therefore avoided by some authors
- (6) A tableau can be used to make either of the following kinds of argument:
- (a) If you know the input and the output, a tableau can be used to *prove that a particular constraint ranking is necessary* for the right output to win
- This technique is known as a **ranking argument**
 - This scenario is what we are doing when we are doing familiar kinds of phonological analysis: trying to make a proposal about the **grammar** (the constraint ranking) for a **particular language**
- (b) If you know the ranking, then a tableau can be used to show *what output would win* for a particular assumed input, or *what input(s) should be proposed* to ensure that a particular output will win
- This technique is useful for testing the **predictions** of a constraint ranking
 - This technique also allows us to test the predictions of a **constraint set**, if we look at *all the possible rankings* for the constraints in that set (more on this later)