Today's topics:

- **OT fundamentals**
- **Basics of OT formalism**

Background preparation:

- McCarthy (2007), sec 1–4; focus on sec 3–4

0. Today's objectives

After today's class, you should be able to:

- Explain key concepts of OT
  - Constraint-based
  - Parallel
  - GEN, CON, EVAL
- Work with tableaus and use relevant notation
  - Comparative tableaus
  - Ranking arguments

1. OT is constraint-based and parallel

- Classic OT
  - is a ***constraint-based*** and ***parallel*** phonological model
  - rather than a ***rule-based*** and ***serial/derivational*** phonological model

1. OT is constraint-based and parallel

- **rule**
  - Formally: A rule **identifies** a structural description and **imposes** a structural **change**
    - Abstractly, “A → B / C__D”
  - **Two things** are packaged together in a rule:
    - Rules not only **identify** a certain structure
    - They also specify **how** the grammar should **change** it

1. OT is constraint-based and parallel

- **constraint**
  - An OT formalization of a phonological "target"
  - A constraint identifies a certain phonological structure to which it assigns a **violation**
    - We still need phonological **representations** in our model: what entities can constraints refer to?
    - More later about what makes a *plausible* constraint
  - A constraint says only, "Don't be X" — it does not say *what to do* to something that *is* X
    - How is the *what to do* decided?

1. OT is constraint-based and parallel

- **constraint ranking**
  - Classic OT constraints are **ranked** in a hierarchy
  - A higher-ranked constraint takes precedence over a lower-ranked constraint in choosing the winning output form

Alternative to classic OT: Harmonic Grammar (HG) and related models use constraints that are **weighted** (with numerical "penalties"). Two lower-weighted constraints can "gang up" on a higher-weighted constraint. This isn't possible when constraints are strictly ranked.

1. OT is constraint-based and parallel

- **serial, derivational**

  Some phonological models start with a UR and:

  - change it one step at a time (**serially**)
  - giving rise to a step-by-step **derivation**

1. OT is constraint-based and parallel

- **parallel**
  In classic OT, for a given input (UR):
  - the winning output candidate is chosen in a single step (in **parallel**)
  - even if it differs from the input in >1 respect

Alternative to classic OT: Serial OT and Serial HG apply OT or HG in multiple, serial steps. Take the input, compare all candidates that are *one step away* from the input, and select the winning intermediate output. Then take that intermediate output as the new input; repeat this process until the winning output is the same as its input.

2. The architecture of OT: McCarthy (2007)

- What are the following parts of the OT grammar?
  - GEN
  - CON
  - EVAL

- Which aspects of an OT grammar are universal, and which are language-specific?

2. The architecture of OT: McCarthy (2007)

- Other questions about sections §1–§3 of the McCarthy (2007) reading?
  - We will be discussing ideas from §4 in more depth today and in the next few classes

3. OT and input/output mapping

- How the grammar makes things happen in OT (how OT models **phonological processes**)
  - For any ***input*** form
  - The phonological grammar finds the winning (***optimal***; *most* ***harmonic***) ***output*** form, chosen from among a ***set of output candidates***
- Another way of saying this: the grammar ***maps*** each input form onto its optimal output form

3. OT and input/output mapping

- **input**
  - For now, think of an input as equivalent to a UR
  - We will see later that the notion of *input* in OT is actually broader than this

3. OT and input/output mapping

- **set of output candidates**
  - In classic OT, the *set of output candidates* is unlimited and infinite
  - However, for any given phonological analysis, only certain candidates are interesting and worth discussing explicitly
  - Phonologists usually pay the most attention to those SRs that a language might plausibly choose for the UR they are working with

3. OT and input/output mapping

- **optimal, harmonic**
  - *Optimal* means ‘best’
    - It is not a gradable adjective, so we don’t say *“A is more *optimal* than B”
  - Compare *harmonic:* “A is more *harmonic* than B”
    - *Most harmonic* means the same as *optimal*

3. OT and input/output mapping

- **to map**
  - We can conceive of an OT grammar as a function from inputs to outputs
  - So, we can talk about of *mapping* an input onto the appropriate output (as determined by the grammar)
  - This can often be more useful phrasing than thinking of “turning” an input into an output (because this phrasing assumes a derivation)

4. Constraint tableaus

- **Constraint tableau** (plural: *tableaus* or *tableaux*)
  - a formal tool for phonological analysis in OT

- Each tableau:
  - represents what is known about the constraint ranking of a particular language
  - shows the input-output mapping for a particular input under that ranking

4. Constraint tableaus

A tableau is useful for two kinds of arguments:

- #1: Make a **ranking argument**
  - If you know: the **input** and the **output**
  - A tableau can: *prove that a particular constraint ranking is necessary* for the right output to win
  - ***Comparative tableau*** format is useful for this

4. Constraint tableaus

A tableau is useful for two kinds of arguments:

- #2: Show how the **grammar** of a language operates (the "selection problem" in McCarthy (2007))
  - If you know: the **ranking**
  - A tableau can:
    - show *what output would win* for a particular assumed input
    - show *what input(s) should be postulated* to ensure that a particular output will win
  - ***Violation tableau*** format is useful for this

5. Ranking argument tableau

- Example **ranking argument** tableau: /ap/ → [ap]
  Create a tableau and add violation marks
  - Include these constraints in your analysis:
    MAX, DEP, NOCODA (see Zec 2007 reading)
  - Consider these candidates:
    [ap] (winner), [a], [a.pi], [a.i]
  - **Input** in top left cell
    Output candidates below the input
    - Indicate **winner** with arrow, ☞, etc.
    - List candidates in some useful order

5. Ranking argument tableau

- Example **ranking argument** tableau: /ap/ → [ap]

/ap/	MAX	DEP	NOCODA
► (a) a.p			
(b) a			
(c) a.pi			
(d) a.i			

As a shortcut for syllable trees, nuclei are underlined

- **Constraint violation**: '*' in cell for each
- **W / L marks**: Used in comparative tableau
  - W: This constraint ***prefers winner*** over this loser
  - L: This constraint ***prefers this loser*** over winner

5. Ranking argument tableau

- Example **ranking argument** tableau: /ap/ → [ap]

/ap/	MAX	DEP	NOCODA
► (a) ap			*
(b) a	* W		L
(c) a.pi		* W	L
(d) a.i	* W	* W	L

  - What ranking arguments are made here?
    - W: This constraint ***prefers winner*** over this loser
    - L: This constraint ***prefers this loser*** over winner

5. Ranking argument tableau

- Making **ranking arguments** from W/L marks
  - Any row with an L: this constraint will choose the wrong winner if not dominated
    - Every L in a row must be dominated by at least one W (in same row)
  - If there is no L in a row: no ranking is proven
  - If there is no W in a row: grammar is currently not choosing the correct winner

5. Ranking argument tableau

- Example **ranking argument** tableau: /ap/ → [ap]

/ap/	MAX	DEP	NOCODA
► (a) a̲p			*
(b) a̲	* W		L
(c) a̲.pi̲		* W	L
(d) a̲.i̲	* W	* W	L

  - Ranking arguments here:
    MAX » NOCODA and DEP » NOCODA
  - No ranking can be proven between MAX, DEP
    (Do you see why?)

5. Ranking argument tableau

- Ranking arguments here:
  MAX » NOCODA and DEP » NOCODA
- This ranking can be written as { MAX, DEP } » NOCODA
  - Solid/dotted lines in tableau are ambiguous:
    MAX » NOCODA too?
- Or use a Hasse (tree) diagram:

  MAX DEP
  \ /
  NOCODA

  - Lines indicate **domination** in a Hasse diagram:
    higher » lower

5. Ranking argument tableau

- Example **ranking argument** tableau: /ap/ → [ap]

/ap/	MAX	DEP	NOCODA
► (a) a.p			*
(b) a	* W		L
(c) a.pi		* W	L
(d) a.i	* W	* W	L

When **presenting** an analysis:

- **Constraints** in top row, left (highest) to right
- **Solid** line between A, B means **A » B**
- **Dotted** line means "can't determine ranking"
  - Note ambiguity — **state ranking explicitly** also

6. Violation tableau

- What **output** will a grammar choose for /ap/ if the constraint **ranking** is NoCoda » Max » Dep ?
  - Consider these candidates:
    [ap], [a], [a.pi], [a.i]
  - Create a tableau, add violation marks, and determine the winning output
  - We don't add W/L marks this time: why not?
- Don't try to use this method if you *don't already know the ranking* — this is for testing predictions of a known or assumed ranking

6. Violation tableau

- What **output** will a grammar choose for /ap/ if the constraint **ranking** is NoCoda » Max » Dep ?

/ap/	NoCoda	Max	Dep
(a) ap	*!		
(b) a		*!	
▶ (c) a.pi			*
(d) a.i		*!	*

  - '!' indicates a "fatal" violation
  - Shading is sometimes used to show that a candidate has been eliminated
    (use sparingly — can be ambiguous or hard to read!)

7. For next time

- Read McCarthy (2008), Ch 2, sec 2.1–2.2
- Try out some basic OT concepts with the Māori loanwords data set