



# OT-HELP 2.0: A COMPUTATIONAL TOOL FOR PHONOLOGICAL ANALYSIS

ALI NIRHECHE
anirheche@umass.edu

**Workshop at FLSHR-UM5, Rabat**
**January 2nd, 2026**

# ROADMAP

- OPTIMALITY THEORY VS. HARMONIC GRAMMAR
- WHY USE COMPUTATIONAL TOOLS?
- OT-HELP 2.0
- CASE STUDY 1: ENGLISH NASALS
- CASE STUDY 2: DEFINITE ARTICLE ASSIMILATION IN MSA
- REAL-WORLD APPLICATION

# OPTIMALITY THEORY
# VS.
# HARMONIC GRAMMAR

# THEORETICAL REFRESHER: OPTIMALITY THEORY

- **Core architecture:**
  - **GEN:** Generates infinite candidates.
  - **CON:** UG provides a set of universal constraints (markedness and faithfulness)
  - **EVAL:** Evaluates candidates based on a hierarchy of constraints.
- **Strict Domination:**
  - Constraints are ranked (C1≫C2≫C3).
  - A violation of a higher-ranked constraint is fatal, regardless of how many times lower constraints are violated.
  - *The winner:* The candidate that survives the highest fatal violations.

# BEYOND RANKING: HARMONIC GRAMMAR

- **The shift:**
  - Proposed by Legendre et al (2006).
  - Moves from Ranking to *Weighting*.
- **How it works:**
  - Each constraint has a numerical **Weight** ($w$).
  - Violations are negative integers (usually -1).
  - **Harmony (*H*):** The sum of weighted violations:

$$\text{Harmony}(H)=\sum(\textit{Weight}\times\textit{Violations})$$

  - **The winner:** The candidate with the highest (least negative) Harmony score.

# BEYOND RANKING: HARMONIC GRAMMAR

- Let's take this language which enforces the requirement to only have open syllables through deletion.
- An OT tableau where NoCoda and Dep outranks Max:

| /CVC/ | NoCoda | Dep | Max |
|---|---|---|---|
| a. ☞ CV | | | * |
| b. CVC | *! | | |
| c. CV.CV | | !* | |

- An HG tableau where NoCoda and Dep has more weight than Max:

| /CVC/ | NoCoda $w = 2$ | Dep $w = 2$ | Max $w = 1$ | $\mathscr{H}$ |
|---|---|---|---|---|
| a. ☞ CV | | | -1 | -1 |
| b. CVC | -1 | | | -2 |
| c. CV.CV | | -1 | | -2 |

# WHY HARMONIC GRAMMAR? (THE "GANG EFFECT")

- **The limitation of OT:**
  - In OT, if Constraint A dominates B and C ($A \gg B, C$), one violation of A overcomes any number of violations for B and C.
- **Cumulative interaction (Gang Effects):**
  - In HG, lower-weighted constraints can "gang up" to overcome a higher-weighted constraint.
- **Harmonic Grammar** allows us to test if a language requires this kind of cumulative interaction.

# EXAMPLE OF THE GANG EFFECT

- No gang effect in OT:

| /input/ | A | B |
|---|---|---|
| a. ☞ candidate 1 | | * |
| b. candidate 2 | *! | |

| /input/ | A | C |
|---|---|---|
| a. ☞ candidate 1 | | * |
| b. candidate 2 | *! | |

| /input/ | A | B | C |
|---|---|---|---|
| a. ☞ candidate 1 | | * | * |
| b. candidate 2 | *! | | |

- Gang effect in HG:

| /input/ | A $w = 4$ | B $w = 2.5$ | $\mathscr{H}$ |
|---|---|---|---|
| a. ☞ candidate 1 | | -1 | -2.5 |
| b. candidate 2 | -1 | | -4 |

| /input/ | A $w = 4$ | C $w = 2.5$ | $\mathscr{H}$ |
|---|---|---|---|
| a. ☞ candidate 1 | | -1 | -2.5 |
| b. candidate 2 | -1 | | -4 |

| /input/ | A $w = 4$ | B $w = 2.5$ | C $w = 2.5$ | $\mathscr{H}$ |
|---|---|---|---|---|
| a. candidate 1 | | -1 | -1 | -5 |
| b. ☞ candidate 2 | -1 | | | -4 |

# WHY USE COMPUTATIONAL TOOLS?

# THE LIMITS OF "PEN AND PAPER" PHONOLOGY

- **Small-scale systems:**
  - We are used to analyzing small datasets (3-4 candidates, 3-4 constraints).
  - It is easy to find a ranking for a single input by hand.
- **The reality of research:**
  - Linguistic systems often involve dozens of inputs and constraints.
  - **The factorial growth of rankings:**
    - 3 Constraints = 3! (6 possible rankings).
    - 5 Constraints = 5! (120 possible rankings).
    - 10 Constraints = 10! (3,628,800 possible rankings).
- **The human error factor:**
  - It is often difficult to manually check if a ranking for Input A contradicts the ranking required for Input B and so on.

# ADVANTAGES OF USING COMPUTATIONAL TOOLS

- **Consistency:**
  - Computational tools don't make judgment calls. They apply the logic you give them.
  - They ensures your analysis is internally consistent across all data points.
- **Modeling the learning process (L1 Acquisition):**
  - Many tools (including OT-Help) use learning algorithms (e.g., Recursive Constraint Demotion) that replicate how L1 learners arrives at a grammar.
- **Factorial typology:**
  - Beyond analyzing one language, we want to know: What other languages does my constraint set predict?
  - Does your analysis predict impossible languages? (Overgeneration).
  - Does your analysis miss attested languages? (Undergeneration).

# MODELING THE LEARNER

- **The learning problem:**
  - A child (the learner) is born with a set of universal constraints (CON).
  - The child hears “data" (the winners) from their caregivers.
  - The Task: Find the specific ranking that makes those winners optimal.
- **Replicating the learner:**
  - Computational tools implement Learning Algorithms theorized to reflect how the human mind works.
  - Recursive Constraint Demotion (RCD) is an example of an algorithm used in such tools.

# HOW RCD WORKS

- Tesar & Smolensky (1998) proposed that learning happens "top-down“.
- The learner compares the Winner (what they heard) against a Loser (what their current grammar thinks is better).
  - The learner identifies constraints that prefer no losers (those with no 'L' marks).
  - These constraints are placed in the highest possible ranking tier.
  - The learner then discards any loser that is ruled out by those constraints.
  - The process repeats for the remaining constraints and remaining losers.

# HOW RCD WORKS (EXAMPLE)

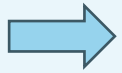
- Let's look at our earlier pattern of coda deletion and see how the ranking is learned under RCD.
- **Step 1:** All constraints are unranked.

| /CVC/ | MAX | NOCODA | DEP |
|---|---|---|---|
| a. CV ~ CVC | L | W | |
| b. CV ~ CV.CV | L | | W |

# HOW RCD WORKS (EXAMPLE)

- **Step 2:** both NoCoda and Dep do not prefer a loser, so they are put at  a higher ranking tier.

| /CVC/ | Max | NoCoda | Dep |
|---|---|---|---|
| a. CV ~ CVC | L | W | |
| b. CV ~ CV.CV | L | | W |



| /CVC/ | NoCoda | Dep | Max |
|---|---|---|---|
| a. CV ~ CVC | W | | L |
| b. CV ~ CV.CV | | W | L |

# HOW RCD WORKS (EXAMPLE)

- **Step 3:** NoCoda and Dep eliminate CVC and CV.CV.

| /CVC/ | NoCoda | Dep | Max |
|---|---|---|---|
| a. ~~CV ~ CVC~~ | ~~W~~ | | ~~L~~ |
| b. ~~CV ~ CV.CV~~ | | ~~W~~ | ~~L~~ |

- Since there are no other losers, the final ranking is: **NoCoda, Dep >> Max**

# OT-HELP 2.0

# TOOLS FOR RUNNING OT AND HG SIMULATIONS

- **OTSoft (Hayes et al 2013):**
  - An excel-based powerful tool for OT analysis, but requires Windows/Excel setup.
- **Praat (Boersma & Weenink 2022):**
  - Known for phonetic analysis, but contains an OT learning module (GLA), but has a steep learning curve and requires coding knowledge.
- **OT-Help (Staubs et al 2010):**
  - **Our focus today**
  - Java/web-based tool.
  - User-friendly interface.
  - It can solve for **both** Ranked Constraints (OT) and Weighted Constraints (Harmonic Grammar).

# SETTING UP OT-HELP 2.0

1. Go to OT-help 2.0 webpage: https://people.umass.edu/othelp/download.html
2. Download OT-help 2.0 by clicking on OT-Help2.jar
3. Download or124.jar. This file must go inside a folder you should create and name "OTH-lib". This folder should be within the same folder where you have OT-Help2.jar
4. If you don't have Java installed, you must install it. Here is where to download Java: https://www.java.com/en/download/
   - These instructions work for both Windows and Mac.

# HOW OT-HELP 2.0 WORKS

- **Input:**
  - A simple .txt file (tab-delimited).
  - Format: Matrix of inputs, candidates, and violations (basically your OT tableau).
- **Processing:**
  - **For OT:** Uses Recursive Constraint Demotion (Tesar 1995; Tesar and Smolensky 1998) to find a ranking.
  - **For HG:** Uses Linear Programming to find weights.
- **Output:**
  - Solvable? (Yes/No).
  - The ranking or weights.
  - The predicted typology (list of all possible languages).

# FORMATTING INPUT FILE

- **Rows 1 & 2:** Constraint names
- **Column 1:** Inputs
- **Column 2:** Candidates
- **Column 3:** Winner indication (Put a "1" next to the actual output)
- **Columns 4+:** Violation marks (Integers: 1, 2, etc.)

| | | | Dep | NoCoda | Max |
|---|---|---|---|---|---|
| | | | Dep | NoCoda | Max |
| CVC | CV | 1 | | | 1 |
| | CVC | | | 1 | |
| | CVCV | | 1 | | |

- Let’s run the coda deletion simulation in OT-help.

# CASE STUDY 1: ENGLISH NASALS

# ENGLISH NASALS (DATA FROM UMASS AMHERST LING302)

- Let's look at the nasal and oral vowels of English. What is their distribution?

| | | | | | |
|---|---|---|---|---|---|
| 1. | kɑd | 'cod' | 5. | kɑ̃n | 'con' |
| 2. | pæ̤d | 'pad' | 6. | pæ̤̃n | 'pan' |
| 3. | spɹɪg | 'sprig' | 7. | spɹɪ̃ŋ | 'spring' |
| 4. | eɪb | 'Abe' | 8. | ẽɪm | 'aim' |

# GENERALIZATION

- **The generalization:**
  - Vowels are **oral** generally.
  - Vowels become **nasalized** only when preceding a nasal consonant.
  - This is an **allophonic** distribution (Complementary Distribution).
- **The goal:**
  - We want our grammar to force nasalization in:
    - /pan/ → [pæ̃n]
  - But prevent it in:
    - /pad/ → [pæd].

# CONSTRAINTS

- **Markedness constraints:**
  - ***VN:** Assign one violation mark to every sequence of Oral Vowel + Nasal Consonant. (Don't have oral vowels before nasals).
  - ***V-nas**: Assign one violation mark to every nasal vowel. (Nasal vowels are marked).
- **Faithfulness constraint:**
  - **IDENT(nasal):** Output nasality must match Input nasality.
- What is the ranking of these constraints? Create tableaux for /pad/ and /pan/.

# OT ANALYSIS

- **Ranking:**
  - ***VN >> *V-nas, IDENT(nasal)**

| | | *VN | *V-nas | IDENT(nasal) |
|---|---|---|---|---|
| /pad/ | ☛ pad | | | |
| | pãd | | *! | * |
| /pan/ | pan | *! | | |
| | ☛ pãn | | * | * |

# HG ANALYSIS

- **Weighting:**
  - The weight of *VN alone must be heavier than the sum of the weights of *V-nas and IDENT(nasal) combined.

| | | ***VN w = 3** | ***V-nas w = 1** | **IDENT(nasal) w = 1** | ***H*** |
|---|---|---|---|---|---|
| /pad/ | ☛ pad | | | | **0** |
| | pãd | | -1 | -1 | -2 |
| /pan/ | pan | -1 | | | -3 |
| | ☛ pãn | | -1 | -1 | **-2** |

# CREATING THE INPUT FILE

- **Instructions:**
  1. Open your text editor (e.g., notepad).
  2. Type the constraint names in the first two rows.
  3. Enter the inputs /pan/ and /pad/ with their candidates (do not use '/' in the text editor).
  4. Use **"1"** to mark the winner, and violations.
- Note:
  - *In the text file, use TABS between columns, not spaces*.
  - You should use HTML code for special characters like IPA symbols and diacritics (ã -> ã)
  - *You may also edit the .txt file in an Excel spreadsheet.*

| | | | Dep | NoCoda | Max |
|---|---|---|---|---|---|
| | | | Dep | NoCoda | Max |
| CVC | CV | 1 | | | 1 |
| | CVC | | | 1 | |
| | CVCV | | 1 | | |

# CREATING THE INPUT FILE

- Check your text file against the one below.

| | | | *VN | *V-nas | IDENT(nasal) |
|---|---|---|---|---|---|
| | | | *VN | *V-nas | IDENT(nasal) |
| pad | pad | 1 | | | |
| | pãd | | | 1 | 1 |
| pan | pan | | 1 | | |
| | pãn | 1 | | 1 | 1 |

# RUNNING THE OT SOLUTION

- **Instructions:**
  1. Open OT-Help2.jar.
  2. Drag your text file into the OT-Help window.
  3. Click **"OT Solution"**.
  4. Analyze the Result:
     - Status: **Solved** (Blue background).
     - Ranking Found: ***VN** >> ***V-nas**, **IDENT(nasal)**
     - Does this match our hand-written analysis?
     - What can you say about the typology?

# RUNNING THE HG SOLUTION

- **Instructions:**
  - Go back to the main page (or refresh).
  - Click **“HG Solution"**.
  - Analyze the Result:
    - Look at the numbers assigned.
    - Does this match our hand-written analysis (the weight of *VN is heavier than the sum of the weights of *V-nas and IDENT(nasal) combined)?

# CASE STUDY 2: DEFINITE ARTICLE ASSIMILATION IN ARABIC

# DEFINITE ARTICLE ASSIMILATION IN MSA

- Let's look at the following data from MSA. What is the generalization?

| | | |
|---|---|---|
| a. | al-bint | DEF-girl |
| | al-kalaam | DEF-speech |
| | al-qaanuun | DEF-law |
| | al-ʕajn | DEF-eye |
| b. | ar-rabiiʕ | DEF-spring |
| | aθ-θawb | DEF-tissue |
| | as-salaam | DEF-peace |
| | aʃ-ʃams | DEF-sun |

# GENERALIZATION

- **General rule (faithfulness):**
  - Usually, the /l/ remains [l].
- **Assimilation context:**
- When /l/ is followed by a **Coronal** consonant ($t, d, s, z, \theta, r, n, \int$), it totally assimilates.
  - /al-salaam/ → [as-salaam]

# ADDITIONAL DATA

- Now, let’s look at the following additional data:

| | |
|---|---|
| al-ʒanuub | DEF-south |
| al-ʒaziira | DEF-island |
| al-ʒabal | DEF-mountain |
| al-ʒuhd | DEF-effort |

# BLOCKING EFFECT

- **The blocking effect:**
  - The sound /ʒ/ is coronal, so we *expect* it to assimilate (*[aʒ-ʒanuub]).
  - **However**, it does not! It stays [al-ʒanuub].
  - The assimilation is **blocked** specifically by /ʒ/.

# CONSTRAINTS

- To model this pattern, we need three constraints:
  - ***l-[cor]:** Assign one violation mark to every sequence of [l] + coronal consonant (this drives the assimilation).
  - **IDENT(lat):** The output lateral feature value must match the input. (Don't change /l/ to [s], [t], etc.).
  - ***ʒʒ**: Assign one violation mark to every geminated [ʒ] in the output.
- What is the ranking of these constraints? Create tableaux for /al-bint/, /al-ʃams/, and /al-ʒanub/.

# OT ANALYSIS

- **Ranking:**
  - ***ʒʒ >> *l-[cor] >> IDENT(lat)**

| | | *ʒʒ | *l-[cor] | IDENT(lat) |
|---|---|---|---|---|
| /al/+/bint/ | ☛ albint | | | |
| | abbint | | | *! |
| /al/+/ʃams/ | alʃams | | *! | |
| | ☛ aʃʃams | | | * |
| /al/+/ʒanub/ | ☛ alʒanub | | * | |
| | aʒʒanub | *! | | * |

# HG ANALYSIS

- **Weighting:**
  - The weight of *l-[cor] must be higher than the weight of IDENT(lat) and lower than the combined weights of IDENT(lat) and *ʒʒ.

| | | ***ʒʒ**<br>w = 2 | ***l-[cor]**<br>w = 2 | **IDENT(lat)**<br>w = 1 | ***H*** |
|---|---|---|---|---|---|
| /al/+/bint/ | ☛ albint | | | | **0** |
| | abbint | | | -1 | -1 |
| /al/+/ʃams/ | alʃams | | -1 | | -2 |
| | ☛ aʃʃams | | | -1 | **-1** |
| /al/+/ʒanuub/ | ☛ alʒanuub | | -1 | | **-2** |
| | aʒʒanuub | -1 | | -1 | -3 |

# RUNNING THE OT AND HG SOLUTIONS

- **Activity:**
  - Create a .txt input file for this problem
    - Make sure to either use HTML codes or alphabet letters for IPA symbols:
      - For [ʃ], you can either use [sh] or [ʃ]
      - For [ʒ], you can either use [zh] or [ʒ]
  - Run the OT and HG solutions on OT-help

# REAL-WORLD APPLICATION

# THE REALITY OF RESEARCH

- So far, we have looked at small-scale problems (3-4 constraints, 2-3 tableaux, 2 candidates)
- In actual phonological and morphological research, the data is rarely this clean.
- **Case Study: Moroccan Arabic Broken Plurals:**
  - This is data from my current work on Moroccan Arabic broken plural.
  - I aim to predict how broken plurals can be derived from their corresponding singulars.
  - **The complexity:**
    - Multiple possible affixes.
    - Multiple changes between input and output: vowel changes, deletion, and epenthesis.
    - Many possible outputs for the same input.

# DATA: CCaC PLURALS

- CCaC plurals are derived from CCC and CVC roots.

| | Root | Singular | Plural | Gloss |
|---|---|---|---|---|
| a. | klb | kəlb | klab | ‘dog’ |
| | bnt | bənt | bnat | ‘girl’ |
| | qnt | qənt | qnat | ‘corner’ |
| b. | dib | dib | djab | ‘wolf’ |
| | bir | bir | bjar | ‘well’ |
| | fil | fil | fjal | ‘elephant’ |

# DATA: CCuC PLURALS

- CCuC plurals are derived from CCC and CVC roots.

| | Root | Singular | Plural | Gloss |
|---|---|---|---|---|
| a. | dnb | dənb | dnub | ‘sin’ |
| | frʕ | fərʕ | fruʕ | ‘branch’ |
| | knz | kənz | knuz | ‘treasure’ |
| b. | dˤar | dˤar | dˤjur | ‘house’ |
| | rasˤ | rasˤ | rjusˤ | ‘head’ |
| | nif | nif | njuf | ‘nose’ |

# DATA: CCaCi PLURALS

- CCaCi plurals are derived from CCC and CVC roots.

| | Root | Singular | Plural | Gloss |
|---|---|---|---|---|
| a. | dʕw | dəʕwa | dʕawi | ‘prayer’ |
| | frq | fərqa | fraqi | ‘girl’ |
| | ħkm | ħəkma | ħkami | ‘cheek’ |
| b. | lil | lila | ljali | ‘night’ |
| | ʕaf | ʕafja | ʕwafi | ‘fire’ |
| | saq | saqja | swaqi | ‘flume’ |

# DATA: CCaCəC PLURALS

- CCaCəC plurals are derived from CCCC, CVCC and CCVC roots.

| | Root | Singular | Plural | Gloss |
|---|---|---|---|---|
| a. | fndq | fəndəq | fnadəq | 'hotel' |
| | mskn | məskin | msakən | 'poor' |
| b. | xatm | xatəm | xwatəm | 'ring' |
| | gitˁn | gitˁun | gjatˁən | 'tent' |
| c. | blasˁ | blasˁa | blajəsˁ | 'place' |
| | dqiq | dqiqa | dqajəq | 'minute' |

# THE SCALE OF THE PROBLEM

- **The Dataset:**
  - **9 Tableaux** (Different input types)
  - **Up to 6 Candidates** per tableau (Competing outputs)
  - **8 Constraints**
- **Why you cannot do this by hand:**
  - With 8 constraints, there are **40,320** possible rankings
  - It is cognitively impossible to track crucial ranking arguments across 40,000 possibilities without making an error
- Let’s run this problem on OT-Help

# BEYOND OT-HELP

- It gets even more complicated when the data involving lexical statistics, variability, and exceptionality: OT-help can't handle these cases.
- Consider this additional broken plural data from Moroccan Arabic:

| | Root | Singular | Plural | Gloss |
|---|---|---|---|---|
| a. | ktb | ktab | ktub~ktuba | 'book' |
| | qlb | qəlb | qlub~qluba | 'heart' |
| b. | bit | bit | bjut~bjuta | 'house' |
| | ħitˁ | ħitˁ | ħjutˁ~ħjutˁa | 'wall' |

- Let's try to run add this data to the input file and run it in OT-help

# BEYOND OT-HELP

- Another example that involves variable and exceptional patterns is definite article assimilation in Moroccan Arabic, which is far more complicated than in MSA.
  - **128 Tableaux** (Different input types)
  - **3 Candidates** per tableau (Competing outputs)
  - **369 Constraints**
- More recent tools and algorithms can handle such cases, e.g., MaxEnt learners (Staubs 2011; Nirheche 2024).
- I used HGR (Staubs 2011) in to handle the definite article assimilation example (Nirheche 2025).

# SUMMARY

- We’ve seen that, as datasets grow, manual calculation becomes impossible.
- Tools like OT-Help ensure **consistency** and **reproducibility**.
- **OT-Help** is an example tool that is accessible, web-based, and handles both Ranking (OT) and Weighting (HG).
- Resources for OT-help 2.0:
  - Webpage: https://people.umass.edu/othelp/
  - Manual: https://people.umass.edu/othelp/OTHelp.pdf

# THANK YOU